
Deep Latent-Variable Models for Neural Text Generation

A dissertation submitted towards the degree
Doctor of Engineering
(Dr.-Ing.)
of the Faculty of Mathematics and Computer Science
of Saarland University

by
Xiaoyu Shen, M.Sc.

Saarbrücken
2021

Day of Colloquium 5th of November, 2021

Dean of the Faculty Univ.-Prof. Dr. Thomas Schuster
Saarland University, Germany

Examination Committee

Chair Prof. Dr. Vera Demberg

Reviewer, Advisor Prof. Dr. Dietrich Klakow

Reviewer, Advisor Prof. Dr. Gerhard Weikum

Reviewer Prof. Dr. Hinrich Schütze
Academic Assistant Dr. Volha Petukhova

ABSTRACT

Text generation aims to produce human-like natural language output for down-stream tasks. It covers a wide range of applications like machine translation, document summarization, dialogue generation and so on. Recently deep neural network-based end-to-end architectures have been widely adopted. The end-to-end approach conflates all sub-modules, which used to be designed by complex handcrafted rules, into a holistic encode-decode architecture. Given enough training data, it is able to achieve state-of-the-art performance yet avoiding the need of language/domain-dependent knowledge. Nonetheless, deep learning models are known to be extremely data-hungry, and text generated from them usually suffer from low diversity, interpretability and controllability. As a result, it is difficult to trust the output from them in real-life applications. Deep latent-variable models, by specifying the probabilistic distribution over an intermediate latent process, provide a potential way of addressing these problems while maintaining the expressive power of deep neural networks.

This dissertation presents how deep latent-variable models can improve over the standard encoder-decoder model for text generation. We start from an introduction of encoder-decoder and deep latent-variable models, then go over popular optimization strategies like variational inference, dynamic programming, soft relaxation and reinforcement learning. Finally, we elaborate on:

1. How latent variables can *improve the diversity of text generation* by learning holistic, sentence-level latent representations. By doing so, a latent representation can be first sampled, from which diverse text can be generated. We present effective algorithms to simultaneously train the representation learning and text generation via variational inference. Further, to address the uni-modal and inconsistency limitations of variational inference, we propose a wake-sleep variation and mutual information-enhanced training objective. Experiments show they outperform standard variational inference and non-latent-variable models on the dialogue generation tasks.
2. How latent variables can *improve the controllability and interpretability of text generation* by adding finer-grained, latent specifications on the intermediate generation process. We illustrate using latent variables to stand for word alignment, content selection, text segmentation and field-segment correspondences. We derive efficient training algorithms for them so that the text generation can be explicitly controlled by manipulating the latent variables, which are human interpretable by definition.
3. How to *overcome the sparsity of training samples* by treating non-parallel text as latent variables. The training can be performed like in the standard EM algorithm which is stable to converge. We show it can be successfully applied

in dialogue generation and significantly enrich the topic of generation space by utilizing non-conversational text.

ZUSAMMENFASSUNG

Textgenerierung zielt darauf ab, eine menschenähnliche Textausgabe in natürlicher Sprache für Anwendungen zu erzeugen. Es deckt eine breite Palette von Anwendungen ab, wie maschinelle Übersetzung, Zusammenfassung von Dokumenten, Generierung von Dialogen usw. In letzter Zeit werden dafür hauptsächlich End-to-End-Architekturen auf der Basis von tiefen neuronalen Netzwerken verwendet. Der End-to-End-Ansatz fasst alle Submodule, die früher nach komplexen handgefertigten Regeln entworfen wurden, zu einer ganzheitlichen Codierungs-Decodierungs-Architektur zusammen. Bei ausreichenden Trainingsdaten kann eine Leistung auf dem neuesten Stand der Technik erzielt werden, ohne dass sprach- und domänenabhängiges Wissen erforderlich ist. Deep-Learning-Modelle sind jedoch als extrem datenhungrig bekannt und daraus generierter Text leidet normalerweise unter geringer Diversität, Interpretierbarkeit und Kontrollierbarkeit. Infolgedessen ist es schwierig, der Ausgabe von ihnen in realen Anwendungen zu vertrauen. Tiefe Modelle mit latenten Variablen bieten durch Angabe der Wahrscheinlichkeitsverteilung über einen latenten Zwischenprozess eine potenzielle Möglichkeit, diese Probleme zu lösen und gleichzeitig die Ausdruckskraft tiefer neuronaler Netze zu erhalten.

Diese Dissertation zeigt, wie tiefe Modelle mit latenten Variablen Texterzeugung verbessern gegenüber dem üblichen Encoder-Decoder-Modell. Wir beginnen mit einer Einführung in Encoder-Decoder- und Deep Latent Variable-Modelle und gehen dann auf gängige Optimierungsstrategien wie Variationsinferenz, dynamische Programmierung, Soft Relaxation und Reinforcement Learning ein. Danach präsentieren wir Folgendes:

1. Wie latente Variablen Vielfalt der Texterzeugung verbessern können, indem ganzheitliche, latente Darstellungen auf Satzebene gelernt werden. Auf diese Weise kann zunächst eine latente Darstellung ausgewählt werden, aus der verschiedene Texte generiert werden können. Wir präsentieren effektive Algorithmen, um gleichzeitig das Lernen der Repräsentation und die Texterzeugung durch Variationsinferenz zu trainieren. Um die Einschränkungen der Variationsinferenz bezüglich Uni-Modalität und Inkonsistenz anzugehen, schlagen wir eine Wake-Sleep-Variation und ein auf Transinformation basierendes Trainingsziel vor. Experimente zeigen, dass sie sowohl die übliche Variationsinferenz als auch nicht-latente Variablenmodelle bei der Dialoggenerierung übertreffen.
2. Wie latente Variablen die Steuerbarkeit und Interpretierbarkeit der Texterzeugung verbessern können, indem feinkörnigere latente Spezifikationen zum Zwischengenerierungsprozess hinzugefügt werden. Wir veranschaulichen die Verwendung latenter Variablen für Wortausrichtung, Inhaltsauswahl, Textsegmentierung und Feldsegmentkorrespondenz. Wir leiten für sie effiziente Train-

ingsalgorithmen ab, damit die Texterzeugung explizit gesteuert werden kann, indem die latente Variable, die durch ihre Definition vom Menschen interpretiert werden kann, manipuliert wird.

3. Überwindung der Seltenheit von Trainingsmustern durch Behandlung von nicht parallelem Text als latente Variablen. Das Training kann wie beim Standard-EM-Algorithmus durchgeführt werden, der stabil konvergiert. Wir zeigen, dass es bei der Dialoggenerierung erfolgreich angewendet werden kann und den Generierungsraum durch die Verwendung von nicht-konversativem Text erheblich bereichert.

ACKNOWLEDGEMENTS

First and foremost I would like to express my sincerest gratitude to my advisors Prof. Dr. Dietrich Klakow and Prof. Dr. Gerhard Weikum for their great supports of my PhD study. Their deep academic insights and rich knowledges helped me a lot in the research work during my PhD. More importantly, their attitudes and encouragement will be always inspiring to me in my future work. Besides, working with them, I also learned how to define, discover, address, present and publish my research works, which will be absolutely important in my future research works. In addition, I would like to show my appreciation to Prof. Dr. Verena Wolf, Dr. Mathias Humbert and Prof. Dr. Akiko Aizawa, who kindly guided me through my initial stage of research work. They always encouraged me to explore some new ideas, selflessly shared their experience and provided high-quality feedbacks, which deeply influenced my understanding on good academic activity. Thank you all for your time and effort for reviewing my work and providing valuable feedback.

In the following, I would like to thank people who I have collaborated with: Pascal Berrang, Prof. Dr. Satoshi Sekine, Hui Su, Yang Zhao, Prof. Dr. Kentaro Inui, Prof. Dr. Jun Suzuki, Prof. Dr. Vera Demberg, Ernie Chang, Liqiang Wang and Yafang Wang. I also learned a lot from you and grewed a lot working with you, from your academic ability or personality. Besides, I would like to thank many other members (or former members) in the LSV group of Saarland University, including Youssef Oualil, Clayton Greenberg, Mittul Singh, Thomas Trost, Aditya Mogadala, David Ifeoluwa Adelani, Michael a. Hedderich, Marius Mosbach, Volha Petukhova, Thomas Kleinbauer, Dana Ruiter, Dawei Zhu, Alexander Blatt, Anupama Chingacham, Ali Davody, Fech Scen Khoo, etc. However, the people I would like to show appreciations are not only limited to the mentioned people, but all LST Saarland and D5 members in the MPI-Informatics. Thanks for your talents and hard working, maintaining excellent academic circumstance. Your every amazing presentations and discussions are memorable to me.

Next, I would like to thank our secretary Angelika Obree and system administrator Nicolas Louis. Thanks a lot for your careful and patient work. You always kindly reminded me of many important issues and gave me a lot of suggestions, which were of great for my life in Saarbrücken. Besides, I would like to thank IT service staff, for their work on providing a stable and wonderful environment. I would also like to thank Michelle Carnell and Susnne Vohl from the graduate school office who patiently helped me settle up in Saarbrücken and provided help through my first year of coursework. I am funded by the IMPRS-CS fellowship. The works I finished throughout my PhD phase have been supported by the DFG collaborative research center SFB 1102, Research Grants Council of Hong Kong (PolyU 152036/17E, 152040/18E), the National Natural Science of China under Grant No. 61602451, JSPS KAKENHI Grant Number JP19104418, AIRPF Grant Number 30A1036-8 and SFB

248 "Foundations of Perspicuous Software Systems" (E2). I am very grateful to all the above generous support.

Finally, I would like to thank my parents. I could not have pursued my PhD without your supports and love.

CONTENTS

1	Introduction	1
1.1	Encoder-Decoder	2
1.2	Deep latent-variable model	4
1.3	Contributions and Thesis Outline	5
1.4	Publications	6
2	Background on Optimizing Latent-Variable Models	9
2.1	Direct Marginalization	10
2.2	Variational Approximation	16
2.3	Improvement of Variational Approximation	20
2.4	Generative Adversarial Networks	25
2.5	Improvement of GAN	33
2.6	When reparameterization not applicable	41
2.7	VAE in Natural Language Generation	47
2.8	Summary	52
3	VAE with Improved decoder	53
3.1	Introduction	53
3.2	VED in Dialogue Generation	54
3.3	Improving Variational Encoder-Decoders	58
3.4	Experiments	61
3.5	Conclusion	67
4	VAE with Mutual Information Maximization	69
4.1	Introduction	69
4.2	Model Structure	71
4.3	Relationship to Existing Methods	75
4.4	Experiments	76
4.5	Additional Information	82
4.6	Conclusion	84
5	Latent Variable as Content Selection	85
5.1	Introduction	85
5.2	Background and Notation	87
5.3	Content Selection	87
5.4	Related Work	92
5.5	Experiments	92
5.6	Conclusion	100

6	Latent Variable as Alignment	101
6.1	Introduction	101
6.2	Background	103
6.3	Generalized Pointer Generator (GPG)	104
6.4	Related Work	107
6.5	Experiments and Results	107
6.6	Conclusion	116
7	Latent Variable as Segmentation and Correspondence	117
7.1	Introduction	117
7.2	Related Work	119
7.3	Background: Data-to-Text	120
7.4	Approach	120
7.5	Experiment Setup	125
7.6	Results	126
7.7	Conclusion	130
7.A	Error Analysis	131
7.B	Controlling output structure	132
8	Diversify Dialogue with Non-Paired Text	135
8.1	Introduction	135
8.2	Related Work	137
8.3	Dataset	138
8.4	Approach	138
8.5	Experiments	142
8.6	Results	144
8.7	Conclusion	148
9	Conclusions and Future Prospects	149
9.1	Summary of Thesis	149
9.2	Challenges	150
	List of Figures	153
	List of Tables	157
	Index	161
	Bibliography	161

TEXT generation is the subfield of artificial intelligence and computational linguistics that focuses on computer systems that can produce understandable texts in English or other human languages (Reiter and Dale, 2000). Specifically, we define text generation as the task of “*Given some input information, provide coherent human-like text as the output, based on task-specific requirement*”. In this thesis, we make no assumption about the input. It can be of arbitrary format or even multiple-sourced. Depending on the input context and requirement, it covers a broad range of different tasks such as:

- machine translation (Och *et al.*, 1999): translate from one language to another
- text summarization (Clarke and Lapata, 2010): summarize the gist of the input into concise sentences
- data-to-text (Reiter and Dale, 2000): describe structured knowledge with human languages
- dialogue generation (Vinyals and Le, 2015): build chatbots that can converse with people like human beings
- image captioning (Xu *et al.*, 2015): describe the contents in an image with human languages
- grammar error correction (Dale *et al.*, 2012): correct the grammar errors in the input and output the corrected sentence
- paraphrase generation (Bannard and Callison-Burch, 2005): generate paraphrases of the input sentence
- style transfer (Shen *et al.*, 2017b): transfer the style of the input (e.g., positive, female, polite) sentence into another (e.g., negative, male, impolite)

Early text generation systems are normally rule-based and contain pipe-line structures of content determination, text structuring, sentence aggregation, etc. (Thompson, 1977; Goldberg *et al.*, 1994; Reiter *et al.*, 1995), which requires enormous human labor to design hand-crafted rules. The domain knowledge required for one task generally cannot be easily transferred to another one. The recent advances in deep learning have given it a new impetus, where a single encoder-decoder neural architecture is devised to merge the traditional pipe-line architecture (Sutskever *et al.*, 2014). Relying on the powerful generalizing capability of deep neural networks, neural text generation systems can effectively learn the underlying generating distribution

of human languages from large-scale dataset without pre-specified domain knowledge (Bahdanau *et al.*, 2015; Xu *et al.*, 2015; Rush *et al.*, 2015; Wen *et al.*, 2015; Shen *et al.*, 2017c; Zhao *et al.*, 2017d, 2018b).

1.1 ENCODER-DECODER

Given a specific text generation task, denote by $X = x_1, x_2, \dots, x_m$ as the input information and $Y = y_1, y_2, \dots, y_n$ as the output text. m and n are the length of the input and output respectively. All text generation tasks can be converted into an encoder-decoder structure, where an encoder encodes the input information into machine-readable continuous vectors, based on which the decoder estimates the probability of the target output word by word ¹. Compared with traditional statistical methods based on n-gram counting and smoothing (Chen and Goodman, 1996), neural network-based encoder-decoder architectures can capture much longer dependency relations and generalize better at unseen scenarios. Normally, the probability is computed autoregressively as:

$$p_\theta(Y|X) = \prod_{t=1}^n p_\theta(y_t|X, y_{<t}) \quad (1.1)$$

The encoder encodes each x_i into a vector h_i . At each time step t , the previously-generated words are compressed into hidden state d_t of the decoder. The input information is usually represented by means of the attention mechanism (Bahdanau *et al.*, 2015), which is a weighted average of source vectors. The output probability is defined as:

$$\begin{aligned} p_\theta(y_t|X, y_{<t}) &= p_\theta(y_t|A_t, d_t) \\ p_\theta(y_t|A_t, d_t) &= \text{softmax}(W_1 d_t + W_2 A_t) \\ A_t &= \sum_i \alpha_{t,i} h_i \\ \alpha_{t,i} &= \frac{e^{f(h_i, d_t)}}{\sum_j e^{f(h_j, d_t)}} \end{aligned} \quad (1.2)$$

A_t is the attention vector at time step t . f is a score function to compute the similarity between h_i and d_t (Luong *et al.*, 2015). W_1 and W_2 are learnable parameters. The attention mechanism can be seen as performing an approximation of the latent alignment (Kim *et al.*, 2017). If the input information at position i is important for the prediction of word y_t , we expect the similarity function $f(h_i, d_t)$ should be large and so as the weight $\alpha_{t,i}$. Figure 1.1 illustrates the typical architecture of an encoder-decoder model for text generation.

¹There has been a surge of research interest in unifying all NLP tasks, including understanding and generation, into a single encoder-decoder architecture (McCann *et al.*, 2019; Radford *et al.*, 2019; Raffel *et al.*, 2019). Although this dissertation focuses on text generation, relevant techniques can be easily extended to text understanding tasks by replacing the decoder with different classifiers.

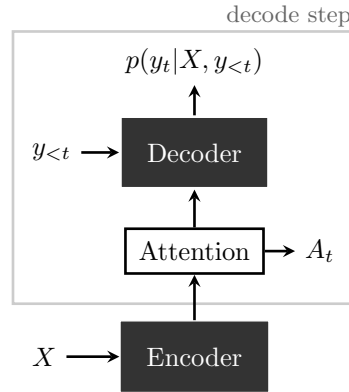


Figure 1.1: Illustration of a typical encoder-decoder architecture for text generation. Figure taken from (Xie, 2017).

The choice of the encoder can differ from task to task, e.g. residual networks for images (He *et al.*, 2016b), graph convolutional networks (Kipf and Welling, 2017) for structured data and transformers (Vaswani *et al.*, 2017) for sequence input. The decoder is usually implemented as the LSTM (Hochreiter and Schmidhuber, 1997) or transformer neural network for their excellent capability at modelling sequential dependencies.

Details can differ from our definitions above in down-stream variants. For example, non-autoregressive models decode output in parallel instead of recursively to speed up the process (Gu *et al.*, 2018a; Lee *et al.*, 2018). The output softmax can be augmented with weight tying (Press and Wolf, 2017; Inan *et al.*, 2017) or pointer generators (Gu *et al.*, 2016; Gulcehre *et al.*, 2016). Sparsity constraints can be imposed to improve the interpretability and efficiency of the attention mechanism (Martins and Astudillo, 2016; Child *et al.*, 2019). The essential idea is to estimate the probability of $p_\theta(Y|X)$ with an end-to-end encode-decode process. Specifically, in this thesis, we focus on models with *tractable density estimations* of $p_\theta(Y|X)$ and optimized by *maximum likelihood*. Namely, in the training stage, model parameters are updated to maximize the likelihood of parallel input-output pairs, defined by $p_\theta(Y|X)$. Though other variants based on generative adversarial networks (Yu *et al.*, 2017; Fedus *et al.*, 2018; Zhou *et al.*, 2020), which targets a lower bound of Jensen-Shannon divergence, or energy-based models (Schmidt *et al.*, 2019; Parshakova *et al.*, 2019; Deng *et al.*, 2020), which directly estimate the sequence-level density albeit with intractable partition functions, have shown promising results as for improving the coherence and diversity of generated text, their training process is significantly more complex and unstable. When combined with latent-variable models, the training difficult would be aggregated and therefore very limited attempts have been done.

Though state-of-the-art performances have been achieved, seq2seq models have been constantly criticized for the (1) low diversity of outputs, (2) poor generalization under limited or noisy supervised data and (3) lack of interpretability and controllability (Sriram *et al.*, 2018; Dušek *et al.*, 2020; Holtzman *et al.*, 2020). Latent variable models, by specifying the probabilistic distribution over a intermediate latent process,

provide a tool to address these problems in a principled way. By integrating the latent variable framework into the encoder-decoder architecture (denoted as “deep latent-variable model”), we are able to inject additional stochasticity, prior knowledge or structured dependencies without sacrificing the model capacity, thereby effectively alleviating the above-mentioned three problems. Recently proposed training algorithms like variational inference (Kingma and Welling, 2014; Rezende *et al.*, 2014) and automatic differentiation tools like PyTorch (Paszke *et al.*, 2019) also make it convenient to efficiently train them on large-scale datasets. As a result, improving text generation with deep latent-variable models have been a hot topic in recent years, with applications spreading across different domains (Bowman *et al.*, 2016; Serban *et al.*, 2017d; Deng *et al.*, 2018; Shen *et al.*, 2019b; He *et al.*, 2020).

1.2 DEEP LATENT-VARIABLE MODEL

The encoder-decoder architecture defined above belongs to the so-called fully visible belief networks (FVBN) (Frey *et al.*, 1996; Frey, 1998), which use the chain rule of probability to decompose a probability distribution over an n -dimensional vector into a product of one-dimensional probability distributions (Goodfellow, 2016). The main feature of FVBN is that all variables (for text generation, the input X and output Y) are *explicitly observable*. The only source of stochasticity is the data distribution itself. Given a fixed input X , we will always have a deterministic estimation of $p_\theta(Y|X)$.

Latent variable models, on the contrary, hypothesise there exist latent variables which are *not directly observable and are assumed to affect the response variables*. Applied to text generation, the decoder needs to define two distributions: the prior distribution $p_\theta(z|X)$ of the latent variable z and the likelihood distribution $p_\theta(Y|X, z)$ of the output text. By making it “deep”, $p_\theta(z|X)$ and $p_\theta(Y|X, z)$ are built upon the deep neural network structure to take advantage its modelling power. Y is now dependent not only on the input X but also the stochastic latent variable z . To generate text, latent variables are first sampled from $p_\theta(z|X)$, then text are decoded based on $p_\theta(Y|X, z)$. The process is intuitive since languages are not generated out of blue for humans. Before producing a text, we need to think about the structure, topic, style and so on. z is used to stand for these implicit influencing factors. As these factors are latent and no ground-truth supervision is available, the optimization is usually proceeded by marginalizing over z to find the latent distribution that can best explain the observable $X - Y$ training pairs:

$$p_\theta(Y|X) = \sum_z p_\theta(z|X) p_\theta(Y|X, z) \quad (1.3)$$

The model is then trained still based on the maximum likelihood objective with the density estimated as in Eq. 1.3. Figure 1.2 illustrates the graphical model for deep latent-variable models in text generation.

While latent-variable models have a long history in natural language tasks like statistical alignment (Brown *et al.*, 1993) and topic modelling (Blei *et al.*, 2003), early research focuses more on the *inference* (i.e., the posterior probability $p_\theta(z|X, Y)$)

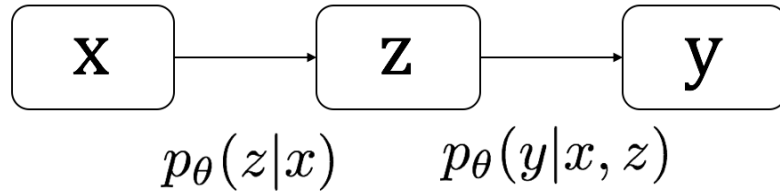


Figure 1.2: Graphical model of deep latent-variable model. θ is parameterized by deep neural networks

rather than the *generation* performance (i.e., improving the estimation of $p_\theta(z|X)$ and $p_\theta(Y|X, z)$). To make the inference tractable, independence assumptions (like 0th or 1st order markov) are usually made for the generation process. As a result, they often excel at inferring latent factors but perform rather poorly as generative models (Daumé III and Marcu, 2005; Angeli *et al.*, 2010). By utilizing modern deep neural networks to model the generative distribution, unbounded dependencies across words can be captured so that the generation performance can be greatly improved. Nonetheless, the power of deep generative models also comes at a cost: latent variables might be ignored given sufficiently powerful generative models, yielding a random inference distribution $p_\theta(z|X, Y)$ (Bowman *et al.*, 2016). In practice, for deep latent-variable models, there is often a trade-off between the inference and generation, which can be tailored according to specific task requirements (Chen *et al.*, 2017; Alemi *et al.*, 2018; Shen *et al.*, 2019b). In this dissertation, as we target at text generation, we will mostly cover techniques improving the generation performance, even if it might sacrifice the inference performance.

Deep neural networks can in theory capture unbounded dependencies to improve the generation, but it comes with the cost: Eq. 1.3 is usually intractable and no analytical solution exists. Even accurate estimations become extremely expensive since running neural networks sequentially over text is itself very time-consuming. In the next section, we will go over popular strategies to address this optimizing challenge and discuss under which circumstances they should be used.

1.3 CONTRIBUTIONS AND THESIS OUTLINE

As mentioned above, non-latent encoder-decoder models suffer from the problems of (1) low diversity of generations, (2) poor controllability and interpretability, and (3) requiring large-amount of supervised data. Our main contribution is devising novel latent-variable models within acceptable computational complexity to address the three problems in text generation. Specifically, we propose models to:

1. *improve the diversity of text generation* by learning holistic, sentence-level latent representations. By doing so, a latent representation can be first sampled, from which diverse text can be generated. We present effective algorithms

to simultaneously train the representation learning and text generation via variational inference. Further, to address the uni-modal and inconsistency limitations of variational inference, we propose a wake-sleep variation (Chapter 3) and mutual information-enhanced (Chapter 4) training objective. Experiments show they outperform standard variational inference and non-latent-variable models on the dialogue generation tasks.

2. *improve the controllability and interpretability of text generation* by adding finer-grained, latent specifications on the intermediate generation process. We illustrate using latent variables to stand for word alignment (Chapter 6), content selection (Chapter 5), text segmentation and field-segment correspondences (Chapter 7). We derive efficient training algorithms for them so that the text generation can be explicitly controlled by manipulating the latent variable, which are human interpretable by its definition.
3. *overcome the sparsity of supervised data* by treating non-parallel text as latent variables. The training can be performed like in the standard EM algorithm which is stable to converge. We apply it on dialogue generation and significantly enrich the generated responses utilizing only non-conversational text (Chapter 8).

Before delving into detailed works, we first go over popular optimization strategies in Chapter 2, including the cases when exact marginalization is applicable and when variational approximation must be utilized. The pros and cons of different strategies are explained to get a holistic overview of latent-variable models.

Finally, in Chapter 9, we draw some conclusions and summarize how latent-variable models can be used to improve text generation.

1.4 PUBLICATIONS

The made contributions mentioned in this dissertation centers around the following publications that includes collaborations of many people. The following list details the authorship, and the papers in which the results were published.

[6] *Improving Variational Encoder-Decoders in Dialogue Generation.*

Xiaoyu Shen*, Hui Su*, Shuzi Niu, and Vera Demberg.

In Proc. of the AAAI Conference on Artificial Intelligence (AAAI), 2018.

The initial idea came after a collaboration work with Hui. After some preliminary discussion, I designed the algorithm for the improved variational encoder-decoder, implemented the system and wrote the final paper. Hui helped evaluate the systems and visualize the T-SNE graph for latent variables. He also constantly provided suggestions through the paper writing. Shuzi and Vera both provided suggestions during our discussions. The details of this work will be discussed at Chapter 3.

[5] *Nexus Network: Connecting the Preceding and the Following.*

Xiaoyu Shen*, Hui Su*, Wenjie Li, and Dietrich Klakow.

In Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018.

Wenjie initializes the idea of utilizing the future utterances in a dialogue. Hui implemented the first version of the system and the evaluation code. Based on it, I designed the algorithm, derived the formulas and drew its connection with the mutual information theory. I also wrote most part of the paper. Dietrich helped check the derivations and proofread the work. The details of this work will be discussed at Chapter 4.

[4] *Select and Attend: Towards Controllable Content Selection in Text Generation.*

Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow, Satoshi Sekine

In Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.

I designed the algorithms, implemented the whole system and wrote the final paper. Jun suggested using the bpe subword to remove the OOV problem and proofread the paper. Kentaro suggested testing it on the sentence compression task. Satoshi raised the initial idea of the data-to-text task. Hui and Dietrich followed up through discussions. The details of this work will be discussed at Chapter 5.

[3] *Improving Latent Alignment in Text Summarization by Generalizing the Pointer Generator.*

Xiaoyu Shen, Yang Zhao, Hui Su, Dietrich Klakow.

In Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.

I proposed the initial idea, designed the algorithm, implemented the system and wrote the final paper. Yang and Dietrich joined the discussions and provided feedbacks. Hui helped draw the graphs and proofread the paper. The details of this work will be discussed at Chapter 6.

[2] *Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence.*

Xiaoyu Shen, Ernie Chang, Hui Su and Dietrich Klakow

In Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL), 2020.

I designed the idea, derived the formulas of the marginal likelihood, implemented the system and wrote the final paper. Ernie helped process the dataset and evaluate the system. Dietrich provided feedbacks, introduced some professionals in the data-to-text area to help answer my questions, and helped proofread the paper. Hui joined the related discussions. The details of this work will be discussed at Chapter 7.

[1] *Diversifying Dialogue Generation with Non-Conversational Text.*

Hui Su*, Xiaoyu Shen*, Sanqiang Zhao, Xiao Zhou, Pengwei Hu, Rongzhi Zhong, Cheng Niu and Jie Zhou.

In Proc. of the Annual Meeting of the Association for Computational Linguistics

(ACL), 2020.

I came up with the idea, designed the training algorithms and decide which baseline algorithms we should compare with. I also wrote the final paper. Hui implemented the baseline and our systems, run the whole evaluations. Xiao helped crawl and filter the non-conversational dataset. Sanqiang, Pengwei, Rongzhi, Cheng and Jie joined the discussions and provided feedbacks. The details of this work will be discussed at Chapter 8.

THE main difficulty of optimizing deep latent variable models lies in computing the marginalization in Eq. 1.3. When using deep neural networks to parameterize $p_\theta(z|X)$ and $p_\theta(Y|X, z)$, the marginalization is usually intractable. The general idea of optimizing deep latent-variable models can be summarized as follows:

1. When computing the marginalization in Eq. 1.3 is tractable, optimize the exact marginal likelihood by gradient descent. To make it tractable, we can make some necessary simplification and assumptions, which will be discussed in Section 2.1.
2. When the marginal likelihood of Eq. 1.3 is intractable and difficult to simplify, estimate it with variational approximations then optimize over the estimated marginal likelihood. Specifically, when reparameterization tricks (Kingma and Welling, 2014) are applicable, we can train the model like variational autoencoders, which will be discussed in Section 2.2
3. When reparameterization tricks cannot be easily applied, the non-differentiability problem exists. We need to resort to soft relaxation, REINFORCE or EM algorithm, which we will cover at Section 2.6. We will also discuss the potentials and challenges of applying them to text generation.

In practice, one should always try direct marginalization if possible. Though it has been reported that with careful initialization, variational approximations can even outperform exact marginalization by a small margin (Deng *et al.*, 2018), variational methods are significantly trickier to train and highly depend on a good initialization and hyperparameter tuning. Figure 2.1 illustrates how we can optimize latent-variable model. In the following section, we will introduce these optimization techniques and explain them with example applications.

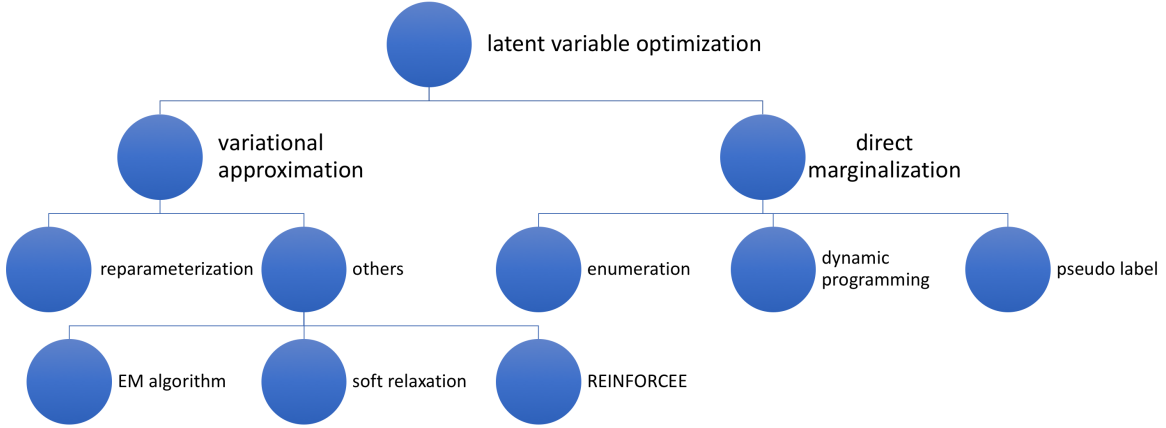


Figure 2.1: Illustration of optimization technique for latent-variable models

2.1 DIRECT MARGINALIZATION

When the exact marginalization is tractable, the optimization is straightforward. We can perform gradient descent on model parameters θ to maximize the likelihood of training samples:

$$\max_{\theta} \mathbb{E}_{X,Y} \log p_{\theta}(Y|X) = \max_{\theta} \mathbb{E}_{X,Y} \log \sum_z p_{\theta}(Y|X, z) p_{\theta}(z|X) \quad (2.1)$$

The gradient with respect to θ is given by:

$$\begin{aligned} \nabla_{\theta} \mathbb{E}_{X,Y} \log p_{\theta}(Y|X) &= \mathbb{E}_{X,Y} \frac{\nabla_{\theta} p_{\theta}(Y|X)}{p_{\theta}(Y|X)} \\ &= \mathbb{E}_{X,Y} \sum_z \frac{\nabla_{\theta} (p_{\theta}(Y|X, z) p_{\theta}(z|X))}{p_{\theta}(Y|X)} \\ &= \mathbb{E}_{X,Y} \sum_z \frac{p_{\theta}(Y|X, z) p_{\theta}(z|X)}{p_{\theta}(Y|X)} \nabla_{\theta} (\log(p_{\theta}(Y|X, z) p_{\theta}(z|X))) \\ &= \mathbb{E}_{X,Y} \mathbb{E}_{z \sim p_{\theta}(z|X, Y)} \nabla_{\theta} (\log(p_{\theta}(Y|X, z) p_{\theta}(z|X))) \end{aligned} \quad (2.2)$$

The last line of Eq 2.2 shows its connection with the generalized expectation-maximization (EM) algorithm (Dempster *et al.*, 1977; Neal and Hinton, 1998). Maximizing the marginal likelihood is equal to the M-step of the EM algorithm, where $p_{\theta}(Y|X, z)$ and $p_{\theta}(z|X)$ are optimized according to the posterior distribution $p_{\theta}(z|X, Y)$ defined by the current parameter θ (Salakhutdinov *et al.*, 2003; Sutton *et al.*, 2012).

As the maximization step cannot be done analytically for deep neural networks, stochastic gradient descent is performed stepwise as an approximation. By means of the modern automatic differentiation tools for neural networks, we avoid the necessity to calculate the posterior distribution manually. In practice, once the marginal likelihood is computed, we can optimize over it by simply calling the automatic backpropagation function (Eisner, 2016; Kim *et al.*, 2018a).

2.1.1 Enumeration

In the simplest case, the exact marginalization is possible by direct enumeration. This usually happens for discrete latent variables following a categorical distribution. A typical example is the pointer generator (Gu *et al.*, 2016; See *et al.*, 2017; Merity *et al.*, 2017).

Under the pointer generator, the output probability is not simply a softmax distribution over a fixed vocabulary, but further allows the model to directly copy words from the source input. It can be especially beneficial when handling rare or unseen words in the testing stage. At each decoding step t , the model first computes a generation probability $p_{gen} \in [0, 1]$. p_{gen} is the probability of enabling the generation mode instead of the point mode. In the generation mode, the model computes the probability over the whole vocabulary. In the point mode, the model computes which source word to copy based on the attention distribution a_t . The final probability $p_\theta(y_t|X, y_{<t})$ is computed as a combination of the generation probability and the point probability:

$$\begin{aligned} p_{gen} &= \sigma(\text{MLP}_g([d_t \circ A_t])) \\ p_{vocab} &= \text{softmax}(W_1 d_t + W_2 A_t) \\ p_\theta(y_t|X, y_{<t}) &= p_{gen} p_{vocab}(y_t) + (1 - p_{gen}) \sum_i a_{t,i} \delta(y_t|x_i) \\ \delta(y_t|x_i) &= \begin{cases} 1, & \text{if } y_t = x_i. \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

where σ is a sigmoid function and MLP_g is a learnable multi-layer perceptron. \circ denotes vector concatenation. A_t is the attention vector computed as in Eq 1.2. The graphical model of the pointer generator is depicted in Figure 2.2.

There are two types of latent variables in the pointer generator: a binary variable to decide whether to enable the generation mode or point mode, and a categorical variable $a_{t,i}$ to decide which position to copy suppose the point mode is entered. Since $\delta(y_t|x_i)$ is a simple indicator function, the marginal likelihood can be cheaply computed by enumerating over all paths of latent variables.

Mixture of experts (MoE) model (Quandt, 1972; Jacobs *et al.*, 1991) is another popular latent-variable model where the exact enumeration is applicable to compute the marginal likelihood. In this case, the latent variable can be considered as a policy network to decide which expert should be assigned to for different inputs. As for text generation, people have tried improving the diversity of generations by

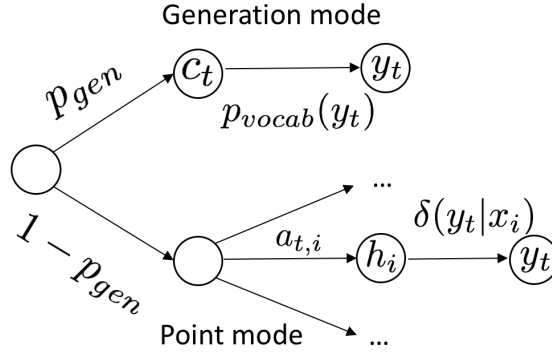


Figure 2.2: Graphical model of the pointer generator.

sentence-level MoE (Lee *et al.*, 2016; He *et al.*, 2018; Shen *et al.*, 2019a) or enhancing the decoder with word-level MoE (Yang *et al.*, 2017a, 2018; Wu *et al.*, 2019), both can be easily optimized by enumerating over all possibilities to compute the marginal likelihood.

In some cases, the exact enumeration is expensive, but can be reasonably estimated by enumerating only over the top- k modes. The exact marginalization in Eq. 1.3 is changed to:

$$\begin{aligned} & \max_{\theta} \mathbb{E}_{X,Y} \log \sum_z p_{\theta}(Y|X,z) p_{\theta}(z|X) \\ & \approx \max_{\theta} \mathbb{E}_{X,Y} \log \sum_{z \in \text{TopK}(p_{\theta}(z|X))} p_{\theta}(Y|X,z) p_{\theta}(z|X) \end{aligned} \quad (2.3)$$

This top- k approximation is widely adopted when the latent variable distribution $p_{\theta}(z|X)$ is expected to be sparse and only a few modes dominate, which is usually the case for discrete latent variables following a categorical distribution, e.g. alignment correspondence in machine translation (Shankar *et al.*, 2018; Shankar and Sarawagi, 2019). Since the conditional likelihood $p_{\theta}(Y|X,z) p_{\theta}(z|X)$ for z not belonging to the top- k set is expected to be vanishingly small, this usually yields a good approximation of exact marginalization.

Direct enumeration is easy to implement and usually leads to decent performance. However, it is applicable only for very limited scenarios. For continuous latent variables, or discrete variables with complex dependency relations, we have to resort to other methods.

2.1.2 Dynamic Programming

Sometimes there is structured dependency among latent variables and thereby the computational cost of direct enumeration will be exponentially high. By making appropriate Markov assumptions about the dependency relation, it is possible to efficiently compute the marginal likelihood with dynamic programming under linear complexity. Some classical latent-variable models like hidden markov model, hidden

semi-markov model and conditional random field all follow this category. It is straightforward to extend them “deep” by parameterizing the transition, emission probability and potential functions with deep neural networks. As an example, Figure 2.3 illustrates the graphical model for the 1st order HMM word alignment model. It has been used in statistical machine translation to induce the word alignment between the source and target language (Och *et al.*, 1999). Nowadays

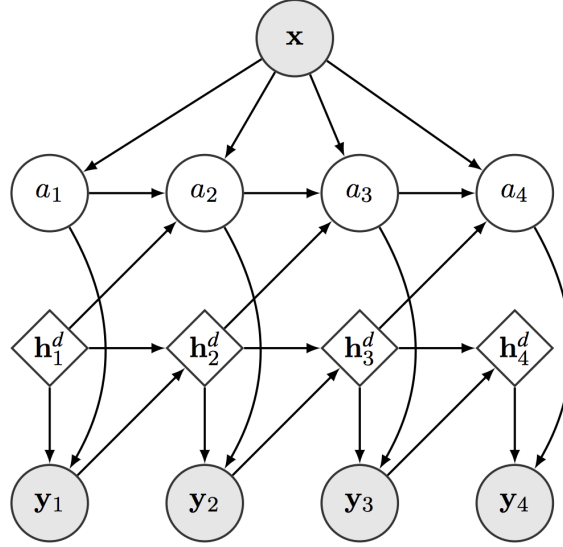


Figure 2.3: Graphical model of the HMM word alignment model. h_t^d is the hidden state of the decoder at time t which encapsulates all the history words y_1, \dots, y_{t-1} . Figure taken from (Wu *et al.*, 2018)

many works incorporate this latent alignment relation into neural encoder-decoder models, yielding comparable or better performance while producing interpretable word alignment. At every time step t , the generation process of word y_t is:

1. Select a source word a_t to align based on the categorical distribution $p(a_t|h_t, a_{t-1})$, where a_{t-1} is the aligned word at the time stamp of $t-1$ and h_t is the hidden state of the decoder which encapsulates all the history words y_1, \dots, y_{t-1}
2. Generate word y_t based on the categorical distribution $p(y_t|h_t, a_t)$, where a_t is the aligned source word selected based on $p(a_t|h_t, a_{t-1})$ and $p(y_t|h_t, a_t)$ is the output distribution of y_t conditioned on the aligned source word and history words y_1, \dots, y_{t-1} .

Here a 1st-order Markov assumption is made for the alignment probability $p(a_t|h_t, a_{t-1})$ where a_t conditions only on the previous aligned position a_{t-1} but irrelevant with a_1, \dots, a_{t-2} . In this case, we can efficiently compute the marginal likelihood with dynamic programming. Specifically, with the forward variable $\beta_{t,i} = p(y_t, a_t = i|h_t)$.

We can go through the recursion:

$$\begin{aligned}\beta_{1,i} &= p(y_1|a_1 = i, h_1)p(a_1 = i) \\ \beta_{t,i} &= p(y_t|a_t = i, h_t) \sum_{j=1}^m p(a_t = i|a_{t-1} = j, h_t)\beta_{t-1,j}\end{aligned}\tag{2.4}$$

where h_1 is the initial hidden state of the decoder, m is the length of the source input and $p(a_1 = i)$ is the alignment probability for the first target word.

The recursion goes for $i = 1, 2, \dots, n$ where n is the number of the target output. The marginal likelihood is then computed as:

$$p(y_1, \dots, y_n|x_1, \dots, x_m) = \sum_{i=1}^m \beta_{n,i}\tag{2.5}$$

The recursion in dynamic programming often leads to numerical precision problems, it is therefore necessary to define all the operations in the log space (Kim *et al.*, 2017). Note that the Markov assumption is only made for the dependency relation among latent variables a_1, \dots, a_n . The probability $p(a_t|a_{t-1}, h_t)$ and $p(y_t|a_t, h_t)$ can condition on all the history words y_1, \dots, y_{t-1} which are not latent. Therefore, the long-term dependency can usually be implicitly captured through the observable words in the target side and the performance will not significantly drop due to the simplified Markov assumption, as indicated by many works in machine translation (Shankar and Sarawagi, 2019), character transduction (Wu *et al.*, 2018) and summarization (Yu *et al.*, 2016). It is especially useful when inductive bias, e.g., monotonic or proximity bias, needs to be injected to the latent alignment (Yu *et al.*, 2016; Shankar and Sarawagi, 2019). We can easily specify these bias by adding constraints to $p(a_t|a_{t-1}, h_t)$.

Dynamic programming allows us to specify Markov dependency between latent variables yet still permits tractable inference. However, it assumes the generation of each target word conditions only on some specific part of the source input and the current latent variable only conditions on the immediate previous one, which is not the case for many applications.

2.1.3 Pseudo Label

If the dependency among latent variables and the target words are too complex to model and hard to simplify, the marginal likelihood might be intractable even with dynamic programming. For example, we might assume before generating the target output, we should first generate an outline skeleton, syntactic tree or other structured predictions. In these cases, the output needs to wait until the full intermediate predictions are produced, the dynamic programming cannot be easily applied ². Furthermore, since the intermediate prediction is highly structured, top-k

²We can still assume the Markov dependency when generating the intermediate prediction. However, this is rarely true as the intermediate prediction is highly structured. Making such simplification will affect the generation performance (Wiseman *et al.*, 2018).

sampling can easily lead to very inaccurate estimations and thus direct enumeration is challenging.

Suppose we have some oracle, coming either from heuristics, human annotations or an external parser with reasonable accuracy, we can usually treat the oracle as pseudo labels to train the latent variables in a distantly supervised way ³. Formally, the marginal likelihood becomes:

$$p(y_1, \dots, y_n | x_1, \dots, x_m) = p(y_1, \dots, y_n | x_1, \dots, x_m, z = z_o) p(z = z_o | x_1, \dots, x_m) \quad (2.6)$$

where z_o is the pseudo label obtained from the oracle. Figure 2.4 provides an example from (Wang *et al.*, 2018b), where z stands for the syntactic tree which is latent. An external parser is utilized to extract the tree structure for each target language sentence. The extracted tree is considered as the pseudo label to train the latent variable. If the external parser can reach a certain degree of accuracy, training the model in this way can usually lead to remarkably good performance.

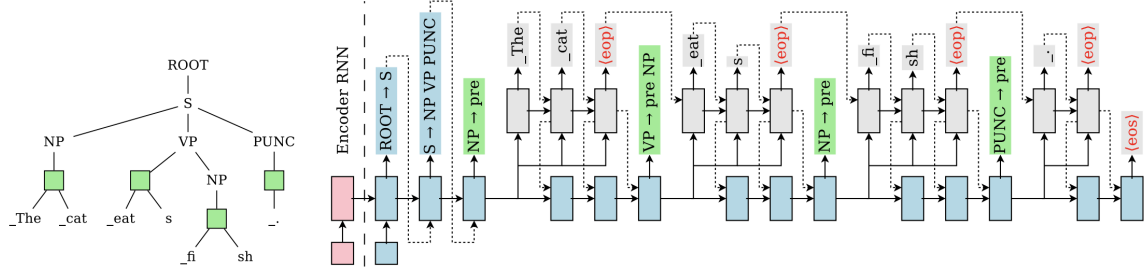


Figure 2.4: Illustration of the tree decoder for machine translation. Before translating to the target language, a syntactic tree is first generated, based on which the final output is produced from a tree-decoder. Figure taken from (Wang *et al.*, 2018b).

Similarly, we can also use z to stand for dependency trees (Elder *et al.*, 2019), content planning (Puduppully *et al.*, 2019), aspects and skeletons (Li *et al.*, 2019). In the case where the top-1 pseudo label produced from the oracle might not be accurate enough, it is also possible to use the top-k results produced from the oracle to reduce the variance. Specifically, the marginal likelihood is computed as:

$$p(y_1, \dots, y_n | x_1, \dots, x_m) = \sum_{z_o \in \text{TopK}(\text{oracle})} p(y_1, \dots, y_n | x_1, \dots, x_m, z = z_o) p(z = z_o | x_1, \dots, x_m) \quad (2.7)$$

It is similar to the top-k approximation in the direct enumeration section. However, instead of getting the top-k from the model's own prediction, it derives the top-k from an external oracle.

³In the extreme case where the oracle is 100% accurate, z becomes not "latent" any more, it basically turns into a fully visible model.

The method of distant supervision is highly effective when an oracle with reasonable accuracy exists. It essentially brings external knowledge from the oracle into the model so that we do not need to learn the latent variables from scratch. However, assuming the existence of such an oracle basically turns all latent variables into semi-explicit (since now all latent variables are clearly defined through the oracle). We need to know in advance what the latent variables are and how it affects the generation process. The construction of the oracle needs to be carefully handcrafted instead of being learnt automatically. In this sense, it has no big difference from the traditional pipeline model where each intermediate process is manually specified. Due to this limit, a typical approach is to not solely rely on the oracle, but rather only use it as a way to warm up the latent-variable model. After having a decent start, then the model can be further trained with other optimization techniques for a better performance Shen *et al.* (2019b).

2.2 VARIATIONAL APPROXIMATION

When the marginal likelihood is difficult to be computed directly, variational approximations (Kingma and Welling, 2014) are a popular strategy to estimate it by projecting the posterior distribution to a tractable parameterized distribution. Recall in Eq. 2.2 that running gradient descent over the marginal likelihood equals optimizing over the posterior distribution $p_\theta(z|X, Y)$. In the case when the marginal likelihood is intractable, $p_\theta(z|X, Y)$ will also avoid any analytical expression because the partition function basically requires computing the marginal likelihood a priori. The main idea of variational approximation is that instead of using the real posterior distribution $p_\theta(z|X, Y)$, we opt for a surrogate $q_\phi(z|X, Y)$ with a *tractable probability density*. The surrogate comes from a distribution family which is much simpler than the real posterior. Therefore, efficiently sampling over the surrogate to estimate the marginal likelihood is possible. In the meantime, to make sure the estimation is within some reasonable error bound, we need to add a regularization which pushes the surrogate to stay close to the real posterior distribution. The commonly chosen regularization imposes a KL-divergence punishment $KL(q_\phi(z|X, Y) || p_\theta(z|X, Y))$. When we use the KL-divergence regularization, it will lead to the well-known evidence lower bound (ELBO) (Jordan *et al.*, 1999): a lower bound of the likelihood in Eq. 1.3.

In this section, for notational simplicity, we assume an unconditional generation process: In the first step, a latent variable z is sampled from a prior distribution $p_\theta(z)$. In the second step, the datapoint x is generated according to the conditional probability $p_\theta(x|z)$. Both distributions are parametrised by θ and are differentiable with respect to θ and z almost everywhere. The posterior distribution of z would be $p_\theta(z|x)$. It can be easily extended to the conditional generation process where both $p_\theta(z)$ and $p_\theta(x|z)$ are conditioned on an additional context, like the case in Eq. 1.3. The graphical model of an unconditional latent-variable model is illustrated in Figure 2.5.

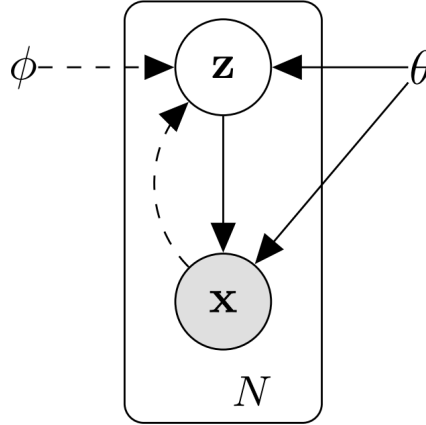


Figure 2.5: Graphical Model of unconditional Latent-variable Model

2.2.1 Evidence Lower Bound

To understand how we end up with the ELBO as an approximation of the likelihood, we start from the simplest intuition to estimate the marginal likelihood $\mathbb{E}_{p_\theta(z)} p_\theta(x|z)$ by Monte Carlo sampling:

$$\begin{aligned} \log p_\theta(x) &= \log \mathbb{E}_{p_\theta(z)} p_\theta(x|z) \geq \mathbb{E}_{p_\theta(z)} \log p_\theta(x|z) \\ &\approx \frac{1}{N} \sum_{i=1}^N \log p_\theta(x|z_i; z_i \sim p_\theta(z)) \end{aligned} \quad (2.8)$$

The first line applies the Jensen's equality since the logarithm function is concave. We can then run Monte-Carlo estimation over the logarithm space to avoid underflow. The above equation will lead to a lower bound of the real likelihood too. However, performing this kind of sampling for every training step is too costly. We need a huge amount of samples to get a reasonable estimation because most sampled z s from the above equation do not contribute to significant values of $p_\theta(x|z)$.

The evidence lower bound (ELBO) addresses this issue by introducing a proposal $q_\phi(x|z)$ to approximate the true posterior distribution $p_\theta(x|z)$ more accurately:

$$\begin{aligned} \mathbb{E}_{q_\phi(x)} \left[\mathbb{E}_{q_\phi(z|x)} \log \frac{p_\theta(x|z) p_\theta(z)}{q_\phi(z|x)} \right] &= \mathbb{E}_{q_\phi(x)} \left[\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x) || p_\theta(z)) \right] \\ &= \mathbb{E}_{q_\phi(x)} \log p_\theta(x) - \mathbb{E}_{q_\phi(x)} KL(q_\phi(z|x) || p_\theta(z|x)) \\ &\leq \mathbb{E}_{q_\phi(x)} \log p_\theta(x) \end{aligned} \quad (2.9)$$

ELBO is a lower bound of the real likelihood $\mathbb{E}_{q_\phi(x)} \log p_\theta(x)$. It essentially applies importance sampling over the proposal $q_\phi(x|z)$ to get a tighter lower bound compared with Eq. 2.8. Notably, as seen in the second line, by maximising the ELBO, we

are maximizing the marginal likelihood and minimizing $KL(q_\phi(z|x)||p_\theta(z|x))$ at the same time. This is a nice property. As the training proceeds, the KL-regularization will push the proposal $q_\phi(z|x)$ to be closer and closer to the real posterior distribution $p_\theta(z|x)$. It will lead to a tighter bound of the marginal likelihood and benefit the learning of generative parameters θ in turn.

Now we get rid of the intractable term $p_\theta(x)$. Though computing the expectation over $q_\phi(z|x)$ is still intractable, we can estimate it by sampling. Unlike the prior $p_\theta(z)$, sampling from the posterior $q_\phi(z|x)$ is more efficient as it is more peaked and contains useful information of the data x .

Eq. 2.9 can effectively approximate the likelihood. However, the ELBO objective takes the expectation over $q_\phi(z|x)$. θ is easy to be optimized when ϕ is known, but the optimization of ϕ is nontrivial because when ϕ is parametrised by neural networks, it is impossible to backpropagate the error to ϕ through sampled variables. (Hinton *et al.*, 1995) proposed training the ELBO objective through a separate wake-sleep alternation. In the wake phase, the objective is:

$$\max_{\theta} \mathbb{E}_{q(x)} [\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x)||p_\theta(z)))] \quad (2.10)$$

which is the same as ELBO except that the inference parameter ϕ is fixed. z is sampled according the fixed distribution $q_\phi(z|x)$ and θ is optimized to reconstruct the original x . Since the sampling distribution is fixed, gradient descent can be applied.

In the sleep phase, the objective is:

$$\max_{\phi} \mathbb{E}_{q(x)} [-KL(p_\theta(z|x)||q_\phi(z|x))] \quad (2.11)$$

The generating parameter θ is fixed. A “dream” sample is obtained from the generative network by ancestral sampling from $p_\theta(z|x)$ and is used as a target for the maximum likelihood training of the inference network. Again, expectation is taken over a fixed distribution $p_\theta(z|x)$, so gradient descent can be used.

The updating of θ follows the correct gradient as it uses the ELBO objective. However, ϕ is optimized to minimize the reversed KL divergence (As can be seen in the second line of Equation 2.9, the original EBLO aims at minimizing $KL(q_\phi(z|x)||p_\theta(z|x))$ while the sleep phase minimizes $KL(p_\theta(z|x)||q_\phi(z|x))$). Although both can lead to the same global optimum, in practice this might lead to unexpected model behaviours. Besides, the alternative training mechanism can be easily stuck in a local optimum since errors from last stages will be reinforced in later stages.

Another way to train with ELBO is to directly estimate the gradient with respect to ϕ according to:

$$\begin{aligned} \nabla_{\phi} \mathbb{E}_{q_\phi(z|x)} \log p_\theta(z|x) &= \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) \nabla_{\phi} \log(q_\phi(z|x)) \\ &\approx \frac{1}{N} \sum_{i=1}^N \log p_\theta(x|z_i) \nabla_{\phi} \log(q_\phi(z_i|x)); z_i \sim q_\phi(z|x) \end{aligned} \quad (2.12)$$

$$\begin{aligned}
& \left. \nabla_{\phi} KL(q_{\phi}(z|x) || p_{\theta}(z)) \right|_{\phi=\phi_0} \\
&= \left. \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)} \log \frac{q_{\phi_0}(z|x)}{p_{\theta}(z)} + \nabla_{\phi} KL(q_{\phi}(z|x) || q_{\phi_0}(z|x)) \right|_{\phi=\phi_0} \\
&= \left. \nabla_{\phi} \mathbb{E}_{q_{\phi}(z|x)} \log \frac{q_{\phi_0}(z|x)}{p_{\theta}(z)} \right|_{\phi=\phi_0} \tag{2.13} \\
&= \left. \mathbb{E}_{q_{\phi}(z|x)} \log \frac{q_{\phi_0}(z|x)}{p_{\theta}(z)} \nabla_{\phi} \log(q_{\phi}(z|x)) \right|_{\phi=\phi_0} \\
&\approx \left. \frac{1}{N} \sum_{i=1}^N \log \frac{q_{\phi_0}(z_i|x)}{p_{\theta}(z)} \nabla_{\phi} \log(q_{\phi}(z_i|x)); z_i \sim q_{\phi}(z|x) \right|_{\phi=\phi_0}
\end{aligned}$$

The second line in Equation 2.13 comes from the fact that ϕ_0 is a local minimum in $KL(q_{\phi}(z|x) || q_{\phi_0}(z|x))$ thus the gradient with respect to ϕ is 0. Equation 2.13 is needed when KL divergence cannot be analytically computed. Estimating the gradient by this kind of sampling suffers from a very high variance and is not practical way of training the ELBO objective (Paisley *et al.*, 2012).

2.2.2 Reparameterization trick

The main reason of Eq. 2.12's high variance is that it treats $q_{\phi}(z|x)$ as a black box. Random samples from $q_{\phi}(z|x)$ are the only way of getting access to the distribution. To utilize the information from not only random samples but also the parameterization format and distribution family of $q_{\phi}(z|x)$, (Kingma and Welling, 2014; Rezende *et al.*, 2014) proposed a reparameterization trick that allows for efficient estimation of the ELBO and its gradient. When $q_{\phi}(z|x)$ belongs to the location scale family like a Gaussian distribution, we can reparameterize the random variable $\tilde{z} \sim q_{\phi}(z|x)$ by a deterministic transformation $g_{\phi}(x, \epsilon)$ with respect to a noise variable ϵ . For example, if $q_{\phi}(z|x)$ defines a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, we can reparameterize z as $\mu + \sigma\epsilon, \epsilon \sim \mathcal{N}(0, 1)$. Then the ELBO becomes:

$$\begin{aligned}
& \mathbb{E}_{q(x)} [\mathbb{E}_{q_{\phi}(z|x)} \log p_{\theta}(x|z) - KL(q_{\phi}(z|x) || p_{\theta}(z))] \\
&= \mathbb{E}_{q(x)} \mathbb{E}_{p(\epsilon)} [\log p_{\theta}(x, g_{\phi}(x, \epsilon)) - \log q_{\phi}(g_{\phi}(x, \epsilon)|x)] \\
&\approx \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N \log p_{\theta}(x^i | z^{(i,j)}) p_{\theta}(z^{(i,j)}) - \log q_{\phi}(z^{(i,j)} | x^i) \tag{2.14} \\
& z^{(i,j)} = g_{\phi}(\epsilon^{(i,j)}, x^i), \quad \epsilon^{(i,j)} \sim p(\epsilon)
\end{aligned}$$

When $KL(q_\phi(z|x)||p_\theta(z))$ can be analytically computed, the ELBO can be estimated via a simpler form:

$$\begin{aligned} ELBO &= \mathbb{E}_{q_\phi(x)} [\mathbb{E}_{p(\epsilon)} \log p_\theta(x|g_\phi(x, \epsilon)) - KL(q_\phi(z|x)||p_\theta(z))] \\ &\approx \frac{1}{MN} \sum_{i=1}^M [\sum_{j=1}^N \log p_\theta(x^i|z^{(i,j)}) - KL(q_\phi(z|x^{(i)})||p_\theta(z))] \quad (2.15) \\ z^{(i,j)} &= g_\phi(\epsilon^{(i,j)}, x^i), \quad \epsilon^{(i,j)} \sim p(\epsilon) \end{aligned}$$

M is the batch size and N is the sample size. (Kingma and Welling, 2014) shows that when M is large ($M \geq 100$), setting $N = 1$ is enough. Equation 2.15 typically has less variance and the derivative with respect to both θ and ϕ can be efficiently estimated. We can use common gradient descent methods for the optimization. In practice, for the sake of training efficiency, $q_\phi(z|x)$ and $p_\theta(z)$ are normally modelled by Gaussian distribution with diagonal covariance matrices, then we can compute the KL divergence via:

$$\begin{aligned} KL(q_\phi(z|x)||p_\theta(z)) &= \frac{1}{2} [\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + \text{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1)] \\ q_\phi(z|x) &\sim \mathcal{N}(\mu_1, \Sigma_1), p_\theta(z) \sim \mathcal{N}(\mu_2, \Sigma_2) \end{aligned} \quad (2.16)$$

d is the dimension size of z . It can be viewed as a regularised denoising autoencoder. As shown in Equation 2.15, the first term is an autoencoder with random noise added on the representation layer, the second term regularises the encoder distribution $q_\phi(z|x)$ to keep it stay close to the prior distribution $p_\theta(z)$.

2.2.3 Requirement of Applying Variational Approximation

In summary, variational approximation is a powerful framework to efficiently train both the generative and inference parameters of latent-variable models. To apply this framework, we need to make sure:

- The latent variable z is continuous. The generative parameter θ and the inference parameter ϕ are differentiable almost everywhere.
- The inference distribution $q_\phi(z|x)$ belongs to, or can be efficiently approximated by a distribution of, the family that the reparameterization trick can be applied.
- The probability density of the distribution $q_\phi(z|x)$, $p_\theta(x|z)$, $p_\theta(z)$ and, in the best case, $KL(q_\phi(z|x)||p_\theta(z))$, are all explicitly computable.

2.3 IMPROVEMENT OF VARIATIONAL APPROXIMATION

The variational approximation is an effective training framework for latent-variable models and converges to the global optimum when $q_\phi(z|x)$, $p_\theta(x|z)$ and $p_\theta(z)$ have

infinite capacity. In practice, as explained above, $q_\phi(z|x)$ and $p_\theta(z)$ are usually modelled with mean-field Gaussian distributions for training efficiency, which inevitably limits its expressiveness to accurately describe the data distribution.

2.3.1 Limitation of Variational Approximation

From the ELBO objective, we have the following equations:

$$\begin{aligned}
ELBO &= \mathbb{E}_{q(x)} [\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x) || p_\theta(z))] \\
&= \mathbb{E}_{q(x)} \log p_\theta(x) - \mathbb{E}_{q(x)} KL(q_\phi(z|x) || p_\theta(z|x)) \\
&= -H(x) - \mathbb{E}_{q_\phi(z)} KL(q_\phi(x|z) || p_\theta(x|z)) - KL(q_\phi(z) || p_\theta(z)) \\
&= -H(x) - \mathbb{E}_{q(x)} KL(q_\phi(z|x) || p_\theta(z|x)) - KL(q(x) || p_\theta(x)) \tag{2.17}
\end{aligned}$$

$$q_\phi(x|z) = \frac{q_\phi(z|x)q(x)}{q_\phi(z)}, q_\phi(z) = \mathbb{E}_{q_\phi(z|x)} q(x), p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$$

$$p_\theta(x) = \mathbb{E}_{p_\theta(x|z)} p_\theta(z)$$

It is easy to see (from the last line of ELBO) that in the global optimum, we have $q_\phi(z|x) = p_\theta(z|x)$ and $q(x) = p_\theta(x)$ assuming parameters θ and ϕ have infinite capacity. θ can accurately describe the generating distribution and ϕ can accurately infer the real posterior given the generative process defined by θ .

Now we consider the case when $q_\phi(z|x)$ cannot infer the exact posterior distribution $p_\theta(z|x)$, there is a gap of $\mathbb{E}_{q(x)} KL(q_\phi(z|x) || p_\theta(z|x))$ between the real likelihood and the ELBO objective function we use. $p_\theta(x)$ also has problems with matching the real distribution $q(x)$ when this gap exists. From the third line in Equation 2.17 we can see our objective is to have $p_\theta(z) = q_\phi(z)$ and $p_\theta(x|z) = q_\phi(x|z)$ in the global optimum. Even if we make $p_\theta(x|z)$ powerful enough such that the second equality is reachable and we have the optimal $p_\theta^*(x|z) = q_\phi(x|z)$, the first equality still cannot be satisfied when $q_\phi(z|x)$ cannot match $p_\theta(z|x)$, because if we have $q_\phi(z) = p_\theta(z)$, then $q_\phi(z|x) \sim q_\phi(x|z)q_\phi(z) = p_\theta^*(x|z)p_\theta(z)$ and we can have $q_\phi(z|x) = p_\theta^*(z|x)$. When $p_\theta(z) \neq q_\phi(z)$, $p_\theta^*(x) = \mathbb{E}_{p_\theta^*(z)} p_\theta^*(x|z) = \mathbb{E}_{p_\theta^*(z)} q_\phi(x|z) \neq \mathbb{E}_{q_\phi(z)} q_\phi(x|z) = q(x)$. Namely, $p_\theta(x)$ does not converge to $q(x)$ even in the global optimum. Therefore, making $q_\phi(z|x) = p_\theta(z|x)$ is important not only for the inference task but also for the generating task if we hope $p_\theta(x)$ converges to the real $q(x)$ in the global optimum. Though in theory we can make $q_\phi(z|x) = p_\theta(z|x)$ by having either a flexible $q_\phi(z|x)$ or $p_\theta(z)$, normally we prefer a more powerful $q_\phi(z|x)$ to approximate the real posterior rather than adjusting the generating distribution to match a limited posterior distribution $q_\phi(z|x)$.

Apart from $q_\phi(z|x)$ and $p_\theta(z)$, a weak $p_\theta(x|z)$ is also problematic for modelling the real distribution of data. For example, in the task of image generation, a common choice is to set $p_\theta(x|z) \sim \mathcal{N}(g_\theta(z), I/2)$, which is a very weak assumption because we are trying to map the posterior distributions of any data point to a fixed variance

factored Gaussian family. In this case we have:

$$ELBO = \mathbb{E}_{q(x)} [\mathbb{E}_{q_\phi(z|x)} - \|g_\theta(z) - x\|^2 - KL(q_\phi(z|x) || p_\theta(z))] \quad (2.18)$$

which can be seen as a regularised noisy autoencoder with L2 loss. We further have the following proposition (Zhao *et al.*, 2017b):

$$\begin{aligned} &\text{The optimal solution to Equation 2.18 given } \phi \text{ is:} \\ &g_\theta(z) = \mathbb{E}_{q_\phi(z|x)} x \end{aligned} \quad (2.19)$$

This means unless $q_\phi(z|x)$ is a lossless mapping, $g_\theta(z)$ will always tend to average over all possible data x given a specific latent variable z , which is the reason that the VAE often generates realistic but fuzzy natural images. To better model the generating process, a more flexible likelihood distribution $p_\theta(x|z)$ is needed. Nonetheless, $p_\theta(x|z)$ is normally less a focus than $q_\phi(z|x)$ and $p_\theta(z)$ as its distribution is relatively easy. Besides, $p_\theta(x|z)$ is essentially a fully visible model if we condition on a known z value, then we can use any neural decoders to model it. **The most difficult thing is to have $q_\phi(z|x) = p_\theta(z|x)$ and most current efforts are concentrated on devising a more flexible, yet still efficient, $q_\phi(z|x)$ to approximate $p_\theta(z|x)$.** In the following section, we would briefly introduce common techniques of improving the flexibility of $q_\phi(z|x)$.

2.3.2 Auxiliary Variable

The auxiliary variable VAE (Salimans *et al.*, 2015) improves $q_\phi(z|x)$ by imposing a set of auxiliary variables $y = z_0, z_1, \dots, z_{T-1}$. It introduces a stochastic Markov chain $q_\phi(y, z_T|x) = q_\phi(z_0|x) \prod_{i=1}^T q_\phi(z_i|z_{i-1}, x)$ and each distribution is modelled by a mean-field Gaussian distribution. Now we have the marginal distribution $q_\phi(z_T|x) = \int_y q_\phi(y, z_T|x)$, which is a very rich distribution family and can be used as a close fit to the real posterior distribution $p_\theta(z|x)$. However, the probability density of $q_\phi(z_T|x)$ is intractable. To make the training work we need to define another auxiliary inference distribution $r(y|z_T, x) = \prod_{i=1}^T r(z_{i-1}|x, z_i)$. The objective is:

$$\begin{aligned} \mathcal{L}_{aux} &= \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(y, z_T|x)} [\log p_\theta(z_T) p_\theta(x|z_T) r(y|z_T, x) - \log q_\phi(y, z_T|x)] \\ &= \mathbb{E}_{q(x)} \log p_\theta(x) - \mathbb{E}_{q(x)} KL(q_\phi(z_T|x) || p_\theta(z_T|x)) \\ &\quad - \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(z_T|x)} KL(q_\phi(y|z_T, x) || r(y|z_T, x)) \\ &= ELBO - \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(z_T|x)} KL(q_\phi(y|z_T, x) || r(y|z_T, x)) \\ &\leq ELBO \\ &\leq \mathbb{E}_{q(x)} \log p_\theta(x) \end{aligned} \quad (2.20)$$

Though $q_\phi(z_T|x)$ is theoretically rich, the inference needs a chain of sampling from $q_\phi(z_t|x, z_{t-1})$ which does not allow parallel computation. What's more, because of the additional auxiliary variables, the objective function is a less tight lower bound than ELBO, we are losing more accuracy when maximising the real likelihood.

2.3.3 Normalising Flow

Normalising flow (Dinh *et al.*, 2014; Rezende and Mohamed, 2015) also defines a chain of z_0, z_1, \dots, z_T such that $q_\phi(z_T|x)$ has a much broader distribution family than $q_\phi(z_0|x)$. The difference is that it requires the transformation chain to be **invertible**. Let $z_i = f_i(z_{i-1}, x)$ and $z_0 \sim q_\phi(z_0|x)$, then we can derive $q_\phi(z_T|x)$ by using the Jacobian matrices as bellow:

$$\log(q_\phi(z_T|x)) = \log(q_\phi(z_0|x)) - \sum_{i=1}^T \log \det \left| \frac{\partial f_i}{\partial z_{i-1}} \right| \quad (2.21)$$

With the known probability density, we can compute the ELBO objective by:

$$\begin{aligned} ELBO &= \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(z_T|x)} [\log p_\theta(z_T) p_\theta(x|z_T) - \log q_\phi(z_T|x)] \\ &= \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(z_0|x)} [\log p_\theta(z_0) p_\theta(x|z_T) - \log q_\phi(z_0|x) - 2 \sum_{i=1}^T \log \det \left| \frac{\partial f_i}{\partial z_{i-1}} \right|] \\ z_T &= f_T \circ f_{T-1} \cdots f_1(z_0) \end{aligned} \quad (2.22)$$

where \circ means the nested function which recursively applies to the variable. f_i is usually defined such that the determinant of the corresponding Jacobian matrix can be efficiently computed. For example, (Dinh *et al.*, 2014) proposed a transformation that clipping the whole dimension of z into halves:

$$z_t = (z_t^\alpha, z_t^\beta) = f_t(z_{t-1}^\alpha, z_{t-1}^\beta) = (z_{t-1}^\alpha, z_{t-1}^\beta + m_t(x, z_{t-1}^\alpha)) \quad (2.23)$$

m_t can be an arbitrarily complex function. Since f_i has a lower triangular Jacobian with all diagonal terms as 1 and the determinant of a lower triangular matrix equals the product of the diagonal terms, we have $\det \left| \frac{\partial f_i}{\partial z_{i-1}} \right| = 1$ for $i = 1, 2, \dots, T$.

In (Rezende and Mohamed, 2015), the author proposed the planar normalising flow and radial normalising flow in which the Jacobian determinant can also be efficiently computed and brings more flexibility to the approximated posterior distribution.

In contrast with auxiliary variable VAEs, the normalising flow methods limits the power of transformation by restricting it to be invertible, the resulting distribution is in consequence less flexible, but the computable probability density makes the objective a tighter lower bound, so it is basically deriving a less accurate solution for a more accurate objective.

2.3.4 Gaussian Autoregressive Flow

Under the framework of the normalising flow, the Gaussian Autoregressive Flow (Kingma *et al.*, 2016b) defines a very powerful invertible transformation where each dimension

of the new variable is dependent on all the previous dimensions. The transformation function is:

$$\begin{aligned} z_{t,0} &= \mu_{t,0} + \sigma_{t,0}z_{t-1,0} \\ z_{t,i} &= \mu_{t,i}(z_{t,0:i-1}) + \sigma_{t,i}(z_{t,0:i-1})z_{t-1,i}, i = 1, 2, \dots, D \end{aligned} \quad (2.24)$$

$\mu_{t,i}$ and $\sigma_{t,i}$ can be parametrised by neural networks. Since each dimension is only dependent on the previous ones, we have a lower triangular Jacobian. The determinant of the Jacobian matrix can be easily computed by:

$$\det \left| \frac{\partial f_i}{\partial z_{i-1}} \right| = \prod_{i=1}^D \sigma_{t,i}(z_{t,0:i-1}) \quad (2.25)$$

Then it can be optimized with the objective specified in Equation 2.22. Though powerful, the transformation needs a sequential transformation process as specified in Equation 2.24, the cost of the transformation process is $\mathcal{O}(DT)$ and grows linearly as the dimension size. As a result, Gaussian Autoregressive Flow models are not practical applications in reality.

2.3.5 Inverse Autoregressive Flow

With respect to the problem of the Gaussian autoregressive flow, (Kingma *et al.*, 2016b) proposed the inverse autoregressive flow that can be efficiently trained in parallel while still maintaining the flexibility of $q_\phi(z|x)$. The transformation function is simply the reverse of Equation 2.24:

$$\begin{aligned} z_{t,0} &= \frac{z_{t-1,0} - \mu_{t-1,0}}{\sigma_{t-1,0}} \\ z_{t,i} &= \frac{z_{t-1,i} - \mu_{t-1,i}(z_{t-1,0:i-1})}{\sigma_{t-1,i}(z_{t-1,0:i-1})}, i = 1, 2, \dots, D \end{aligned} \quad (2.26)$$

which is still invertible. The advantage is that now z_t is only dependent on z_{t-1} and we do not have to transform dimension by dimension. The transformation can be written in a vectorised version:

$$z_t = \frac{z_{t-1} - \mu_{t-1}(z_{t-1})}{\sigma_{t-1}(z_{t-1})} \quad (2.27)$$

where the division is performed element-wise. Once we have the vector z_{t-1} , $\mu_{t-1}(z_{t-1})$ and $\sigma_{t-1}(z_{t-1})$, Every dimension of z_t can be computed in parallel. The determinant of the Jacobian matrix is simply the reciprocal of Equation 2.25.

$\mu_{t-1}(z_{t-1})$ and $\sigma_{t-1}(z_{t-1})$ can be computed by the masked autoencoder (MADE) (Germain *et al.*, 2015), where the output $y_i = f(x_{1:i-1})$ and can be implemented as an

MLP with a mask matrix:

$$y = \sigma((M \odot Wx + b));$$

$$M = \begin{vmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 \end{vmatrix} \quad (2.28)$$

In (Kingma *et al.*, 2016b), the author used a transformation that is equivalent to Equation 2.27 up to reparameterization: $z_t = \mu_{t-1}(z_{t-1}, h) + \sigma_{t-1}(z_{t-1}, h) \odot z_{t-1}$. Figure 2.6 shows the detailed structure of the inverse autoregressive flow, where h is an additional output from the initial encoder and are inputted at every transformation step.

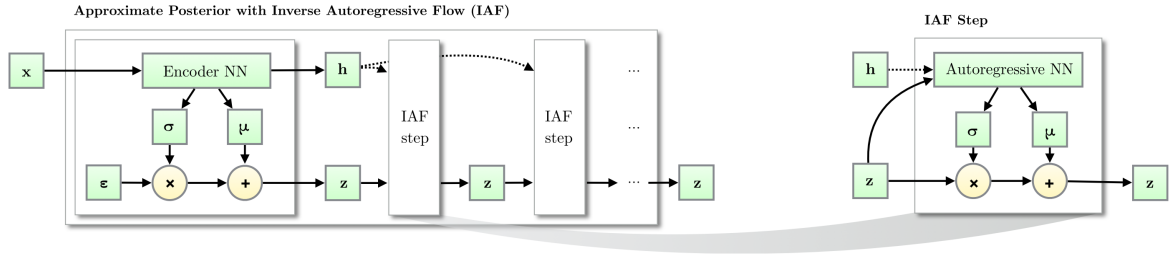


Figure 2.6: Inverse Autoregressive Flow. It consists of an initial sample z drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of z , each with a simple Jacobian determinants.

In addition to the above mentioned methods, we can also use adversarial learning to get a more flexible distribution, which we will see in the next section.

2.4 GENERATIVE ADVERSARIAL NETWORKS

Generative adversarial networks (GANs) (Goodfellow *et al.*, 2014) is another popular technique to approximate an unknown distribution. The main difference of it from previously introduced improving techniques is that it does not directly model the probability density. Instead, they introduce a discriminator to compete with the generator model so that eventually the generator will be driven to recover the real data distribution. In the end, we can get an effective sampler to help us get samples from the unknown distribution.

2.4.1 Overview

The generating process that GANs assume is the same as in variational approximation. First a latent variable is sampled from a prior distribution $z \sim p_\theta(z)$, then a multi-layer-perceptron transformation is applied to directly get a data sample $x = g_\theta(z)$. By this process we implicitly define a data distribution $p_\theta(x)$ whose density distribution is unknown but can be sampled from. The universality of neural networks gives it the potential to fit arbitrary distributions. Our goal is to find the optimal parameter θ such that this implicitly defined $p_\theta(x)$ is as close as possible to the real distribution $q(x)$. There are usually two perspectives of viewing the generative adversarial network: the game theory perspective and the density ratio perspective. The following section will explain these two perspectives.

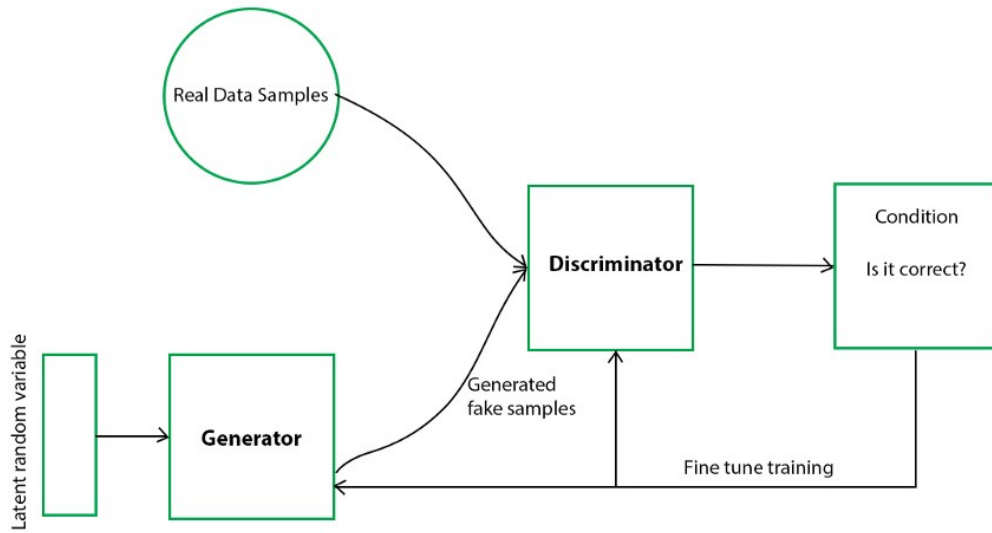


Figure 2.7: Illustration of Generative Adversarial Networks. The discriminator and the generator compete with each other. In the end, the generator will be able to generate real-like samples to fool the discriminator.

2.4.2 From the Game Theory Perspective

The classical definition of generative adversarial networks is from a game theory perspective. According to this definition, the generator, parameterized by θ , generates fake samples with the above-mentioned generating process. We have an extra discriminator D which is trained to distinguish these fake samples from real data samples. The generator and discriminator will compete with each other and form

an adversarial game. The generator can be thought of as analogous to a team of counterfeiters, trying to produce fake currency. On the contrary, the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their skills until the counterfeits are indistinguishable from the genuine articles (Goodfellow *et al.*, 2014). The overall objective is:

$$\begin{aligned} \min_{\theta} \max_D \mathcal{L}(\theta, D) &= \mathbb{E}_{x \sim q(x)} \log D(y = 1|x) + \mathbb{E}_{x \sim p_{\theta}(x)} \log D(y = 0|x) \\ &= \min_{\theta} \max_D \mathcal{L}(\theta, D) = \mathbb{E}_{x \sim q(x)} \log D(y = 1|x) + \mathbb{E}_{z \sim p_{\theta}(z)} \log D(y = 0|g_{\theta}(z)) \end{aligned} \quad (2.29)$$

$D(y = 1|x)$ and $D(y = 0|x)$ output the probability that x is a real or fake respectively. $D(y = 1|x) + D(y = 0|x) = 1$. In the inner loop, D is optimized to correctly identify the input sample. In the outer loop, g_{θ} is driven to produce real-looking samples to fool the trained discriminator. D and g_{θ} are updated alternatively until a Nash equilibrium is achieved.

We can prove for a fixed generator defined by θ_0 , the optimal discriminator is:

$$D_{\theta_0}^*(y = 1|x) = \frac{q(x)}{q(x) + p_{\theta_0}(x)} \quad (2.30)$$

which is intuitive as it outputs a confidence score proportional to the relative probability density. Now suppose the discriminator reaches its optimum for θ_0 , let $Q_{\theta_0}(x) = \frac{q(x) + p_{\theta_0}(x)}{2}$, if we want to update the generator, we have the following results by feeding the optimal discriminator into Equation 2.29:

$$\begin{aligned} &\nabla_{\theta} \mathbb{E}_{x \sim q(x)} \log \frac{q(x)}{2Q_{\theta_0}(x)} + \mathbb{E}_{x \sim p_{\theta}(x)} \log \frac{p_{\theta}(x)}{2Q_{\theta_0}(x)} \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} \mathbb{E}_{x \sim p_{\theta}(x)} \log \frac{p_{\theta}(x)}{Q_{\theta_0}(x)} \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} KL(p_{\theta} || Q_{\theta_0}) \Big|_{\theta=\theta_0} \quad (\text{From Equation 2.13}) \end{aligned} \quad (2.31)$$

Further we can prove:

$$\begin{aligned} &\nabla_{\theta} 2JS(p_{\theta} || q) \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} KL(p_{\theta} || Q_{\theta}) + KL(q || Q_{\theta}) \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} \mathbb{E}_{p_{\theta}} \log \frac{p_{\theta}}{Q_{\theta_0}} + \mathbb{E}_{p_{\theta}} \log \frac{Q_{\theta_0}}{Q_{\theta}} + \mathbb{E}_q \log \frac{q}{Q_{\theta_0}} + \mathbb{E}_q \log \frac{Q_{\theta_0}}{Q_{\theta}} \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} \mathbb{E}_{p_{\theta}} \log \frac{p_{\theta}}{Q_{\theta_0}} - 2KL(Q_{\theta} || Q_{\theta_0}) \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} KL(p_{\theta} || Q_{\theta_0}) \Big|_{\theta=\theta_0} \end{aligned} \quad (2.32)$$

which is the same as the gradient we just derived. Therefore, if we assume $D(x)$ is optimal for any fixed θ_0 , optimizing the generator with respect to Equation 2.29 is equivalent to minimizing the Jensen-Shannon divergence between $p_\theta(x)$ and $q(x)$, where the unique global minimum exists if and only if $p_\theta(x) = q(x)$. In (Goodfellow *et al.*, 2014), it was proved that the Nash equilibrium of this game yields a stationary point where $p_\theta(x) = q(x)$.

However, in practice it is impossible to ensure we can always learn an optimal discriminator. More generally, for any discriminator D_{θ_0} , not necessarily optimal, we have:

$$\begin{aligned}
\min_{\theta} \mathcal{L}(\theta, D_{\theta_0}) &= \min_{\theta} \mathcal{L}(\theta, D_{\theta_0}^*) - \mathbb{E}_{x \sim q(x)} \log \frac{D_{\theta_0}^*(y=1|x)}{D_{\theta_0}(y=1|x)} - \mathbb{E}_{x \sim p_\theta(x)} \log \frac{D_{\theta_0}^*(y=0|x)}{D_{\theta_0}(y=0|x)} \\
&= \min_{\theta} \mathcal{L}(\theta, D_{\theta_0}^*) - 2\mathbb{E}_{x \sim Q_\theta(x)} \mathbb{E}_{y \sim D_{\theta_0}^*(y|x)} \log \frac{D_{\theta_0}^*(y|x)}{D_{\theta_0}(y|x)} \\
&= \min_{\theta} JS(p_\theta || q) - 2\mathbb{E}_{x \sim Q_\theta(x)} KL(D_{\theta_0}^*(y|x) || D_{\theta_0}(y|x)) \\
&\leq \min_{\theta} JS(p_\theta || q)
\end{aligned} \tag{2.33}$$

which means we are basically minimizing a lower bound of the Jensen-Shannon divergence, the bound is tight if and only if the discriminator is optimal. This is not a nice property since normally we would like to minimizing an upper-bound. Minimizing a lower bound instead of an upper bound produces no guarantee about the real divergence. As seen from the last but one line at Equation 2.33, we can optimize by either minimizing the first Jensen-Shannon divergence or maximizing the second KL divergence, we cannot guarantee the generator is moving to the correct direction. It could be that the real divergence is still large even if the objective we are minimizing is already zero. Therefore, the discriminator should be trained close to the optimal to get a tighter bound of the Jensen-Shannon divergence.

2.4.3 From the Density Ratio Perspective

The other view is to consider generative adversarial networks as optimizing by means of estimating the density ratio (Mohamed and Lakshminarayanan, 2017). This reverses the game theory perspective. We start directly from our objective: closing the divergence between $p_\theta(x)$ and $q(x)$. However, as explained above, $p_\theta(x)$ has an unknown density and can only be sampled from. We cannot directly apply gradient descent to optimize it. To solve this problem, a discriminator is imposed to estimate the ratio between $p_\theta(x)$ and $q(x)$. With this ratio, it is possible to estimate the gradient of our objective function. From this perspective, the generator and the discriminator are more like collaborating with each other than competing: the discriminator is trained to freely share the ratio information, with which the generator estimates the gradient value and updates the parameters accordingly.

The density ratio $r_\theta(x)$ can be estimated by:

$$\begin{aligned} r_\theta(x) &= \frac{p_\theta(x)}{q(x)} = \frac{p(x|y=0)}{p(x|y=1)} = \frac{D_\theta^*(y=0|x)p(x)}{p(y=0)} / \frac{D_\theta^*(y=1|x)p(x)}{p(y=1)} \\ &= \frac{D_\theta^*(y=0|x)p(y=1)}{D_\theta^*(y=1|x)p(y=0)} = \frac{D_\theta^*(y=0|x)}{D_\theta^*(y=1|x)} \text{ for } p(y=1) = p(y=0) = 0.5 \end{aligned} \quad (2.34)$$

which can be easily obtained if we have trained the discriminator D to the optimal. Normally we choose a uniform marginal distribution $p(y=1) = p(y=0) = 0.5$. Changing the marginal label distribution is useful if we want more in-depth control: whether we want to precisely fit one of the models (a “high precision, low recall” task such as generation) or explain all of the modes (a “high recall, low precision” task such as retrieval) (Creswell and Bharath, 2016). It has been shown varying the marginal probability value is related to optimizing a generalised Jensen-Shannon divergence (Huszár, 2015). Once we have the ratio, under some circumstances, the gradient information can be estimated. For example, suppose our objective is the KL divergence, we can estimate the gradient with respect to θ by:

$$\begin{aligned} & \left. \nabla_\theta KL(q||p_\theta) \right|_{\theta=\theta_0} \\ &= \left. \nabla_\theta \mathbb{E}_{x \sim q(x)} \log \frac{q(x)}{p_\theta(x)} \right|_{\theta=\theta_0} \\ &= \left. \nabla_\theta \mathbb{E}_{x \sim p_{\theta_0}(x)} \frac{q(x)}{p_{\theta_0}(x)} \log \frac{q(x)}{p_\theta(x)} \right|_{\theta=\theta_0} \quad (2.35) \\ &= \left. \nabla_\theta \mathbb{E}_{x \sim p_{\theta_0}(x)} \frac{q(x)}{p_{\theta_0}(x)} \right|_{\theta=\theta_0} \\ &= \left. \mathbb{E}_{\epsilon \sim p(\epsilon)} \nabla_\theta \frac{1}{r_{\theta_0}(g_\theta(\epsilon))} \right|_{\theta=\theta_0} \end{aligned}$$

which is the same as maximum likelihood since $KL(q||p_\theta) = -H(q) - \mathbb{E}_q p_\theta$ and the first term is a constant. If our objective is the Jensen-Shannon divergence, according to Equation 2.32 we have:

$$\begin{aligned} & \left. \nabla_\theta JS(p_\theta||q) \right|_{\theta=\theta_0} \\ &= \left. \nabla_\theta \frac{1}{2} KL(p_\theta||Q_{\theta_0}) \right|_{\theta=\theta_0} \quad (2.36) \\ &= \frac{1}{2} \mathbb{E}_{p(\epsilon)} \nabla_\theta \log \frac{2r_{\theta_0}(g_\theta(\epsilon))}{1 + r_{\theta_0}(g_\theta(\epsilon))} \end{aligned}$$

which is the same as the original GAN objective, we can recover the original form by replacing r_{θ_0} with the discriminator.

When we train the discriminator and the generator alternatively. Each time the discriminator learn an estimated ratio by distinguishing fake samples from real ones, then the generator uses it to perform gradient descent. In (Nowozin *et al.*, 2016), it was proved the gradient of any distribution that falls into the f -divergence family can be estimated by learning such a density ratio. From this perspective, we can more easily understand why an optimal discriminator D is expected: A discriminator close to the optimum can provide a more accurate estimation of the density ratio, the resulting gradient thus has a larger chance of pushing model parameters in the right direction.

2.4.4 Problem of GAN

Though theoretically appealing, GANs have a lot of problems in practice. The training process can be very unstable. Because of the inaccuracy of the discriminator, we have no idea whether the model is becoming better or not. A stopping point is hard to be well defined. A most popular problem is the gradient vanishing. In the earlier training stage, the generator is very naive. The discriminator can be easily trained to reject all fake samples. As a result, the gradient passed to the generator vanishes, the adversarial game ends up with the winning of the discriminator. For example, suppose we define the discriminator as a sigmoid function, which is a common choice for classification:

$$D(y = 1|x) = \sigma(x) = \frac{1}{1 + e^{-x}} \in (0, 1) \quad (2.37)$$

The gradient with respect to θ in the original objective 2.29 is:

$$\begin{aligned} \nabla_{\theta} \mathcal{L}(\theta, D) &= \nabla_{\theta} \mathbb{E}_{x \sim p_{\theta}(x)} \log D(y = 0|x) \\ &= \nabla_{\theta} \mathbb{E}_{\epsilon \sim p(\epsilon)} \log(1 - \sigma(g_{\theta}(\epsilon))) \\ &= \mathbb{E}_{\epsilon \sim p(\epsilon)} \frac{-\sigma(g_{\theta}(\epsilon))(1 - \sigma(g_{\theta}(\epsilon)))}{1 - \sigma(g_{\theta}(\epsilon))} \nabla_{\theta} g_{\theta}(\epsilon) \\ &= \mathbb{E}_{\epsilon \sim p(\epsilon)} (-\sigma(g_{\theta}(\epsilon)) \nabla_{\theta} g_{\theta}(\epsilon)) \approx 0 \end{aligned} \quad (2.38)$$

The last line comes because if the discriminator can reject almost all fake samples, then $\sigma(g_{\theta}(\epsilon)) = D(y = 1|g_{\theta}(\epsilon)) \approx 0$. The gradient above becomes negligible and the generator does not have any motivation to evolve. We can more easily understand this condition from the density ratio perspective. As shown in Equation 2.36, if the discriminator behaves perfectly, the ratio is a constant and no gradient information will be passed.

As explained in the last section, ideally we would expect an optimal discriminator as it provides a theoretically sound guarantee for the model, but in most cases an optimal discriminator will achieve an accuracy of 100%, then the generator will saturate. The model is faced with an awkward dilemma: A good discriminator provides a good density estimator to train the generator in the right direction. However, the better the discriminator is, the smaller the gradient would be. A weak

discriminator can avoid the gradient vanishing problem but it will end up with an incorrect objective. In practice we must carefully tune up the degree to which the discriminator should be trained, so the training process is notoriously unstable.

In the original GAN paper, the author proposed flipping the sign on the discriminator's cost to obtain a cost for the generator. The cost for the generator becomes:

$$\min_{\theta} -\mathbb{E}_{p_{\theta}(x)} \log D(y = 1|x) \quad (2.39)$$

The only difference is that our objective switches from increasing $D(y = 0|x)$ to reducing $D(y = 1|x)$. Intuitively it is still doing the same thing, but with this new objective we can get a much stronger gradient. When using a sigmoid function, the gradient is:

$$\begin{aligned} & \nabla_{\theta} -\mathbb{E}_{x \sim p_{\theta}(x)} \log D(y = 1|x) \\ &= \nabla_{\theta} \mathbb{E}_{\epsilon \sim p(\epsilon)} \log \sigma(g_{\theta}(\epsilon)) \\ &= \mathbb{E}_{\epsilon \sim p(\epsilon)} \frac{-\sigma(g_{\theta}(\epsilon))(1 - \sigma(g_{\theta}(\epsilon)))}{\sigma(g_{\theta}(\epsilon))} \nabla_{\theta} g_{\theta}(\epsilon) \\ &= \mathbb{E}_{\epsilon \sim p(\epsilon)} (-(1 - \sigma(g_{\theta}(\epsilon))) \nabla_{\theta} g_{\theta}(\epsilon)) \\ &\approx \mathbb{E}_{\epsilon \sim p(\epsilon)} (-\nabla_{\theta} g_{\theta}(\epsilon)) \end{aligned} \quad (2.40)$$

Though avoiding the gradient vanishing problem, the training process still has massively unstable updates with this new objective. The reason is that we are not minimizing the Jensen-Shannon divergence any more. In (Arjovsky and Bottou, 2017), it was proved the new objective is minimizing a reversed KL divergence under an optimal discriminator:

$$\begin{aligned} & \nabla_{\theta} \left(-\mathbb{E}_{x \sim p_{\theta}(x)} \log D_{\theta_0}^*(y = 1|x) \right) \Big|_{\theta=\theta_0} \\ &= \nabla_{\theta} (KL(p_{\theta}||q) - 2JS(p_{\theta}||q)) \Big|_{\theta=\theta_0} \end{aligned} \quad (2.41)$$

Surprisingly, the second Jensen-Shannon divergence term has an opposite sign from the first reversed KL divergence term, which is pushing the model to move far from the real distribution. What's more, the model easily falls into the mode collapse problem: The generator maps various latent variables to the same data points and has a much less diversity than the real data distribution. Many people believe the problem comes from the asymmetry of the reversed KL divergence. Figure 2.8 depicts the asymmetry of KL divergence (Goodfellow, 2016), where $p(x)$ is the real data distribution and $q^*(x)$ is the optimal model distribution to minimize the KL divergence in two directions. The normal KL divergence objective (same as maximum likelihood) tends to cover the full real distribution (higher recall with lower precision), while the reversed KL divergence objective will focus only on a few modes (higher precision with lower recall). The reason of this asymmetric

behaviour lies in the difference of cost: The normal KL divergence severely punishes low probabilities assigned to real data samples while only moderately punishes high probabilities assigned to fake data samples. In contrast, the reversed KL divergence severely punishes high probabilities assigned to fake data samples while only moderately punishes low probabilities assigned to real data samples. In consequence, the reversed KL divergence objective leads to a generator that often synthesises mode-dropping but real-looking data samples.

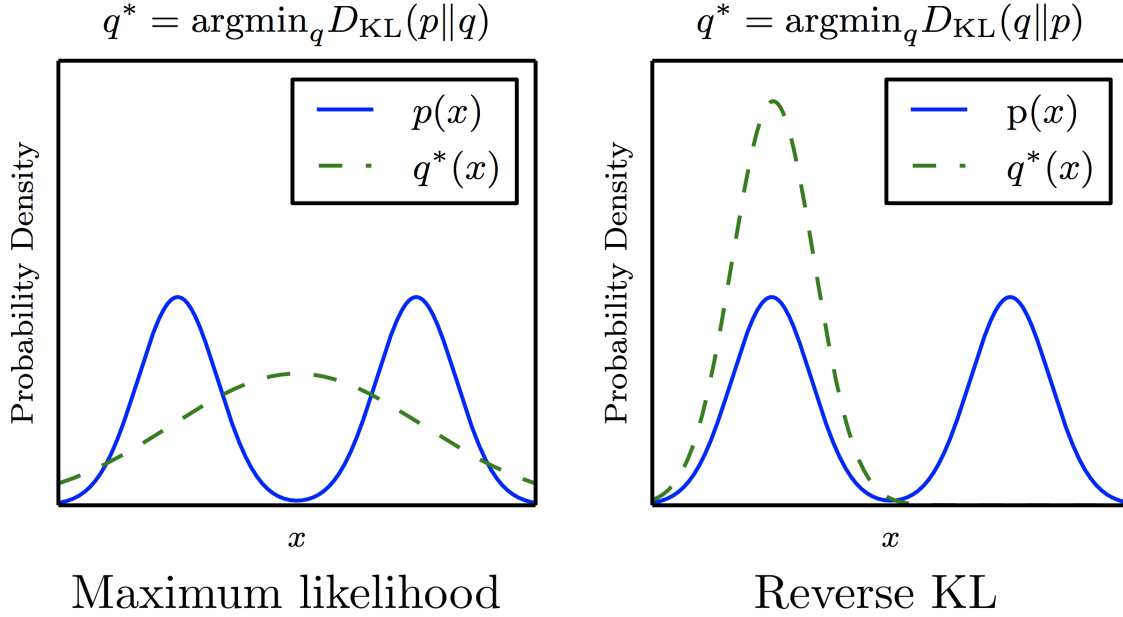


Figure 2.8: Asymmetry of KL divergence

However, people have tried training GANs with a normal KL divergence objective by Equation 2.35. The converged model still generates real-looking samples but only covers a small amount of modes, some researchers suggest the mode-collapse problem comes more from the adversarial training mechanism than from the choice of divergence. For example, in (Metz *et al.*, 2017), the authors argued an optimal generator for any fixed discriminator is a delta function at the x to which the discriminator assigns highest data probability. Therefore, in standard GAN training, each generator update step is a partial collapse towards a delta function.

2.4.5 Requirement for Applying GAN

In summary, generative adversarial networks are a powerful framework to efficiently train a generative distribution for latent-variable models. There are only two requirement for applying this framework:

- The data distribution x must be continuous. The generative parameter θ are differentiable almost everywhere.
- We can sample from, but not necessarily know the exact density function, the likelihood distribution $p_\theta(x|z)$.

It is more powerful than variational approximations because it can in theory approximate arbitrary data distribution and does not need to explicitly specify the likelihood density function. In contrast, the function family of the posterior distribution in the variational approximation severely limits its power. We must have a very powerful posterior distribution in order to be able to match the performance of GAN. However, GANs do not allow inference on the known data points as it only models the generative process. Besides, the training of GANs is much more unstable and difficult than using the variational approximation.

Many tricks have been proposed to stabilise the GAN training procedure and avoid the gradient-vanishing or mode-collapse problem. As mentioned above, the GAN adversarial game can be easily stuck into the discriminator-winning equilibrium and the generator loses its gradient, so one obvious solution is to make harder the task for the discriminator. In the following, we briefly introduce some common practices to stabilize the training of GANs.

2.5 IMPROVEMENT OF GAN

2.5.1 One-Sided Label Smoothing

One-sided label smoothing (Salimans *et al.*, 2016) tries to make the discriminator's task harder by randomly flip the label of real data with probability $\pi < 1$, so that each real data sample has some chance of being fake thus crippling the discriminator's decision. The objective for the discriminator becomes:

$$\max_D \mathbb{E}_{x \sim q(x)} (\pi \log D(y = 1|x) + (1 - \pi) \log D(y = 0|x)) + \mathbb{E}_{x \sim p_\theta(x)} \log D(y = 0|x) \quad (2.42)$$

The optimal discriminator for this task is:

$$D_{\theta_0}^*(y = 1|x) = \frac{(1 - \pi)q(x)}{q(x) + p_{\theta_0}(x)} \quad (2.43)$$

The good thing is that the crippled optimal discriminator scales down the original optimal discriminator to a constant which does not influence its shape, but now the discriminator can never achieve a 100% accuracy because of the random flipping.

2.5.2 Instance Noise

(Sønderby *et al.*, 2016) argued one-sided label smoothing did not fundamentally address the gradient-vanishing problem as it punishes all the discriminators equally.

There is still a large set of discriminators which achieve near-optimal loss, it is just that the near-optimal loss is now larger. Each of these possibly provides very different gradients to the generator. Thus, training the discriminator D might find a different near-optimal solution each time depending on initialisation.

Instead, they proposed adding random noise on the data rather than on the label. Now the objective for the generator becomes:

$$\min_{\theta} d(p_{\sigma} * p_{\theta} || p_{\sigma} * q) \quad (2.44)$$

p_{σ} is a noise function. When d is KL divergence and p_{σ} is the Gaussian random noise, minimizing the above objective can be proved to equal minimizing the Bregman divergence between p_{θ} and q . The advantage of instance noise is that the Bayesian-optimal is unique, the discriminator is less prone to overfitting because it has a wider training distribution, and the log-likelihood-ratio becomes better behaved because there is always an overlap between two distributions and the discriminator can never overwhelmingly win. In (Roth *et al.*, 2017), it was further shown instance noise is equal to gradient regularisation, where we can achieve the same effect without explicitly adding noise. Figure 2.9 compares one-sided label smoothing and instance noise, where the latter has a broader distribution and fewer sub-optimal discriminators.

2.5.3 Minibatch Discrimination

As mentioned, the model tends to generate samples covering only a few modes of the real data distribution in order to more easily fool the discriminator, which is called mode collapse. Minibatch Discrimination (Salimans *et al.*, 2016) tackles this problem by handing in multiple data samples to the discriminator at a time. Every time the discriminator sees a minibatch of fake data or real data samples, it should assign a confidence score to the whole set instead of to single samples. If the generator still generates samples only covering a few modes, its entropy within a minibatch is much lower than that of the real data distribution, the discriminator can easily identify it by means of the entropy difference.

We can show ⁴:

$$\begin{aligned} KL(P^{(N)} || Q^{(N)}) &= N \cdot KL(P || Q) \\ JS(P^{(N)} || Q^{(N)}) &\leq N \cdot JS(P || Q) \end{aligned} \quad (2.45)$$

where $P^{(N)}$ and $Q^{(N)}$ mean the distribution of minibatches with size N . Therefore, when using the KL divergence as the target, the minibatch discriminator does not change our objective, but it does not hold for the Jensen-Shannon divergence.

⁴<http://www.inference.vc/understanding-minibatch-discrimination-in-gans/>

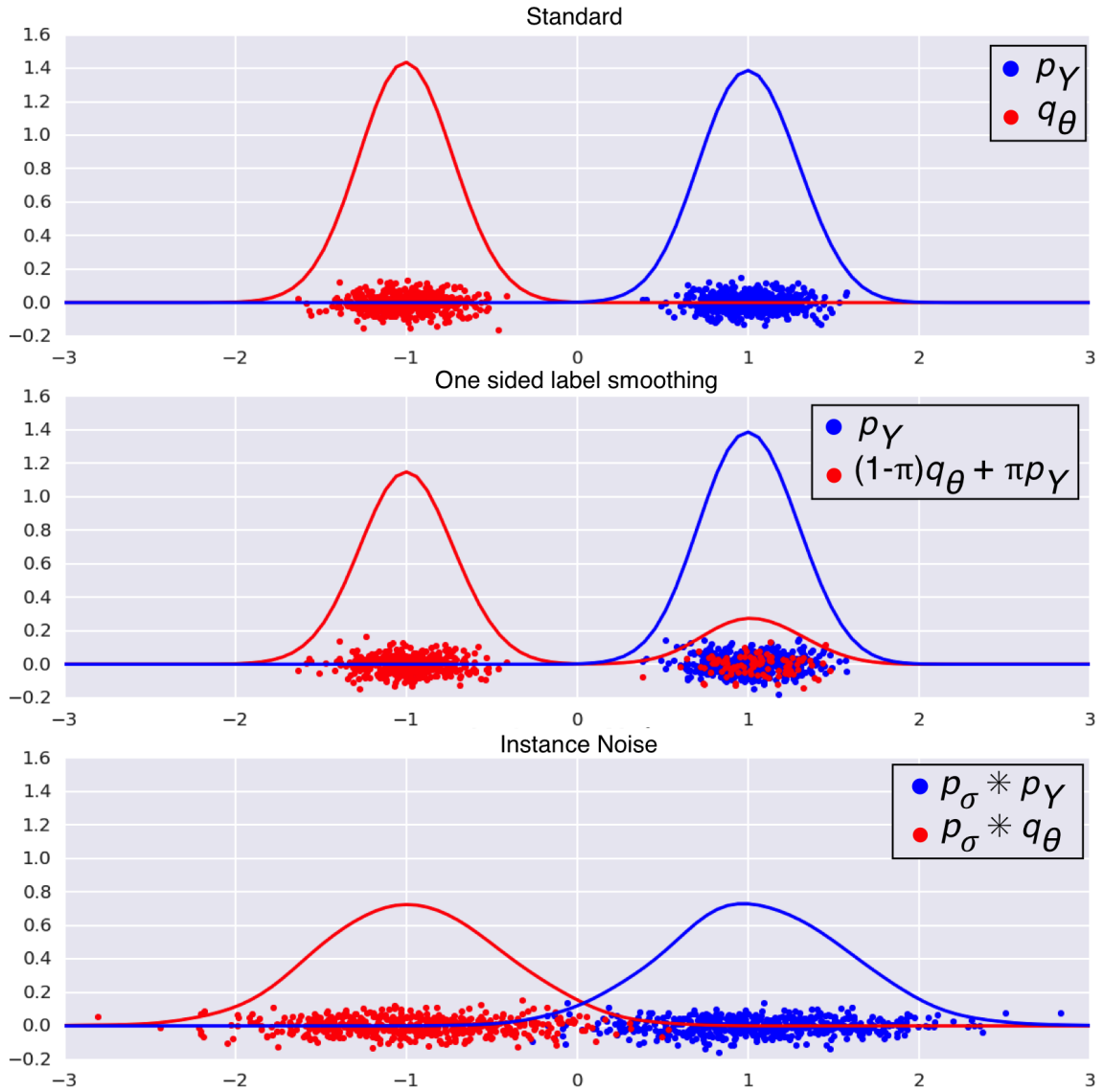


Figure 2.9: Comparison of one-sided label smoothing and instance noise, p_Y denotes real data distribution and q_θ is the generated distribution

2.5.4 Unrolled GAN

Unrolled GAN (Metz *et al.*, 2017) addresses the mode-collapse problem by allowing the generator to “look one step ahead at the future”. At each time step, the discriminator is updated only once based on the current generator. The generator, in contrast, receives gradient information from not only the current discriminator, but also from updated discriminators in the next k steps. Intuitively we can think of it as giving some advantages to the generator because it is allowed to “forsee” what the discriminator will react to it in the next k steps while the discriminator

can only make changes according to the current instant generator. Since the target of the generator is to fool not only the current discriminator but also the future discriminators, the mode-collapse problem will be reduced as it will not blindly move towards high-density areas of any single discriminator. The structure is shown in Figure 2.10. Specifically, when $k \rightarrow 0$, it becomes the original GAN training method, when $k \rightarrow \infty$, we can assume the generator updates based on a nearly-optimal discriminator and we are minimizing the exact Jensen-Shannon divergence. Experiments find generally a larger k value leads to less mode-collapse. This implies an accurate gradient from a close-to-optimal discriminator is clearly expected, if we can find a way to avoid the gradient-vanishing problem.

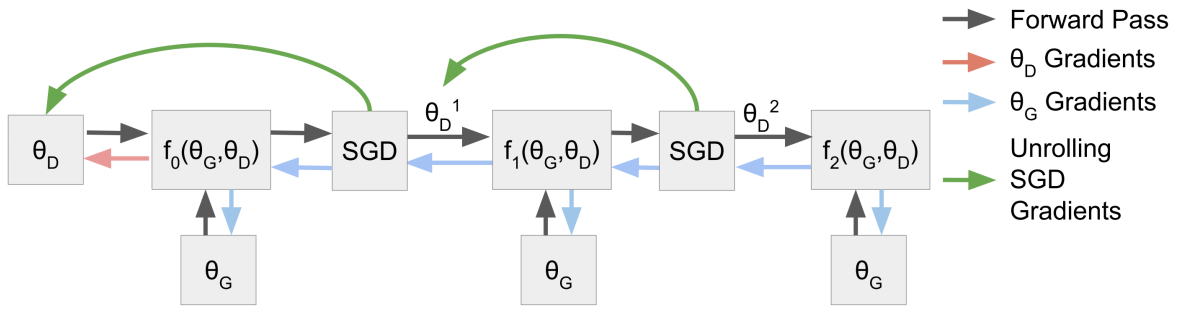


Figure 2.10: Illustration of unrolled gan

2.5.5 Wasserstein GAN

(Arjovsky *et al.*, 2017) fundamentally analysed the training instability of GANs and proposed the Wasserstein GAN framework. It defines the Wasserstein distance, or Earth mover distance as:

$$W(p_\theta || q) = \inf_{\lambda \sim \Pi(p_\theta, q)} \mathbb{E}_{(x, y) \sim \lambda} ||x - y|| \quad (2.46)$$

where $\Pi(p_\theta, q)$ means the set of all possible joint distributions $p(x, y)$ such that $\sum_x p(x, y) = p_\theta(x)$ and $\sum_y p(x, y) = q(y)$. Intuitively we can think of each distribution as a set of earths, where more earths indicating higher probability density. The Wasserstein distance basically measures the least amount of earths to be moved to make them equal. The advantage of Wasserstein distance over KL divergence and Jensen-Shannon divergence is that it can still provide meaningful gradient even when two distributions have completely no overlap. Its gradient is smooth and can continuously drive both distributions to move towards each other. As shown in Figure 2.11, the gradient of the original GAN vanishes when there is no overlap, while Wasserstein GAN has a clean gradients over the whole space.

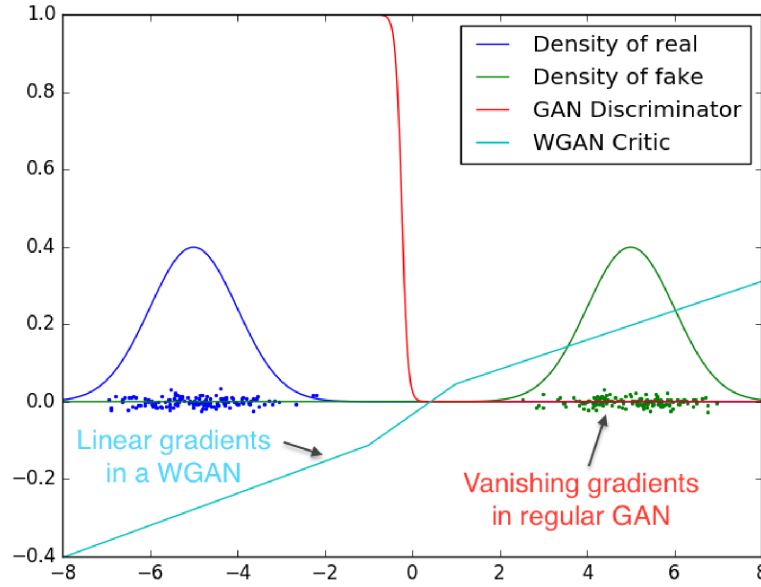


Figure 2.11: Gradient of original GAN and Wasserstein GAN

We can transform Wasserstein distance to a more computable form by applying the Kantorovich-Rubinstein duality (Villani, 2008):

$$W(p_\theta || q) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_\theta} f(x) - \mathbb{E}_{x \sim q} f(x) \quad (2.47)$$

where the supremum is taken over functions f which have a bounded Lipschitz constant K . Specifically, $\|f\|_L \leq K$ means for any x , we have $\|\nabla_x f(x)\| \leq K$.

We can approximate the set of f using a feedforward neural network with its weight restricted in a compact space $[-c, c]$. The resulting function space should be a K -Lipschitz function with some constant K . We do not need to know the exact K value as it only affects the ratio. In practice it can be adjusted by the learning rate. Such a feedforward neural network with weight clipping is referred to as “critic” in (Arjovsky *et al.*, 2017), in replacement of the discriminator in the original GAN. The training objective of Wasserstein GAN is then:

$$\min_{\theta} \max_{f_c} \mathbb{E}_{x \sim q} f_c(x) - \mathbb{E}_{x \sim p_\theta} f_c(x) \quad (2.48)$$

st. weight of $f_c \in [-c, c]$

Since the critic f_c can be fully trained towards the optimum without the problem of gradient vanishing, the mode-collapse problem will also be alleviated. As shown in the unrolled GAN, more training steps of the discriminator can reduce the mode-collapse problem. If every time the critic is optimal thus can provide an accurate gradient, the generator will be stably driven to minimize the Wasserstein distance, which in the end leads to $p_\theta(x) = q(x)$.

(Gulrajani *et al.*, 2017) improves Wasserstein GAN by introducing a more advanced way of restricting the function space of feedforward neural networks. Instead of weight clipping, it adds an additional penalty term to directly prevent the function space from crossing the Lipschitz bound, the improved objective is:

$$\begin{aligned} \min_{\theta} \max_{f_c} \mathbb{E}_{x \sim q} f_c(x) - \mathbb{E}_{\tilde{x} \sim p_{\theta}} f_c(\tilde{x}) - \lambda \mathbb{E}_{\hat{x} \sim p(\hat{x})} (\|\nabla_{\hat{x}} f_c(\hat{x})\| - K)^2 \\ \epsilon \sim U[0, 1], x \sim q, \tilde{x} \sim p_{\theta}, \hat{x} = \epsilon x + (1 - \epsilon) \tilde{x} \end{aligned} \quad (2.49)$$

λ is an adjustable weight, where a large λ value can ensure f_c is a K -Lipschitz function. Experiments show the new objective can make use of the neural network weights more reasonably and increase the expressive power.

Apart from Wasserstein GAN, there are also efforts which use other divergence metrics. For example, energy-based GAN (Zhao *et al.*, 2017a) minimizes the TV-divergence, least-square GAN (Mao *et al.*, 2016) minimizes the Pearson χ^2 divergence. Boundary-equilibrium GAN (Berthelot *et al.*, 2017) minimizes a lower bound of the Wasserstein distance between auto-encoder loss distributions.

2.5.6 VAE+GAN

There have been lots of attempts of unifying the structure of VAE and GAN to combine the strengths from both sides. Compared with GAN, VAE has an extra encoding process which allows inference over data points. The extra inference capability can be used to improve the generator's performance, for example, by forcing generated data samples to be encoded to the same latent space as the real data to prevent mode collapse (Srivastava *et al.*, 2017). In contrast, GANs have the advantage that it does not need to explicitly define the probability density, which allows more flexibility and can in theory approximate arbitrary distribution.

We can apply GANs to include more flexible distributions within the VAE framework. As defined in Section 2.2, in VAE, there are three distributions we need to model: $p_{\theta}(z)$, $q_{\phi}(z|x)$ and $p_{\theta}(x|z)$. GANs can be applied on any of them:

Firstly, in the latent space, we can implicitly define a posterior distribution q_{ϕ} by transforming a Gaussian random noise through $g_{\phi}(\epsilon)$, g_{ϕ} is implemented as a multi-layer perceptron. We can optimize with the original VAE objective:

$$\min_{\theta, \phi} \mathbb{E}_{q(x)} [-\mathbb{E}_{q_{\phi}(z|x)} \log p_{\theta}(x|z) + KL(q_{\phi}(z|x) || p_{\theta}(z))] \quad (2.50)$$

The second term about q_{ϕ} is intractable but can be easily estimated by the density ratio method as in Equation 2.35. We can train a discriminator to distinguish samples from $q_{\phi}(z|x)$ and $p_{\theta}(z)$ to estimate such a ratio. This idea has been extensively explored in (Rosca *et al.*, 2017; Mescheder *et al.*, 2017; Huszár, 2017).

Secondly, in the data space, features learned from the GAN discriminator can be used to define a more reasonable loss function for $p_{\theta}(x|z)$. For example, in (Larsen *et al.*, 2016), we assume $p_{\theta}(D_l(x)|z) = \mathcal{N}(D_l(g_{\theta}(z)), I)$, $D_l(x)$ denote the hidden

representation of the l th layer of the discriminator, the objective is defined as:

$$\min_{\theta, \phi} \mathbb{E}_{q(x)} \left[\frac{1}{2} \mathbb{E}_{q_\phi(z|x)} \|D_l(x) - D_l(g_\theta(z))\|^2 - KL(q_\phi(z|x) || p_\theta(z)) \right] + \mathcal{L}_{GAN} \quad (2.51)$$

The GAN objective is trained simultaneously with the VAE objective, the discriminator is to distinguish real samples from $q(x)$ and fake sample drawn from $p_\theta(x|z)$. In contrast with the Gaussian assumption in Equation 2.18, the target here is the hidden representation extracted by the GAN discriminator. We expect the discriminator can learn high-level invariant representative features that can better follow the Gaussian distribution. Similar ideas are also applied in (Dosovitskiy and Brox, 2016; Lamb *et al.*, 2016a).

Lastly, the GAN idea can also be applied in the joint space of latent variables and data samples. For example, in (Dumoulin *et al.*, 2017; Donahue *et al.*, 2017; Pu *et al.*, 2017), the VAE is optimized through:

$$\min_{\theta, \phi} JS(q(x)q_\phi(z|x) || p_\theta(z)p_\theta(x|z)) \quad (2.52)$$

It differs from the original ELBO objective but leads to the same global optimum. All the three distributions $p_\theta(z)$, $q_\phi(z|x)$ and $p_\theta(x|z)$ can be implicitly defined without known probability density. The discriminator is applied to the samples from the joint distribution $q_\phi(x, z)$ and $p_\theta(x, z)$. Then the obtained density ratio estimate can be used to approximate the gradient as in Equation 2.36.

There are many other interesting ways of integrating VAE and GAN. For example, infoGAN (Chen *et al.*, 2016) applies the variational inference idea to additionally maximise the lower bound of the mutual information between data x and a subset c of the whole latent space (c, z) . The objective is:

$$\min_{\theta, \phi} \mathcal{L}_{GAN} - \lambda \mathbb{E}_{c \sim p(c), x \sim g_\theta(c, z)} \log Q_\phi(c|x) \quad (2.53)$$

As proved in Equation 2.61, the second term is a lower bound of the $I(x, c)$. By adding the additional cost, infoGAN effectively prevents the latent variables from being ignored and interpretable features can be encoded into the latent variable subset c . Adversarial Generator-Encoder Networks (Ulyanov *et al.*, 2017) define a new adversarial game by using only an encoder module and a generator module. The objective is defined as:

$$\max_e \min_g d(e(g(Z)) || e(X)) \quad (2.54)$$

It can be proved there is a Nash equilibrium in this adversarial game if and only if $g^*(Z) = p(Z)$. Unlike the previously-mentioned methods, it does not need an additional discriminator to play the adversarial role. The competition is made between the encoder and the decoder. (Hu *et al.*, 2018) thoroughly analyses the connection between VAEs and GANs and provides a unified view of them. It shows techniques in one model can be easily transferred to the other one and brings performance improvement.

(Kim *et al.*, 2018b) propose the adversarially regularized autoencoder which applied similar ideas on text generation. It builds a normal autoencoder on text plus a GAN component which forces latent codes drawn from a prior distribution to be indistinguishable from those encoded from real text. The final objective contains a reconstruction loss of real text from the autoencoder part and an adversarial loss on the latent space from the GAN part. These two parts are trained simultaneously. The structure is shown in Figure 2.12.

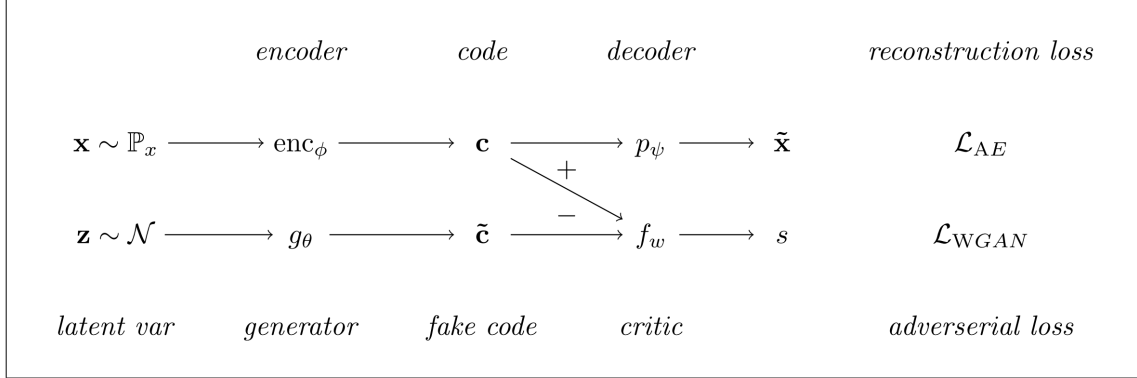


Figure 2.12: Model structure of adversarially regularized autoencoder

Similarly, (Lamb *et al.*, 2016b) proposed Professor forcing that augmented the maximum-likelihood-based (Teacher Forcing) language models by applying GANs on the hidden state of recurrent neural networks. The discriminator is trained to distinguish RNN hidden states with input from either real text data (teacher forcing) or generated text data (free running). Since the input is continuous hidden states, backpropagation can be applied without approximation. The GAN component is built in order for the RNN hidden state to have similar dynamics facing real data or generated data so that the testing performance can be more close to the training performance. The model structure is shown in Figure 2.13

Unlike the last two sections, applying GANs on latent space functions only as a regularizer and does not directly affect the generating process. The GAN component can bring extra benefits like smoother latent representations or more consistent behaviour between training and testing performance, but it still needs the traditional maximum likelihood training criteria to provide error signals.

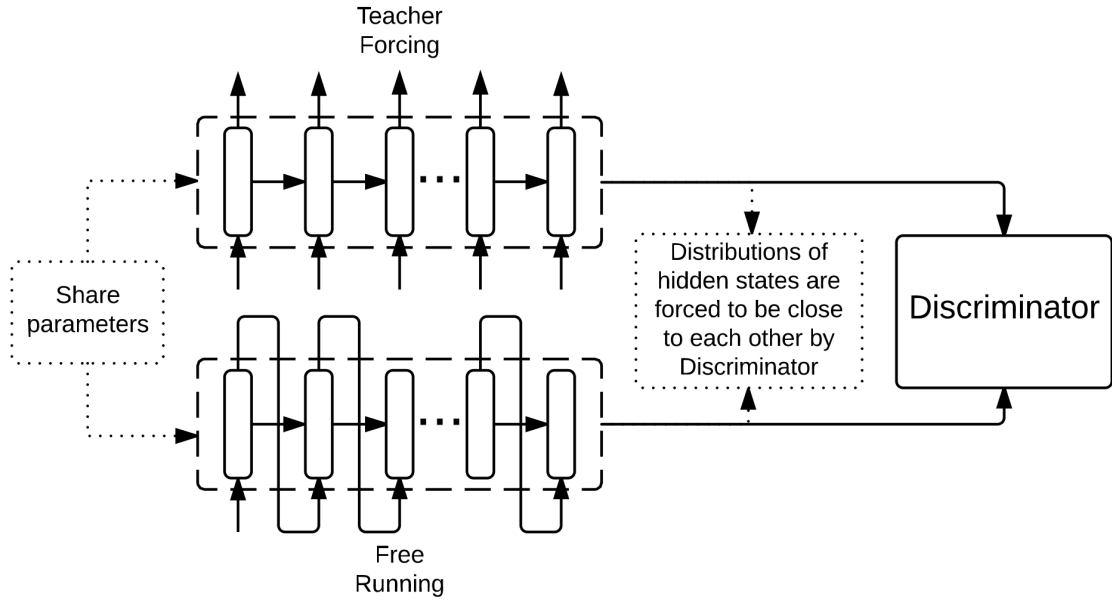


Figure 2.13: Model structure of Professor forcing

2.6 WHEN REPARAMETERIZATION NOT APPLICABLE

The reparameterization trick offers an effective way of getting a low-varianced, unbiased estimation from the proposal distribution $q_\phi(z|x)$. It can be used in both variational approximation and GANs to backpropagate gradients through sampled latent variables. However, it only applies to some specific distributions. When $q_\phi(z|x)$ comes from the distribution family where the reparameterization trick is not directly applicable, there are generally three ways of circumventing this difficulty: EM algorithm, soft relaxation and reinforcement learning. The following sections will introduce these three methods to train latent-variable models when reparameterization tricks are not applicable.

2.6.1 EM algorithm

EM algorithm (Moon, 1996) is a classical way of training latent-variable models. It includes two steps, E-step and M-step:

1. E (Expectation) Step: Compute the posterior distribution $p_\theta(z|x)$ by fixing the generative model $p_\theta(x|z)$
2. M (Maximization) Step: Set $\theta = \operatorname{argmax}_\theta \mathbb{E}_{p_{\theta_0}}(z|x) \log p_\theta(x|z)$. θ_0 is the generative parameter from the last iteration and is fixed in the M-step.

By iteratively performing the E/M step, the log likelihood of the observed data can be gradually improved. In the end, the model is able to find the suitable latent variable z that can explain the generation of x .

The M-step of EM algorithm is similar to minimizing the KL-divergence term in ELBO. The only difference is that in ELBO, the expectation is taken over $q_\phi(z|x)$ whose parameters are simultaneously optimized, while the M-step takes expectation over a fixed posterior distribution $p_{\theta_0}(z|x)$ obtained from the E-step.

We can use this idea to train variational autoencoders. Basically, we can optimize ϕ and θ in an iterative way. At each step, the gradient is not backpropagated to the other. By this means, we can sidestep the non-differentiable issue when reparameterization is not applicable.

Iterative back-translation is a typical application of EM algorithm and has been widely applied in many semi/un-supervised text generation tasks like machine translation and style transfer (Sennrich *et al.*, 2016a; Lample *et al.*, 2018b; Hoang *et al.*, 2018; Subramanian *et al.*, 2019). For many text generation tasks, machine translation for example, it is expensive to obtain parallel text from one language to another. Iterative back translation provides a principled way to effectively utilize unparallel data to facilitate the learning.

Specifically, iterative back-translation assumes the following generation process: Firstly, the condition x is generated from a prior distribution $p(x)$, then the corresponding text y is generated based on $p_\theta(y|x)$. When large amounts of unpaired y is available, we can treat its corresponding x as latent. The generation becomes a latent-variable model. Figure 2.14 depicts this generation process. Iterative back-translation essentially applies EM algorithm to train this latent-variable model.

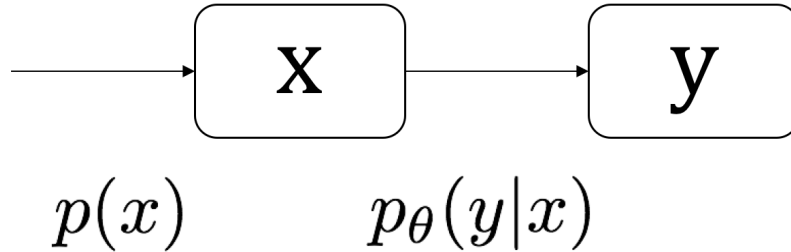


Figure 2.14: Graphical Model of Back Translation

Instead of getting the exact posterior distribution in the E step, iterative back translation resort to a parameterized distribution $q_\phi(x|y)$ to approximate. It utilizes sequence-level knowledge distillation (Kim and Rush, 2016) to minimize $KL(p_\theta(x|y)||q_\phi(x|y))$.

It decompose the training into two steps:

1. E-step: generate pseudo pairs x, y' from unpaired x by $p_\theta(y|x)$. Optimize ϕ to maximize the likelihood of $q_\phi(x|y)$. We can show this equals minimizing $KL(p_\theta(x|y)||q_\phi(x|y))$. By this means, $q_\phi(x|y)$ will be moved towards the real posterior distribution $p_\theta(x|y)$.

2. M-step: generate pseudo pairs x', y from unpaired y by $q_\phi(x|y)$. Optimize θ to maximize the likelihood of $p_\theta(y|x)$. This basically maximizes $\mathbb{E}_{q_\phi(x|y)} \log p_\theta(y|x)$ which treats $q_\phi(y|x)$ as a surrogate of the real posterior distribution $p_\theta(y|x)$.

By repeating the E-step and M-step iteratively, the correspondence between x and y can be established better and better. Many works have shown that even without backpropagating through ϕ in the M-step, the performance will not be affected (Lample *et al.*, 2018b; He *et al.*, 2020). The training process is essentially the same as the wake-sleep algorithm (Hinton *et al.*, 1995). The connection has been analyzed in (Cotterell and Kreutzer, 2018).

2.6.2 Soft Relaxation

Soft relaxation is another popular solution, which approximates discrete variables with continuous substitutes so that the reparameterization trick can be applied. For example, when generating text, the categorical distribution (defined by the softmax layer) is discrete, we cannot use the reparameterization trick to backpropagate the gradient through it. With the soft relaxation technique, we can generate embeddings instead of hard tokens. People have tried to use the same generating mechanism as images with CNNs (Salimans *et al.*, 2016; Shen *et al.*, 2017a). Word tokens are not generated sequentially but independently. The generated sentence is simply a concatenated matrix of every word embedding. By this means, the generated tokens are continuous thus fully differentiable. This method is effective in representation learning, but it cut off the word dependencies thus inappropriate as a generative model.

(Jang *et al.*, 2017; Maddison *et al.*, 2017) proposed Gumbel-softmax to soft-relax samples from the discrete categorical distribution. It has been widely applied in many latent-variable models. Let u be a categorical distribution with probability $\pi_1, \pi_2, \dots, \pi_c$, the samples from this categorical distribution can be approximated by:

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^c \exp((\log(\pi_j) + g_j)/\tau)} \quad (2.55)$$

g_i follows the distribution $\text{Gumbel}(0,1)$ and can be obtained by first sampling $\epsilon_i \sim \text{Uniform}(0,1)$ then generating $g_i = -\log(-\log \epsilon)$. τ is the temperature controlling the sampling accuracy. When $\tau \rightarrow 0$, this is an accurate approximation. It becomes more smooth as τ grows. Figure 2.15 shows the effect of the temperature in Gumbel-softmax. In practice people usually set a high τ value initially then gradually decrease it as the training goes. This approximation technique has been used in many (semi)supervised NLP tasks when generated sentences need to be inputted to another function (Zhou and Neubig, 2017; Yang *et al.*, 2017b; Hu *et al.*, 2017).

With this soft relaxation, we can sequentially generate a sentence word by word. Each time the model outputs a vector with Equation 2.55, which in turn serves as input for the next generating steps. The generated sentence is now a smooth

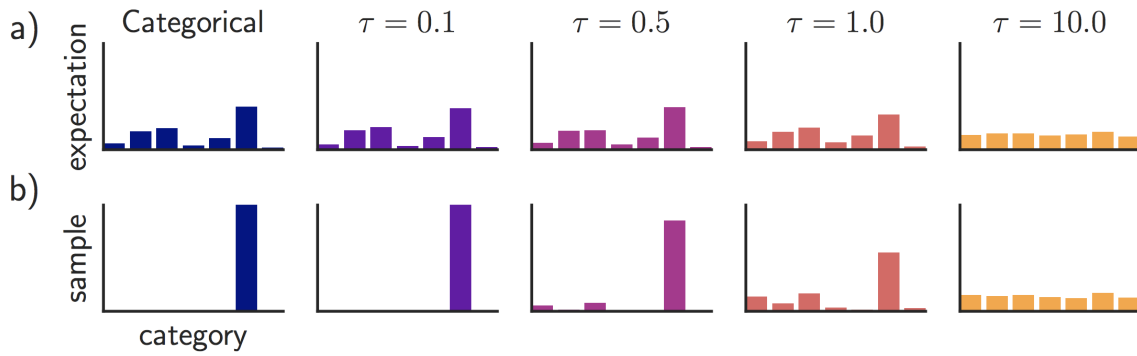


Figure 2.15: Effects of the temperature in Gumbel-softmax

transformation and fully differentiable. We can optimize using gradient descent with the GAN objective as in Equation 2.35 and 2.36. However, unlike in image generation, real natural sentences are discrete with each position being a unique word. Though the approximation technique helps sidestep the non-differentiable problem, turning generates sentences continuous makes the discriminator's task much easier. It can learn to simply reject all samples without a hard code in each position. Though Wasserstein GAN can still produce meaningful gradients in case of a perfect discriminator, the density ratio is less accurate when two distributions are very different. Moreover, languages are different from images in that there is complex inter-dependencies and linguistic correlations. The transition within words is more delicate and needs the fine-grained concern. Relying only on the binary signal from the discriminator has difficulty learning all the grammar rules and long-range word dependencies. More tricks are needed when training GANs on texts like curriculum learning, pre-training with maximum likelihood (Press *et al.*, 2017; Subramanian *et al.*, 2017), or augmented by feature matching via kernelized discrepancy metric (Zhang *et al.*, 2016b, 2017). Table 2.1 shows generates sentences from (Subramanian *et al.*, 2017), where there are still quite a few inconsistent sentences.

To reduce the bias for sequential discrete variables, we can use straight-through Gumbel-softmax. In the forward phase, when generating a sequence, the exact discrete variable is input to the decoder. The soft-relaxation trick of Gumbel-softmax is only used in the backward phase for the gradient computation.

For simpler discrete distributions, like the binary bernoulli distribution, it is even possible to directly use the straight-through estimator. Specifically, let α be the probability of being 1 in the bernoulli distribution. In the forward phase, we set the latent variable as $1(\alpha > 0.5)$ to compute the loss. In the backward phase, we instead use continuous value α to backpropagate the gradient. Likewise, the bias of the gradient computed in this way is related to the peakness of the bernoulli distribution. When α is close to 0 or 1, the bias will be small.

Level	Method	1-billion-word
Word	LSTM	An opposition was growing in China . This is undergoing operation a year . It has his everyone on a blame . Everyone shares that Miller seems converted President as Democrat . Which is actually the best of his children . Who has The eventual policy and weak ?
	CNN	Companies I upheld , respectively patented saga and Ambac. Independence Unit have any will MRI in these Lights It is a wrap for the annually of Morocco The town has Registration matched with unk and the citizens
Character	CNN	To holl is now my Hubby , The gry timers was faller After they work is jith a But in a linter a revent

Table 2.1: Sentences generated by GAN with soft relaxation

2.6.3 Reinforcement Learning

Reinforcement learning, as have been mentioned in the earlier section, requires only samples from a distribution without getting access to the detailed function. It is able to estimate the gradient without reparameterization, which means we only need an estimated value of the probability density ratio but does not need to take the derivative with it. Specifically, if we want to optimize with respect to $KL(q||p_\theta)$, one method is as in Equation 2.35, which requires a continuous output of $g_\theta(\epsilon)$. The other method is through Monte Carlo sampling:

$$\begin{aligned}
\nabla_\theta KL(q||p_\theta) \Big|_{\theta=\theta_0} &= \mathbb{E}_{x \sim q(x)} \nabla_\theta \log p_\theta(x) \Big|_{\theta=\theta_0} \\
&= \mathbb{E}_{x \sim q(x)} \nabla_\theta \log \sum_z p_\theta(x|z) p(z) \Big|_{\theta=\theta_0}
\end{aligned} \tag{2.56}$$

Note that now we only need to take the derivative with respect to $p_\theta(x|z)$ and $p_\theta(z)$, which are computable since z is continuous. Estimating through this method has a high variance since we need to marginalize over z . (Hjelm *et al.*, 2017) proposed

instead minimizing the KL divergence of the joint distribution:

$$\begin{aligned}
& \nabla_{\theta} KL(q(x)p_{\theta_0}(z|x)) || p_{\theta}(x|z)p(z) \Big|_{\theta=\theta_0} \\
&= -\mathbb{E}_{x \sim q(x)} \mathbb{E}_{z \sim p_{\theta_0}(z|x)} \nabla_{\theta} \log p_{\theta}(x|z) \Big|_{\theta=\theta_0} \\
&= -\mathbb{E}_{p \sim p(z)} \mathbb{E}_{x \sim p_{\theta_0}(x|z)} \frac{D^*(y=1|x)}{D^*(y=0|x)} \nabla_{\theta} \log p_{\theta}(x|z) \Big|_{\theta=\theta_0} \\
&\approx -\mathbb{E}_{p \sim p(z)} \frac{1}{Z|_z} \mathbb{E}_{x \sim p_{\theta_0}(x|z)} \frac{\nabla_{\theta} \log p_{\theta}(x|z)}{r_{\theta_0}(x)} \Big|_{\theta=\theta_0}
\end{aligned} \tag{2.57}$$

where $Z|_z = \sum_x p_{\theta}(x|z)/r_{\theta_0}(x)$ to ensure $p_{\theta}(x|z)/r_{\theta_0}(x)$ is a proper density function. In the limit that $r_{\theta_0}(x) = D^*(y=0|x)/D^*(y=1|x)$, $Z|_z = 1$ and it becomes the real posterior $p_{\theta}(z|x)q(x)/p(z)$. Equation 2.57 is essentially one kind of reinforce algorithm (Williams, 1992) with the reward as $1/r_{\theta_0}(x)$ and the baseline as 0. For an optimal discriminator, the reward becomes the density ratio between real and fake samples, the model can get more reward by generating real-like samples.

We can also target the reversed KL divergence by:

$$\begin{aligned}
& \nabla_{\theta} KL(p_{\theta}(x|z)p(z) || q(x)p_{\theta_0}(z|x)) \Big|_{\theta=\theta_0} \\
&= \mathbb{E}_{z \sim p(z)} \nabla_{\theta} \mathbb{E}_{x \sim p_{\theta}(x|z)} \log \frac{p(z)p_{\theta_0}(x|z)}{q(x)p_{\theta_0}(z|x)} \Big|_{\theta=\theta_0} \\
&= -\mathbb{E}_{p \sim p(z)} \mathbb{E}_{x \sim p_{\theta_0}(x|z)} \log \frac{D^*(y=1|x)}{D^*(y=0|x)} \nabla_{\theta} \log p_{\theta}(x|z) \Big|_{\theta=\theta_0} \\
&\approx \mathbb{E}_{p \sim p(z)} \mathbb{E}_{x \sim p_{\theta_0}(x|z)} \nabla_{\theta} \log p_{\theta}(x|z) [-\log r_{\theta_0}(x) - \log Z|_z] \Big|_{\theta=\theta_0}
\end{aligned} \tag{2.58}$$

which is also the same like the policy gradient formula of the reinforce algorithm with reward as $-\log r_{\theta_0}(x)$ and a z -dependent baseline as $\log Z|_z$. When applying it on sequential data like text, this can be viewed as a reinforcement learning problem. The model needs to make decisions on which word to generate based on the previously-generated context. The final goal is to fool the discriminator. Since the reward is only known until a full sentence has been generated, we need to estimate the intermediate value function by Monte Carlo search or another parameterized function. There have been many attempts of applying the reinforce algorithm on generating text with GAN and different techniques to reduce the variance have been proposed (Yu *et al.*, 2017; Li *et al.*, 2016d; Guo *et al.*, 2017; Li *et al.*, 2017a).

The above way of estimating the gradient is basically one simplest example of reinforcement learning. Figure 2.16 illustrates the idea of reinforcement learning. The policy distribution can be updated to maximize the reward with only samples from it. As we have mentioned, reinforcement learning treats the distribution as a

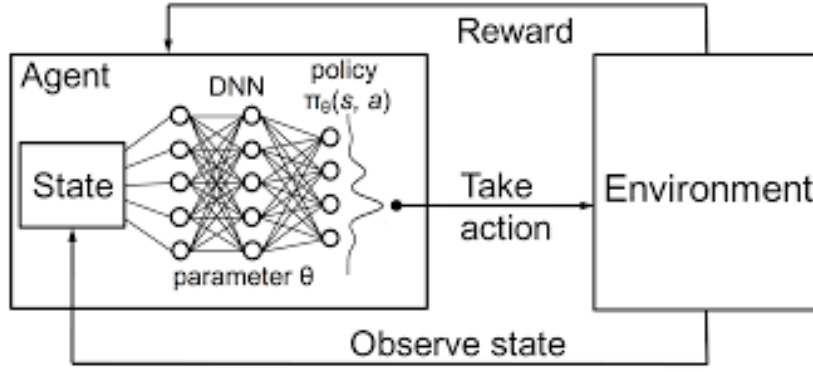


Figure 2.16: Illustration of Reinforcement Learning

blackbox and makes no assumption on it, which makes it more universe compared with the reparametrization. However, it also loses useful information to help reduce the variance. Reinforcement learning is itself a hot topic and we can easily borrow techniques like actor-critic or Q-learning to reduce the variance.

2.7 VAE IN NATURAL LANGUAGE GENERATION

The application of the VAE in natural language generation is quite straightforward. We simply need to replace the likelihood distribution $p_\theta(x|z)$ with a statistical language model that estimates the probability of text sequences. There has been quite a few attempts of applying VAEs in different areas like dialogue generation (Serban *et al.*, 2017d), machine translation (Zhang *et al.*, 2016a), sentence generation (Bowman *et al.*, 2016), document modelling (Miao *et al.*, 2016) and topic models (Srivastava and Sutton, 2017). As explained in Chapter 1, normally for natural language generation tasks we have an input information to condition on, so most above work used the “conditional VAE” framework.

2.7.1 Conditional VAE

The idea of conditional VAEs (CVAEs) (Yan *et al.*, 2016; Sohn *et al.*, 2015) is to condition every distribution in VAEs on an additional context information c . The generating process becomes: z is sampled from a prior distribution $p_\theta(z|c)$, then x is generated from $p_\theta(x|z, c)$. We use $q_\phi(z|x, c)$ to approximate the real posterior distribution $p_\theta(z|x, c)$. The objective is only slightly different:

$$\begin{aligned}
 ELBO &= \mathbb{E}_{q(x|c)} [\mathbb{E}_{q_\phi(z|x, c)} \log p_\theta(x|z, c) - KL(q_\phi(z|x, c) || p_\theta(z|c))] \\
 &= \mathbb{E}_{q(x|c)} [p_\theta(x|c) - KL(q_\phi(z|x, c) || p_\theta(z|x, c))] \\
 &\leq \mathbb{E}_{q(x|c)} p_\theta(x|c) \\
 c &= f(input)
 \end{aligned} \tag{2.59}$$

The conditional VAE can be easily combined with the seq2seq (Sutskever *et al.*, 2014) structure when the input and output are both sequences. For example, in dialogue generation, f can be an LSTM encoder that learns a vectorised representation c of the dialogue history, $p_\theta(x|z, c)$ can be an LSTM decoder that estimates the probability of the ground-truth response. Then we have a CVAE-seq2seq framework that combines advantages from both sides — with the powerful modelling capacity of CVAEs and the end-to-end seq2seq architecture for text modelling.

Though promising results have been achieved, the biggest problem of the VAE application in natural language generation is the “optimizing challenges”.

2.7.2 Optimizing Challenges

The optimizing challenge is that latent variables will be ignored when powerful decoders like LSTMs are used in $p_\theta(x|z)$. In this case, $p_\theta(x|z)$ degenerates to a normal language model with z becoming non-informative to x , which is predictable if we look at the ELBO objective again:

$$ELBO = \mathbb{E}_{q(x)} [\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x) || p_\theta(z))] \quad (2.60)$$

The objective contains reconstruction loss $\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z)$ and KL divergence $KL(q_\phi(z|x) || p_\theta(z))$. When using mean-field Gaussian distributions to parameterize $q_\phi(z|x)$ and $p_\theta(z)$, minimizing the KL divergence is rather easy. We can make $KL(q_\phi(z|x) || p_\theta(z)) = 0$ by turning off the correlation between x and z and set $q_\phi(z|x) = q_\phi(z) = p_\theta(z)$. In contrast, maximising $\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z)$ is more difficult because of the complexity of natural language distributions. Since the LSTM is a universal approximator that can in theory describe arbitrary distribution, it has the potential to accurately model $q(x)$ without reliance to z . Though taking advantage of z will bring more flexibility in the long term, the inherent short sight of gradient descent will inevitably go after the short rewards by ignoring z to avoid paying the extra KL cost. This problematic tendency in learning is compounded by the LSTM decoder’s sensitivity to subtle variation in the hidden states, such as that introduced by the posterior sampling process. In the initial training stage, when $q_\phi(z|x)$ is noisy and does not contain useful information about x , $p_\theta(x|z)$ will also tend to recover x by exploiting the internal patterns of $q(x)$ and neglect the noisy z . Once this has happened, the decoder ignores the encoder and little to no gradient signal passes between the two, yielding an undesirable stable equilibrium with the KL cost term at zero. In practice, we would expect a small construction error with a non-trivial KL cost such that z helps recover the distribution of x . Otherwise there will be no need to impose an additional latent variable.

Similar phenomena is also observed in image generation when expressive $p_\theta(x|z)$ like PixelCNNs (Oord *et al.*, 2016) are used. $p_\theta(x|z)$ tends to model $q(x)$ directly without referring to z . It is more of a problem in language generation since languages are by nature sequential. Using less expressive languages models will fail to capture inter-language dependencies and severely affect the model performance. Several strategies have been proposed to alleviate this problem.

2.7.3 KL cost annealing

KL cost annealing (Bowman *et al.*, 2016) is a simple method that adds an additional weight ϵ to the KL cost term in Equation 2.60. Initially ϵ is set to 0 so that $q_\phi(z|x)$ will try to encode as much information on x as possible to help recover x by $p_\theta(x|z)$. As time goes, ϵ will gradually increase to 1 to recover the original ELBO objective. Specifically, when $\epsilon = 0$, we have:

$$\begin{aligned}
& \max_{\theta, \phi} \mathbb{E}_{q_\phi(x)} \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \epsilon \text{KL}(q_\phi(z|x) || p_\theta(z)) \\
&= \max_{\theta, \phi} \mathbb{E}_{q_\phi(z)} \mathbb{E}_{q_\phi(x|z)} \log q_\phi(x|z) - \log \frac{q_\phi(x|z)}{p_\theta(x|z)} \\
&= \max_{\theta, \phi} \mathbb{E}_{q_\phi(z)} [-H(q_\phi(x|z))] - \mathbb{E}_{q_\phi(z)} \text{KL}(q_\phi(x|z) || p_\theta(x|z)) \quad (2.61) \\
&= \max_{\theta, \phi} I_{q_\phi}(x; z) - H(q_\phi(x)) - \mathbb{E}_{q_\phi(z)} \text{KL}(q_\phi(x|z) || p_\theta(x|z)) \\
&= \max_{\theta, \phi} I_{q_\phi}(x; z) - \mathbb{E}_{q_\phi(z)} \text{KL}(q_\phi(x|z) || p_\theta(x|z))
\end{aligned}$$

At the earlier training stage, when ϵ is small, we are basically trying to approximate

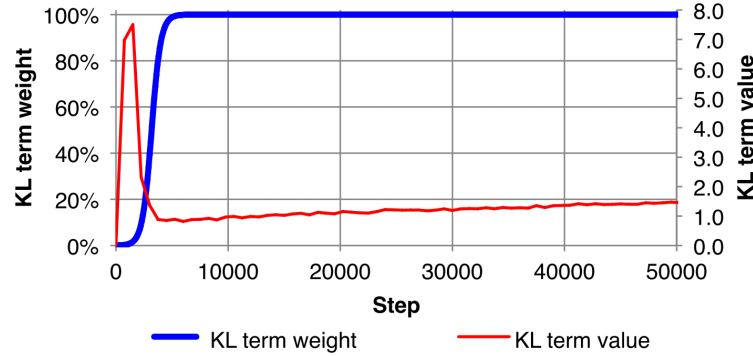


Figure 2.17: Change of KL divergence term

$q_\phi(x|z)$ with $p_\theta(x|z)$ and at the same time maximizing the mutual information between x and z under the distribution defined by $q_\phi(z|x)$, so $q_\phi(z|x)$ will make z informative on x . As ϵ grows to 1, the model will assign more weights to the KL cost term. (Bowman *et al.*, 2016) visualises the change of the KL divergence (Figure 2.17) when applying it on sentence generation. The KL divergence spikes early in training while the model can encode information in z cheaply, then drops substantially once it begins paying the full KL divergence penalty, and finally slowly rises again before converging as the model learns to condense more information into z . In practice, KL cost annealing normally has to be paired with the word drop-out technique (see next section) or early stop to achieve similar results. Otherwise, the KL divergence still eventually vanishes to zero when the weight grows to 1.

2.7.4 Word Drop Out

(Bowman *et al.*, 2016) also proposed word drop-out to encourage the usage of z . In the training phase, some fraction of the history words are randomly replaced with the UNK token indicating unknown words. In this way, we weaken a flexible LSTM decoder by reducing the dependency on history words. The decoder does not have enough history information to recover the exact word thus must turn to the latent variable z . Figure 2.18 depicts the effect of the word keep rate. When more fractions of history words are dropped, the LSTM decoder becomes weaker and has to seek information from z , so the KL divergence grows and z becomes more informative on x .

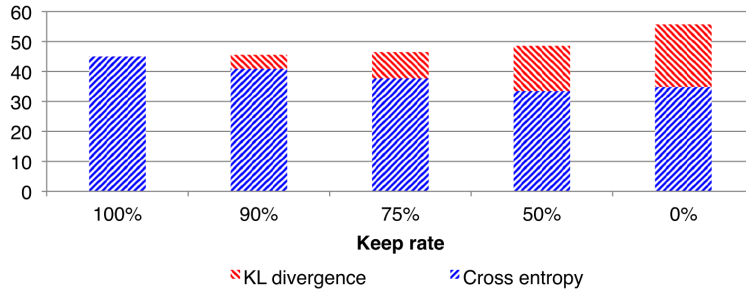


Figure 2.18: Effects of word keep rate. The KL divergence grows as fewer words are kept since the model must rely on more informative z to reconstruct the text.

Interestingly, though seemingly weakening the LSTM decoder, word drop-out has been shown to in fact bring improvement to the performance when a proper keep rate is used. In (Xie *et al.*, 2017), word drop-out is explained as a smoothing technique in neural networks. Specifically, randomly replacing some history words with the UNK token can be seen as an interpolation among different n-gram models. To make a prediction, we use the expected probability over different noise of the context:

$$p_\theta(x_t|x_{<t}) = \mathbb{E}_{\tilde{x}_{<t}} p(x_t|\tilde{x}_{<t}) = \sum_J \pi(|J|) p(x_t|x_J) \quad (2.62)$$

$$\pi(|J|) = \lambda^{|J|} (1 - \lambda)^{t-1-|J|}$$

λ is the word keep rate. $x_{<t}$ is the ground-truth context words, $\tilde{x}_{<t}$ is the noised words after random dropping out. $J \subseteq \{1, 2, \dots, t-1\}$ indicates the indices of un-noised words. $\pi(|J|)$ can be seen as the mixing coefficient of n-gram models $p(x_t|x_J)$. Therefore, word drop-out with a proper rate can not only attenuate the optimizing challenge, but also achieve a lower reconstruction loss by n-gram interpolation.

2.7.5 Additional Loss

We can also prevent the model from ignoring latent variables by imposing additional loss. For example, in (Semeniuta *et al.*, 2017; Zhao *et al.*, 2017c), an additional loss is defined to predict each word without any context information, the model is forced to encode information on z to make such a prediction. The new objective is defined as:

$$\begin{aligned}
 & \max_{\theta, \phi} \mathbb{E}_{q(x)} [\mathbb{E}_{q_\phi(z|x)} [\log p_\theta(x|z) + \log p_\lambda(x|z)] - KL(q_\phi(z|x) || p_\theta(z))] \\
 &= \max_{\theta, \phi} \mathbb{E}_{q(x)} [\log p_\theta(x) - KL(q_\phi(z|x) || \frac{p_\theta(z|x)p_\lambda(x|z)}{Z_x}) + Z_x] \\
 & \text{where } p_\theta(x|z) = \prod_i p_\theta(x_i|x_{<i}, z), p_\lambda(x|z) = \prod_i p_\theta(x_i|z), Z_x = \sum_z p_\theta(z|x)p_\lambda(x|z)
 \end{aligned} \tag{2.63}$$

Since $p_\lambda(x|z)$ is a very weak inputless decoder, the extra reconstruction loss will be high without the help of z , pushing the model to devote more efforts on the reconstruction loss than the KL cost.

The new objective does not change the objective for $p_\theta(z)$ and $p_\theta(x|z)$ because the additional loss has nothing to do with both term. We still have $p_\theta(z) = q_\phi(z)$, $p_\theta(x|z) = q_\phi(x|z)$ and $p_\theta(x) = q(x)$ in the nonparametric limit. However, $q_\phi(z|x)$ is no longer $p_\theta(z|x)$ in the global optimum. As can be seen in Equation 2.63, $q_\phi(z|x)$ will be driven to a mixture of $p_\theta(z|x)$ and $p_\lambda(x|z)$, so the additional loss will lead to a biased inference distribution but an unbiased generating distribution.

In (Zhao *et al.*, 2017c), the authors compared the performance using KL cost annealing (KLA) and additional loss, which they call it bag-of-words loss (BOW) in the paper. The results is shown in Figure 2.19, in which we can see the additional loss effectively avoids the KL-vanishing problem.

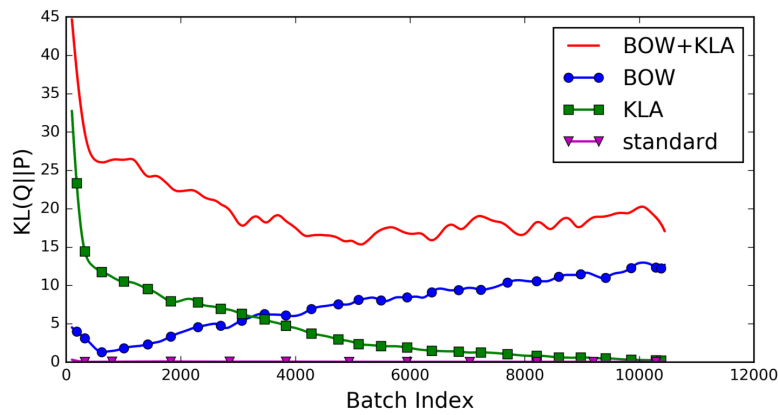


Figure 2.19: Effects of additional loss

2.7.6 Free Bits

The idea of free bits “free bits” (Kingma *et al.*, 2016b) is rather simple. We reserve some space for each dimension in the KL cost term, within which the model is free to encode information into latent variables. Only when the KL divergence exceeds the reserved quota will the model pay attention to it. The objective is defined as:

$$\max_{\theta, \phi} \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \sum_{i=1}^D \max(KL(q_\phi(z_i|x) || p_\theta(z_i)), \epsilon) \quad (2.64)$$

where ϵ is the “free bits” for every dimension.

We can also reserve a quota for the whole KL divergence instead of spreading it over every dimension in average (Yang *et al.*, 2017b), then the objective is:

$$\max_{\theta, \phi} \mathbb{E}_{q(x)} \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \max(KL(q_\phi(z|x) || p_\theta(z)), \epsilon) \quad (2.65)$$

where the KL divergence term is restricted within the range of ϵ . Different from Equation 2.64, here we normally have a few dimensions dominating the KL divergence while most dimensions carry little useful information.

There have been also attempts of reducing the KL-vanishing problem by using CNN decoders (Chen *et al.*, 2017; Yang *et al.*, 2017b; Shen *et al.*, 2017a), where the flexibility of the decoder can be adjusted to trade-off the reconstruction error and KL divergence.

2.8 SUMMARY

In this section, we go over popular optimization techniques for latent-variable models. We also listed the current open challenges, pros and cons of different approaches. In general, the recommended methodology is as stated in the beginning of the section: always start from exact marginalization if possible. When exact marginalization is impossible even with dynamic programming, top-k approximation or pseudo labels, variational approximation can be considered. If variational methods are used, free-bits are currently the most stable regularization to prevent the posterior collapse problem. Advanced techniques, like the flow methods and adversarial training, are not recommended unless the flexibility of the latent variable distribution is confirmed to be a key bottleneck, or the main goal is to explore different alternatives to model the distribution. When the reparameterization trick is applicable, we can simply apply it to estimate the gradient. Otherwise, we can try either EM algorithm, soft relaxation or reinforcement learning. To keep the training more stable, a common practice is to warm up the model with some distant supervision first to initialize it with a decent starting point. In the following contents, we will go through specific applications of latent-variable models and apply techniques from this section to solve them.

THIS chapter presents a novel VAE variant with improved decoding. Variational encoder-decoders (VEDs) have shown promising results in dialogue generation. However, the latent variable distributions are usually approximated by a much simpler model than the powerful RNN structure used for encoding and decoding, yielding the KL-vanishing problem, as mentioned in the last chapter. We propose to separate the training step into two phases: The first phase learns to autoencode discrete texts into continuous embeddings, from which the second phase learns to generalize latent representations by reconstructing the encoded embedding. In this case, latent variables are sampled by transforming Gaussian noise through multi-layer perceptrons and are trained with a separate VED model, which has the potential of realizing a much more flexible distribution. We compare our model with current popular models and the experiment demonstrates substantial improvement in both metric-based and human evaluations (Shen *et al.*, 2018c).

3.1 INTRODUCTION

Recurrent neural networks (RNNs) (Bengio *et al.*, 2003) are widely used in natural language processing tasks. However, given the history context, RNNs estimate the probability of one word at a time and does not work from a holistic sentence representation (Bowman *et al.*, 2016). When applied to dialogue generation, the corresponding result is that it would generate either short, boring responses or long, inconsistent sentences. As the length of generated sentences grows, it would easily deviate from the original intention as such token-level estimation only considers immediate short rewards and neglects global structure consistency. In hence, vanilla RNNs prefer generating generic and safe short responses to avoid the risk of making errors (Vinyals and Le, 2015; Serban *et al.*, 2016; Shen *et al.*, 2017d). One way of improving this deficient generating process is to introduce a sentence-level representation, which can be further conditioned on to ensure the sentence-level consistency.

Deep latent-variable models are a popular way to learn such representations in a generative setting. Latent representations and generators can be jointly trained in an unsupervised way. By learning the probability of synthesizing real data from intermediate latent variables, they are expected to uncover and disentangle causal factors that are most important to explain the data. The exact log-likelihood normally requires integral in high-dimensional space and cannot be analytically expressed. Current approaches solve this intractability problem by imposing a recognition network to approximate the real posterior probability. Variational autoencoders

(VAEs) (Kingma and Welling, 2014; Rezende *et al.*, 2014) bring scalability and stability to the training procedure, which introduces a reparameterization trick to reduce the variance when estimating the backpropagated gradients.

(Serban *et al.*, 2017c) proposed the Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED) structure which applied the conditional VAE (CVAE) (Sohn *et al.*, 2015) with RNN encoder-decoders in dialogue generation, in hope of CVAE’s advantage of learning global representations being a good complement of RNN’s power at modeling local dependencies. However, this simple combination runs into the KL-vanishing problem that the RNN part ends up explaining all the structures without making use of the latent representation. The reason is that RNN is a universal approximator with much more flexibility than the simple gaussian distributed latent variables so that the model lacks enough motivation to utilize them.

Current approaches normally address this problem by weakening the RNN decoder to match the simpler latent variable distribution, which essentially sacrifices the generating capacity for better representation learning and is inappropriate when our main goal is to learn a generative model. In this paper, on the contrary, we take advantage of the universality of RNNs to help realize a more flexible latent variable distribution. By this means, we can not only add motivation for utilizing latent variables, but also strengthen the expressiveness of the generating model. Specifically, we split the whole structure into a CVAE module and an autoencoder (AE) module. The CVAE module learns to generate latent variables while the AE module builds the connection between them and real dialogue utterances. The outputs of the CVAE serve as input latent variables for the AE module, which is potentially much more flexible than restricting the latent variables to follow a fixed distribution. As the RNN encoder-decoders in the AE module are universal approximators, they are adjusted to extract continuous vectors from the dialogue data that can be more easily modelled by the CVAE module. Combined with a scheduled sampling trick, this structure can significantly improve the generating performance. We show this structure can be compared to an adversarial encoder-decoder which substitutes the GAN step with a VAE alternative. Though theoretically less accurate, our framework is preferred to AED as the training process of VAE is much more reliable than GAN in seq2seq tasks and the universality of RNN ensures this inaccuracy can be controlled within an acceptable range.

3.2 VED IN DIALOGUE GENERATION

In this section, we review the VAE and VHRED structure, then analyze where the training difficulty comes from when applied in dialogue generation and how current approaches try to solve this problem.

3.2.1 VAE and VHRED

The variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende *et al.*, 2014) is a popular generative model. Its generating process is as follows: data x is generated by the generative distribution $p_\theta(x|z)$ and z is sampled from the prior distribution $p(z)$. In contrast to calculating the exact log-likelihood, it can be efficiently trained by optimizing a valid lower bound (Jordan *et al.*, 1999). The objective takes the following form:

$$\begin{aligned} -\log p_\theta(x) &\leq -\log p_\theta(x) + \text{KL}(q_\phi(z|x)||p_\theta(z|x)) \\ &= -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \text{KL}(q_\phi(z|x)||p(z)) \end{aligned} \quad (3.1)$$

$p_\theta(z|x)$ is the real posterior distribution of z given the prior distribution $p_\theta(z)$ and the likelihood $p_\theta(x|z)$. The optimizing objective is namely maximizing the likelihood $\log p_\theta(x)$ and at the same time minimizing the mismatch between the approximated posterior $q_\phi(z|x)$, which is parametrized by neural networks, and the real posterior $p_\theta(z|x)$. When the gap $\text{KL}(q_\phi(z|x)||p_\theta(z|x))$ is large, the objective becomes inconsistent and the generating process cannot recover the real data distribution even in the global optimum.

The whole process can be conditioned on an additional context c , which leads to the conditional VAE (Sohn *et al.*, 2015) (CVAE): the output x is generated from the distribution $p_\theta(x|c, z)$, latent variable z is drawn from the prior distribution $p_\theta(z|c)$. The variational lower bound of CVAE is written as follows:

$$-\mathbb{E}_{q_\phi(z|x,c)}[\log p_\theta(x|c, z)] + \text{KL}(q_\phi(z|x, c)||p_\theta(z|c)) \quad (3.2)$$

Specially, to some extent, when both the context c and output x are sequential data, CVAE can also be treated as a seq2seq model (Sutskever *et al.*, 2014).

VHRED (Serban *et al.*, 2017c) is a CVAE with hierarchical RNN encoders, where the first-layer RNN encodes token-level variations and the second-layer RNN captures sentence-level topic shifts. In this case, c in Equation. 3.2 stands for dialogue history, x is the response to be decoded and z is the latent variable reflecting the high-level representation of x . The distribution $q_\phi(z|x, c)$ and $p_\theta(z|c)$ are usually set as simple Gaussian distributions with diagonal covariance matrix.

3.2.2 Optimization Challenges

In VHRED, straightforwardly optimizing with Equation. 3.2 suffers from the KL-vanishing problem because the RNN decoder $p_\theta(x|c, z)$ is a universal function approximator and tends to represent the distribution without referring to the latent variable. At the beginning of the training process, when the approximate posterior $q_\phi(z|x, c)$ carries little useful information, it is natural for the model to blindly set $q_\phi(z|x, c)$ closer to the Gaussian prior $p_\theta(z|c)$ so that the extra cost from the KL divergence can be avoided (Chen *et al.*, 2017).

To better analyze where the optimizing comes from, we can rewrite Equation. 3.2 as the following:

$$-\log \int_z p_\theta(z|c)p_\theta(x|z,c)dz + \text{KL}(q_\phi(z|x,c)||p_\theta(z|x,c)) \quad (3.3)$$

Let's first take a look at the first item, $\log \int_z p_\theta(z|c)p_\theta(x|z,c)dz = \log p_\theta(x|c)$. When the family of $p_\theta(x|z,c)$ is complex enough and includes the real distribution of x , the optimal value of this item is $p(x|c)$ and the reliance on z is not necessary. However, reliance on z provides the model with a chance of taking advantage of z 's distribution and reduces the complexity requirement for the distribution family $p_\theta(x|z,c)$. For example, suppose $p(x|c) = \mathcal{N}(0,1)$ and $p_\theta(z|c) = \mathcal{N}(3,1)$, modeling $p(x|c)$ accurately without reliance on z requires $p_\theta(x|z,c)$ to include the Gaussian distribution, while by means of the linear mapping between x and z $p_\theta(x|z,c)$ can describe the real distribution with only linear complexity. When Gaussian distribution is not covered in the family $p_\theta(x|z,c)$, this model has to exploit the relation between x and z to model the real distribution. Likewise, in dialogue generation, although the RNN decoder $p_\theta(x|c)$ can in theory approximate arbitrary function, perfectly fitting the real dialogue distribution is still difficult due to the optimizing challenge, training corpus size and approximating errors. Therefore, to achieve the global optimum, we believe this first item will always prefer utilizing the latent variables, so long as the decoder $p_\theta(x|z,c)$ is not perfect. The weaker the decoder family is, the more it will be biased to utilizing latent variables. A more flexible prior distribution $p_\theta(z)$ will also increase the chance as it provides more possibilities for utilisation.

The second item is the KL divergence, whose minimum value is 0 if and only if $q_\phi(z|x,c) = p_\theta(z|x,c)$. According to the Bayes theorem, we can express $p_\theta(z|x,c)$ as:

$$p_\theta(z|x,c) = \frac{p_\theta(x|z,c)p_\theta(z|c)}{p_\theta(x|c)} \quad (3.4)$$

By ignoring the latent variable z , $p_\theta(x|z,c)$ and $p_\theta(x|c)$ cancel out, setting $q_\phi(z|x,c) = p_\theta(z|c)$ can easily arrive at the global optimum 0. Otherwise, when $p_\theta(z|c)$ is parametrised as a mean-field Gaussian distribution as in VHRED, the real posterior is impossible to fall into the same distribution family. Firstly, the independence relation cannot be satisfied. To make dimensions of $p_\theta(z|x)$ independent of each other, the likelihood $p_\theta(x|z)$ must exactly disentangle the effect of every dimension, which is unrealistic when $p_\theta(x|z)$ is a categorical distribution modelled by the RNN softmax. Secondly, the real posterior distribution can hardly still follow a Gaussian distribution when the likelihood $p_\theta(x|z)$ is based on discrete sequential data. Normally the training process will adjust $p_\theta(x|z)$ to make the real posterior easier to be modelled by $q_\phi(z|x)$ (Hinton *et al.*, 1995). However, when x represents sentences with variable length, the value of $p_\theta(x|z)$ vanishes greatly when the length grows, which makes the adjusting task much more difficult. This implies the second item will always prefer ignoring the latent variables, so long as the approximated posterior is not powerful enough to perfectly match the real posterior. The weaker

the approximating posterior distribution family is, the more it will be biased to ignoring latent variables.

Above all, the objective function of variational encoder-decoders in dialogue generation is essentially the competition of these two items, who is biased to utilizing or ignoring latent variables respectively. The reason of the KL divergence vanishing in the global optimum is that the second term can gain more from ignoring the latent variables than the first term from utilizing them.

3.2.3 Current Approaches

If we use the ELBO objective, as explained, there are two directions to prevent the KL-vanishing problem: improving the advantage of utilising latent variables in $\log \int_z p_\theta(z|c) p_\theta(x|z, c) dz$ or weakening the advantage of abandoning latent variables in $\text{KL}(q_\phi(z|x, c) || p_\theta(z|x, c))$.

For the former direction, we need to use a smaller distribution family to model the decoder $p_\theta(x|z, c)$. When the decoder is weaker, if ignoring latent variables, it becomes farther from the real distribution at the global optimum thus encouraging latent variables to be exploited. Word drop-out (Bowman *et al.*, 2016) is a common method to weaken the RNN decoder. At each time step, the input word has a certain chance (drop-out rate) of becoming another word, the RNN decoder therefore cannot store a continuous history context. In (Xie *et al.*, 2017), word drop-out is also explained as a special kind of smoothing. Similarly, for CNN decoders, limiting their power can also encode more information to latent variables (Yang *et al.*, 2017b; Chen *et al.*, 2017). Bag-of-word loss proposed by (Zhao *et al.*, 2017c) can also fall into this category. It imposes an extra loss which forces the latent variable to predict the whole sentence without word inputs, which is essentially increasing the weight of the reconstruction loss with the drop-out rate set to 1.

For the latter direction, we need to use a more flexible prior or posterior distribution for latent variables. Once the approximated posterior distribution is powerful enough, the KL divergence can be close to zero without losing the dependence on latent variables. (Serban *et al.*, 2017a) applies a piecewise distribution to replace the Gaussian prior distribution. Though can represent multi-modal conditions, it is still limited as a fixed distribution with pre-defined number of modes. (Salimans *et al.*, 2015) samples latent variables through Markov chains, but it imposed an extra approximation and the objective becomes less accurate. (Rezende and Mohamed, 2015; Kingma *et al.*, 2016a; Chen *et al.*, 2017) use a normalizing flow. Latent variables are first sampled from a simple distribution then passed through several invertible transformations to get better flexibility. Normalizing flow is computationally more costly and has not been applied in text generation yet.

We can also change the original ELBO objective for easier optimization. KL-annealing (Bowman *et al.*, 2016) and free bits (Kingma *et al.*, 2016a) are two popular strategies. In KL-annealing, a small weight is added to the KL divergence term in Equation. 3.2, which starts from zero and gradually increases to 1. This prevents the model from zeroing out the KL divergence at the earlier training stage. Once the

KL divergence vanishes, it is difficult to be recovered for the short sight nature of gradient descent. Free bits reserve some space of KL divergence for every dimension of latent variables. KL divergence is only optimized when exceeding the predefined quota. Similar ideas can be found in (Yang *et al.*, 2017b), which reserved space for the total KL divergence instead of for every dimension.

3.3 IMPROVING VARIATIONAL ENCODER-DECODERS

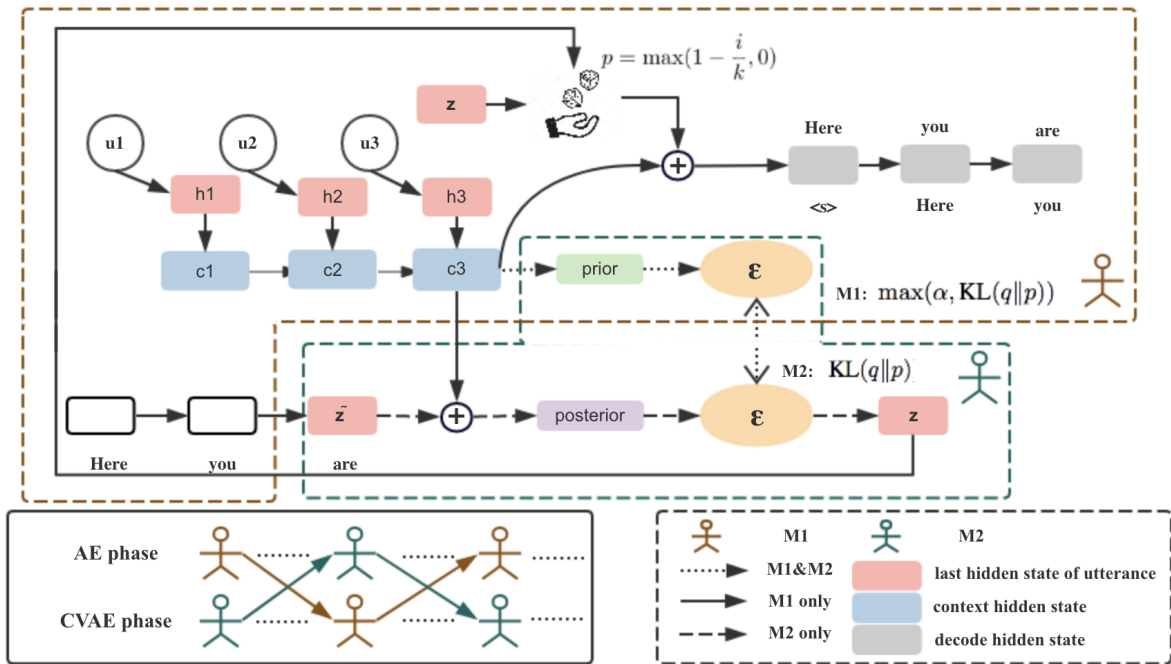


Figure 3.1: Architecture for collaborative variational encoder-decoder. \oplus denotes concatenation of information. $M_1(AE)$ and $M_2(CVAE)$ are represented in brown and green respectively.

As discussed above, two ways for alleviating the optimizing challenge includes weakening the RNN decoders and improving the flexibility of latent variable distributions. The latter class is more fundamental since it also brings more expressiveness to the generating model. Weakening the decoders, though attenuating the KL-vanishing problem, will inevitably hurt the overall performance.

3.3.1 Adversarial Encoder-Decoder

An ideal way of representing the latent variable distribution is to use a universal approximator like neural networks. (Makhzani *et al.*, 2016) proposed adversarial autoencoder (AAE) which samples posterior latent variables by transforming Gaussian

noise through multi-layer-perceptrons. The flexibility of neural networks ensures it can fit arbitrary distribution. However, the probability density is intractable, so adversarial learning (Goodfellow *et al.*, 2014) must be implemented to replace the original KL divergence term.

We can apply this idea to dialogue generation, where AAE is changed to context-dependent adversarial encoder-decoder (AED). The training objective can be represented as:

$$-\mathbb{E}_{q_\phi(z|c,x)} p_\theta(x|c,z) + JS(q_\phi(z|c) || p_\theta(z|c)) \quad (3.5)$$

The training alternates between the autoencoder (AE) phase to optimize $-\mathbb{E}_{q_\phi(z|c,x)} p_\theta(x|c,z)$ and the GAN phase to match the aggregated posterior $q_\phi(z|c)$ and the prior $p_\theta(z|c)$. $q_\phi(z|c, x)$ and $p_\theta(z|c)$ are implicitly defined by passing context-dependent Gaussian random variables ϵ through multi-layer perceptrons. The graphical model is depicted in Figure. 3.2. It can be shown that this objective differs from the original ELBO by adding an extra punishment to the entropy of $q_\phi(x|z, c)$ and using Jensen-Shannon divergence in lieu of KL divergence. In the non-parametric limit, its generating model can recover the exact data distribution.

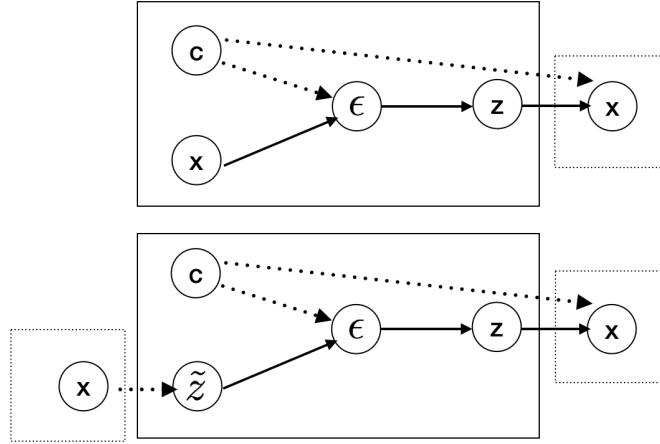


Figure 3.2: Up: adversarial encoder-decoder, down: adversarial encoder-decoder after replacing GAN with VAE, Full line rectangle: GAN and CVAE phase, Dotted line rectangle: AE phase

3.3.2 Replacing GAN with VAE

The idea of AED sounds appealing, but GAN is notoriously difficult to train, especially when both the prior and posterior need to be updated towards each other, the model becomes extremely sensitive to hyper-parameters and the training is very unstable. In consequence, we try replacing the GAN phase with a CVAE alternative. An RNN encoder is first applied to extract the corresponding latent variable target \tilde{z}

for each dialogue turn x , based on which a CVAE is trained to reconstruct it through context-dependent Gaussian noise. The connection to AED can be seen in Figure 3.2. Specifically, we just replace the $JS(q_\phi(z|c)||p_\theta(z|c))$ in Equation 3.5 with the following CVAE objective:

$$-\mathbb{E}_{q_\phi(\epsilon|c,\tilde{z})} p_\theta(z|c,\epsilon) + KL(q_\phi(\epsilon|c,\tilde{z})||p_\theta(\epsilon|c)) \quad (3.6)$$

$q_\phi(\epsilon|c,\tilde{z})$ is an approximated posterior. It can be easily proved when $q_\phi(\epsilon|c,\tilde{z})$ is powerful enough to cover the real posterior $p_\theta(\epsilon|c,\tilde{z})$, objective 3.6 has the same global optimum as in $JS(q_\phi(z|c)||p_\theta(z|c))$. We can therefore instead alternate between the AE phase and the CVAE phase to achieve the same effect as in AED.

3.3.3 Constraining RNN Encoder

The accuracy of the CVAE objective relies on the matching degree of $q_\phi(\epsilon|c,\tilde{z})$ and $p_\theta(\epsilon|c,\tilde{z})$. Therefore, in the AE phase, apart from encoding representative information to reduce the normal AE reconstruction loss, the RNN encoder should also encode utterances in a manner where the real posterior $p_\theta(\epsilon|c,\tilde{z})$ can be more easily modelled by the distribution defined by $q_\phi(\epsilon|c,\tilde{z})$ in the CVAE phase. To do this, we add a KL divergence constraint to the RNN encoder in the AE phase. The RNN encoder has to keep $KL(q_\phi(\epsilon|c,\tilde{z})||p_\theta(\epsilon|c))$ within a specific range. It is also possible to constrain the value of the whole CVAE objective of Equation. 3.6, but we find constraining only the KL divergence is enough when the alternating step is not too large. Note that in the encoder phase, the model can only adjust the RNN encoder-decoders to control the KL divergence, the generating parameters for latent variables are fixed.

3.3.4 Scheduled Sampling Trick

In the AE phase, we also find it useful to initially use the ground-truth encoding \tilde{z} then gradually change to noisier CVAE output z . We apply the scheduled sampling strategy proposed in (Bengio *et al.*, 2015). Before decoding, a coin is flipped to decide whether to feed the real hidden vector \tilde{z} or the noisy z . In the beginning, to make it easy, we mostly pick the real \tilde{z} . As the training proceeds, we gradually improve the difficulty by increasing the chance of selecting noisy z until finally all inputs are replaced with the z . We decide the chance of selecting the real \tilde{z} with a linear decay function as:

$$p = \max(1 - \frac{i}{k}, 0) \quad (3.7)$$

i is the step number and k is a constant controlling the decaying speed. Other decaying functions are also applicable like exponential decay or inverse sigmoid decay.

3.3.5 Training Process

Our model contains a CVAE phase and an AE phase. These two phases are trained iteratively until an equilibrium is achieved.

In the CVAE phase, A sample \tilde{z} is obtained from the AE by transforming dialogue texts into a continuous embedding and is used as a target for the maximum likelihood training of the CVAE. We assume the generative model $p_\theta(z|\epsilon, c) = \mathcal{N}(\tilde{z}, I)$, the loss function is:

$$\min_{\phi} \text{KL}(q_\phi(\epsilon|\tilde{z}, c) || p_\phi(\epsilon|c)) + \frac{1}{2} \mathbb{E}_{q_\phi(\epsilon|\tilde{z}, c)} || g_\phi(\epsilon) - \tilde{z} ||_2^2; \quad (3.8)$$

$$\tilde{z} = f_\theta(x)$$

f_θ is the RNN encoder and is fixed as part of the AE module during training.

In the AE phase, An observation x is sampled from the training data and fed into the transform function to get a continuous vector representation $\tilde{z} = f_\theta(x)$. The corresponding latent variable z is sampled from the posterior distribution $q_\phi(z|\tilde{z}, c)$ provided by the CVAE part. The sampled latent variable z , together with x , forms a target for training the AE. The objective function is:

$$\min_{\theta} \max(\alpha, \text{KL}(q_\phi(\epsilon|\tilde{z}, c) || p_\phi(\epsilon|c)))$$

$$- \mathbb{E}_{q_\phi(z|\tilde{z}, c)} [\log(p_\theta(x|z, c))]; \quad (3.9)$$

$$\tilde{z} = f_\theta(x), z = (1 - p)g_\phi(\epsilon) + p\tilde{z}$$

The first item is used to control KL divergence in a reasonable range such that the transformed z can be more easily modelled by the CVAE phase. α can be used to adjust the leverage between the reconstruction loss and KL divergence, where a lower α value will lead to a lower KL divergence in the end. p is the keeping rate defined in Equation. 3.7. The detailed architecture is depicted in Figure 3.1. We refer to this framework as collaborative VED where the AE and CVAE phase collaborate with each other to achieve a better generating performance.

3.3.6 Model Summary

In summary, we replace the GAN phase of AED with a CVAE alternative. The output of the CVAE part are latent variables, which can represent a much broader distribution family than mean-field Gaussian. As CVAE is in theory less accurate than GAN because it needs to approximate the real posterior, we leverage the more powerful RNN encoder-decoders. In the AE phase, they should autoencode utterenaces to make the real posterior easily representable by the CVAE part.

3.4 EXPERIMENTS

We conduct our experiments on two dialogue datasets: Dailydialog (Li *et al.*, 2017c) and Switchboard (Godfrey and Holliman). Dailydialog contains 13118 daily con-

versations under ten different topics. This dataset is crawled from various websites for English learner to practice English in daily life. Statics show that the speaker turns are roughly 8, and the average tokens per utterance is about 15, which are appropriate for training dialog models. Switchboard has 2400 two-sided telephone conversations under 70 specified topics with manually transcribed speech and alignment. Compared with Dailymail, the turn of every dialogue is much longer and the subject is more disperse. These two datasets are randomly separated into training/validation/test sets with the ratio of 10:1:1.

3.4.1 Models and Training Procedures

For comparison, we also implemented the hred model (seq2seq model with hierarchical RNN encoders), which is the basis of VHRED. Latent variable models are trained by standard KL-annealing with different weights (Bowman *et al.*, 2016; Higgins *et al.*, 2017), with additional BOW loss (Zhao *et al.*, 2017c; Semeniuta *et al.*, 2017), word drop-out (Bowman *et al.*, 2016), free bits (Kingma *et al.*, 2016a) and our collaborative VED (CO) with the scheduled sampling trick (SS). For our framework, we use the encoder RNN as the transformation function $f_{\theta}(x)$. We tuned the parameters on the validation set and measure the performance on the test set. In all experiments, the letters are all transformed to the lower-case, the vocabulary size was set as 20,000 and all the OOV words were mapped to a special token `<unk>`. We set word embeddings to size of 300 and initialized them with Word2Vec embeddings trained on the Google News Corpus. The first, second-layer encoder and decoder RNN in the following experiments are single-layer GRU with 512, 1024 and 512 hidden neurons. The dimension of latent variables is set to 512. The batch size is 128 and we fix the learning rate as 0.0002 for all models. Our framework is trained epochwise by alternatively training the CVAE and DAE part. The probability estimators for VAE are 2-layer feedforward neural networks. At test time, we output the most likely responses using beam search with beam size set to 5 (Graves, 2012) and `<unk>` tokens were prevented from being generated. We implemented all the models with the open-sourced Python library Tensorflow (Abadi *et al.*, 2016) and optimized using the Adam optimizer (Kingma and Ba, 2015). Dialogs are cut into set of slices with each slice containing 80 words then fed into the GPU memory.

3.4.2 Metric-based Evaluation

We compare our model with the basic HRED and several current approaches including KL-annealing (KLA), word drop-out (DO), free-bits (FB) and bag-of-words loss (BOW). The details are summarized in Table 3.1 and 3.2. For KLA, we initialize the weight with 0 and gradually increase to 1 in the first 12000 or 25000 training steps for Dailymail and Switchboard respectively. The word drop-out rate is fixed to 25%. Words are dropped out only in the training step. We set the reserved space for every dimension as 0.01 in free bits (FB) and also try reserving 5 bits for the

whole dimension space (FB-all). We use an α value 5 for our collaborative model (CO) and set the scheduled sampling (SS) weight $k = 2500$ or 5000 for Dailydialog or Switchboard. We also experiment with jointly training the AE and CVAE part in our model and report the results. Table 3.1 measured the perplexity (PPL), KL

Table 3.1: Metric Results, left: Dailydialog, right: switchboard

Model	PPL		KL		NLL	
HRED	43.4	48.3	0.00	0.00	229.1	355.6
KLA	31.8	44.5	4.90	4.36	225.0	331.6
KLA+DO	29.8	40.1	3.80	4.48	223.9	317.0
KLA+BOW	26.8	30.9	12.8	8.92	247.3	321.1
FB	41.7	32.1	3.34	3.90	239.0	322.7
FB-all	29.4	21.7	5.01	4.97	226.1	308.2
CO	26.1	36.5	4.90	4.94	223.6	289.7
CO+DO	25.1	34.4	5.01	4.93	218.7	273.4
CO+SS	23.8	31.8	4.92	4.93	213.2	273.4
CO+SS(joint)	28.5	39.6	5.16	5.02	224.3	301.3

divergence (KL) and negative log-likelihood (NLL). NLL is averaged over all the 80-word slices within every batch. For latent-variable models, NLL is computed as the ELBO, which is the lower bound of the real NLL.

As can be seen, our model CO+SS achieves the lowest NLL over both datasets. The Schedule Sampling (SS) strategy significantly helps bring down the NLL. Word drop-out (DO), though weakening the RNN decoder, improved the performance when combined with both KLA and CO, which verified the assumption that DO can function as a smoothing technique in neural network language models (Xie *et al.*, 2017). KLA itself needs early stop, otherwise the KL divergence will vanish once the weight increases to 1. BOW avoids the KL-vanishing problem, but the overall performance will significantly decrease because adding an additional loss in theory leads to a biased result for latent variables. BOW information is encoded into the latent variable, but it prevents the decoder from stably learning the word order pattern in the training step thus sacrifices the NLL performance. FB-all performs much better than FB, which suggests most important information is concentrated on a few dimensions. Equally reserving space for every dimension is not suitable. Finally, we also testified the necessity of iteratively training our model. Jointly training the model brings recession on both the perplexity and KL divergence on the two datasets.

Figure. 3.3 visualizes the latent variables drawn from VHRED and our framework. We randomly pick a dialogue context "I'd like to invite you to dinner tonight , do you have time ?" and apply the information retrieval based method to gather 10 responses with similar context from the corpus. All the 10 responses are verified by humans as appropriate ones, which span over different possibilities like "Thank you for your invitation. ", "Don't be silly . Let's go Dutch ." and "Are you asking me for a date ? ". For each response, 10 samples are drawn from the posterior latent variable

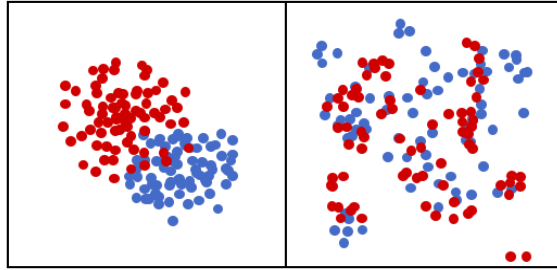


Figure 3.3: T-SNE visualization of sampled latent variables. left: VHRED, right: CO+SS. Red dots correspond to samples from prior distribution, while the blue dots correspond to samples from posterior distribution. Viewable in color mode only.

distribution, which forms 100 posterior latent variable samples (blue dots) in total. Likewise, 100 samples are drawn from the prior latent variable distribution (dots) given only the dialogue context. The visualization clearly indicates the superiority of our framework in modelling more flexible prior and posterior latent variable distributions. In the VHRED model, both the prior and posterior distributions are limited uni-modal Gaussians with only a little overlap. In our framework, the distributions are more diverse and samples from the prior and posterior distribution share more overlap with each other.

Table 3.2: Embedding Results, left: dailydialog, right: switchboard

Model	Average		Greedy		Extrema	
HRED	0.463	0.334	0.445	0.399	0.356	0.280
KLA	0.442	0.317	0.436	0.327	0.327	0.267
KLA+DO	0.458	0.325	0.461	0.341	0.378	0.283
KLA+BOW	0.475	0.340	0.459	0.352	0.386	0.302
FB	0.423	0.336	0.414	0.348	0.349	0.318
FB-all	0.429	0.341	0.439	0.352	0.357	0.325
CO	0.465	0.377	0.465	0.381	0.394	0.331
CO+DO	0.489	0.385	0.471	0.379	0.397	0.337
CO+SS	0.539	0.392	0.477	0.394	0.443	0.340
CO+SS(joint)	0.420	0.347	0.452	0.360	0.351	0.308

Table 3.2 reports the results of the embedding-based topic similarity: Embedding Average (Average), Embedding Extrema (Extrema) and Embedding Greedy (Greedy) (Liu *et al.*, 2016). Unlike the NLL, who measures the token-level match, these embedding-based metrics map responses to a vector space and compute the cosine similarity with golden answers, which can to a large extent measure the sentence-level semantic similarity.

We can see our model still achieved the highest topic similarity according to all the three metrics. This suggests our model can bring improvement for both token-

Table 3.3: Examples of context-response pairs for the neural network models. `__eou__` denotes end-of-utterance and indicates the start of a new turn.

Context	Response
The taxi drivers are on strike again . <code>__eou__</code> What for ? <code>__eou__</code> They want the government to reduce the price of the gasoline . <code>__eou__</code>	KLA: all right . i'll take it . KLA+BOW: well , how much is the price of the price price ? CO+SS: that's good . i want to buy a new car . Gold: It is really a hot potato .
I'd like you to do me a favor . <code>__eou__</code> What is it ? <code>__eou__</code> Can you mail this at the post office for me ? I have to go to the doctor for an appointment and I don't have enough time to go to the post office . <code>__eou__</code>	KLA: i'm sorry , but i'm not sure . i don't want to have a look at the moment . KLA+BOW: oh , i'm sorry , but it's a pleasure to be in my room and again-b . CO+SS: ok , i will . do you have any other questions about the company ? Gold: Sure . And I'd like you to get some cold medicine for me when you go to the doctor's .
I heard you've found a new job . <code>__eou__</code> Yes , I'm now working in a company . <code>__eou__</code> So you're quite happy with your new job ? <code>__eou__</code>	KLA: to be honest , but i'm not familiar with my friends . KLA+BOW: i'd like to . but i would like my mind . CO+SS: not really , but how about you , sue ? Gold: Right . I enjoy what I'm doing .

level coherence and sentence-level topic match. BOW, though not good at the NLL metric, performed remarkably well on this metric, which implies BOW is beneficial for the decoder to generate the correct high-level meaning but fails to transform the meaning to a fluent sentence. In contrast, FB has a relative lower on-topic similarity score compared with its performance on the token-level likelihood.

3.4.3 Human Evaluation

The accurate evaluation of dialogue systems is an open problem. To validate the previous metric-based results, we further conduct a human evaluation on several models. We randomly sampled 100 context from the test corpus and apply 6 different models to generate the best response with beam search. The evaluation is conducted only on the Dailydialog corpus since it is closer to our daily conversation and easier for humans to make the judgement. All the generated responses, together with the dialogue context, are then randomly shuffled and judged on the crowdsourcing website CrowdFlower. People are asked to judge the plausibility of the generated response by giving a binary score in three aspects: grammaticality, coherence with the dialogue context and diversity (ensure the response is not a dull sentence). 54 people are finally involved in evaluating the total 600 responses, each is judged by 3 different people and the score agreed by most people is adopted. We set each person

can judge at most 50 responses and filter by manually-set test questions.

The results shows that our model generates highly fluent sentences compared to other approaches. KLA+BOW, as expected, receives the lowest score on fluency. Our model also achieves relative good scores on coherence and diversity, implying novel responses related to the conversation topic can be generated by our model. However, we notice the human evaluation is rather subjective and not reliable enough. If a sentence is influent, humans tend to reject it though the topic might be coherent and the content might be diverse. It is difficult to give an objective score separately for all the three aspects. We can see models with lower scores on fluency normally also receive lower scores on the other two fields like KLA+BOW and FB-all. Therefore, we consider this evaluation only as a complement to the metric-based results, indicating that humans agree with the generations of our models more than with the others.

Table 3.4: Human Judgements for models trained on Dailymail corpus, F refers to fluent, C refers to coherence and D refers to diversity.

Model	F(%)	C(%)	D(%)
KLA	76	35	50
KLA+DO	80	41	57
KLA+BOW	70	36	48
FB-all	74	29	34
CO+DO	82	49	54
CO+SS	89	44	51

Table 3.3 shows example generated responses. We can see the our improved collaborative VED model with scheduled sampling can more accurately identify the topic and generate more coherent responses. Standard KL-annealing tends to generate smooth sentences but irrelevant to the context. Imposing an additional BOW loss can increase the probability of correctly capturing the main topic, but the generated responses are sometimes grammatically wrong, as also has been shown from the metric-based results. In the first example, the context is about taxi drivers' request for reducing gasoline price, the response from KLA is a fluent natural sentence but not closely related to the context. Model KLA+BOW starts with a reasonable beginning but ends up with influent continuations. Though influent, KLA+BOW model does capture the main topic about price, indicating it can successfully predict the order-insensitive bag of words but fail to establish a natural sentence. In contrast, our model is not only a fluent sentence, but also close to the topic. More importantly, it brings some new information "I want to buy a new car" and is helpful to an interactive conversation. Similar conditions can be seen in the other two examples.

3.5 CONCLUSION

Variational encoder-decoders and recurrent neural networks are powerful in representation learning and natural language processing respectively. Though recently quite a few work has started to apply them on dialogue generation, the training process is still unstable and the performance is hard to be guaranteed. In this work, we thoroughly analyze the reason of the training difficulty and compare different current approaches, then propose a new framework that allows effectively combining these two structures in dialogue generation. We split the whole structure into two parts for more flexible prior and posterior latent variable distributions. The training process is simple, efficient and scales well to large datasets.

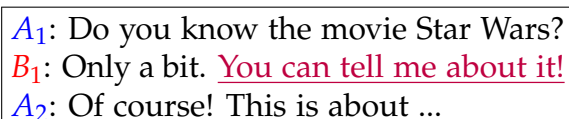
We demonstrate the superiority of our model over other popular methods on two dialogue corpus. Experiments show that our model samples latent variables with more flexible distributions without sacrificing recurrent neural network's capability of synthesizing coherent sentences. Without losing generality, our model should be able to apply on any seq2seq tasks, which we leave for future work.

The proposed training framework is still limited to the maximum likelihood objective. Later on we will show this objective is not suitable for dialogue generation and explain how we can improve over it.

As explained in the last chapter, sequence-to-Sequence (seq2seq) models, though being highly efficient in learning the backbone of human-computer communications, they suffer from the problem of strongly favoring short generic responses. In this chapter, we argue that a good response should smoothly connect both the preceding dialogue history and the following conversations. We strengthen this connection through mutual information maximization. To sidestep the non-differentiability of discrete natural language tokens, we introduce an auxiliary continuous code space and map such code space to a learnable prior distribution for generation purpose. Experiments on two dialogue datasets validate the effectiveness of our model, where the generated responses are closely related to the dialogue context and lead to more interactive conversations (Shen *et al.*, 2018b).

4.1 INTRODUCTION

With the availability of massive online conversational data, there has been a surge of interest in building open-domain chatbots with data-driven approaches. Recently, the neural network based sequence-to-sequence (seq2seq) framework (Sutskever *et al.*, 2014; Cho *et al.*, 2014) has been widely adopted. In such a model, the encoder, which is typically a recurrent neural network (RNN), maps the source tokens into a fixed-sized continuous vector, based on which the decoder estimates the probabilities on the target side word by word. The whole model can be efficiently trained by maximum likelihood (MLE) and has demonstrated state-of-the-art performance in various domains. However, this architecture is not suitable for modeling dialogues. Recent research has found that while the seq2seq model generates syntactically well-formed responses, they are prone to being off-context, short, and generic. (e.g., "I don't know" or "I am not sure") (Li *et al.*, 2016a; Serban *et al.*, 2016). The reason lies in the one-to-many alignments in human conversations, where one dialogue context is open to multiple potential responses. When optimizing with the MLE objective, the model tends to have a strong bias towards safe responses as they can be literally paired with arbitrary dialogue context without semantical or grammatical



A₁: Do you know the movie Star Wars?
B₁: Only a bit. You can tell me about it!
A₂: Of course! This is about ...

Figure 4.1: A conversation in real life

contradictions. These safe responses break the dialogue flow without bringing any useful information and people will easily lose interest in continuing the conversation.

In this paper, we propose NEXUS Network which aims at producing more on-topic responses to maintain an interactive conversation flow. Our assumption is that a good response should serve as a “nexus”: connecting and being informative to both the preceding dialogue context and the follow-up conversations. For example, in Figure 4.1, the response from B_1 is a smooth connection, where the first half indicates the preceding context is a “Do you know” question and the second half informs that the follow-up would be an introduction about *Star Wars*. We establish this connection by maximizing the mutual information (MMI) of the current utterance with both the past and future contexts. In this way, generic responses can be largely discouraged as they contain no valuable information and thus have only weak correlations with the surrounding context. To enable efficient training, two challenges exist.

The first challenge comes from the discrete nature of language tokens, hindering efficient gradient descent. One strategy is to estimate the gradient by methods like Gumbel-Softmax (Maddison *et al.*, 2017; Jang *et al.*, 2017) or REINFORCE algorithm (Williams, 1992), which has been applied in many NLP tasks (He *et al.*, 2016a; Shetty *et al.*, 2017; Gu *et al.*, 2018b; Paulus *et al.*, 2018), but the trade-off between bias and variance of the estimated gradient is hard to reconcile. The resulting model usually strongly relies on sensitive hyper-parameter tuning, careful pre-train and task-specific tricks. (Li *et al.*, 2016a; Wang *et al.*, 2017b) avoid this non-differentiability problem by learning a separate backward model to rerank candidate responses in the testing phase while still adhering to the MLE objective for training. However, the candidate set normally suffers from low diversity and a huge sample size is needed for good performance (Li *et al.*, 2016c).

The second challenge relates to the unknown future context in the testing phase. In our framework, both the history and future context need to be explicitly observed in order to compute the mutual information. When applying it to generating tasks where only the history context is given, there is no way to explicitly take into account the future information. Therefore, reranking-based models do not apply here. (Li *et al.*, 2016d) addresses future information by policy learning, but the model suffers from high variance due to the enormous sequential search space. (Serban *et al.*, 2017c; Zhao *et al.*, 2017c; Shen *et al.*, 2017d) adopt the variational inference strategy to reduce the training variance by optimizing over latent continuous variables. However, they all stick to the original MLE objective and no connection with the surrounding context is considered.

In this work, we address both challenges by introducing an auxiliary continuous code space which is learned from the whole dialogue flow. At each time step, instead of directly optimizing discrete utterances, the current, past and future utterances are all trained to maximize the mutual information with this code space. Furthermore, a learnable prior distribution is simultaneously optimized to predict the corresponding code space, enabling efficient sampling in the testing phase without getting access to the ground-truth future conversation. Extensive experiments have been conducted to validate the superiority of our framework. The generated responses clearly

demonstrate better performance with respect to both coherence and diversity.

4.2 MODEL STRUCTURE

4.2.1 Motivation

Let u_i be the i th utterance within a dialogue flow. The dialogue history H_{i-1} contains all the preceding context u_1, u_2, \dots, u_{i-1} and F_{i+1} denotes the future conversations u_{i+1}, \dots, u_T . The objective of our model is to find the decoding probability $p_\theta(u_i|H_{i-1}, F_{i+1})$ that maximizes the mutual information $I(H_{i-1}, u_i)$ and $I(u_i, F_{i+1})$. Formally, the objective is:

$$\begin{aligned} \max_{\theta} \lambda_1 I(H_{i-1}, u_i) + \lambda_2 I(u_i, F_{i+1}) \\ u_i \sim p_\theta(u_i|H_{i-1}, F_{i+1}) \end{aligned} \quad (4.1)$$

λ_1 and λ_2 adjusts the relative weight. Mutual information is defined over $p_\theta(u_i|H_{i-1}, F_{i+1})$ and the empirical distribution $p(H_{i-1}, F_{i+1})$. Now we assume the future context F_{i+1} is known to us when training the decoding probability, we will address the unknown future problem later.

Directly optimizing with this objective is unfortunately infeasible because the exact computation of mutual information is intractable, and backpropagating through sampled discrete sequences is notoriously difficult to train. The discontinuity prevents the direct application of the reparameterization trick (Kingma and Welling, 2014). Low-variance relaxations like Gumbel-Softmax (Jang *et al.*, 2017), semantic hashing (Kaiser *et al.*, 2018) or vector quantization (van den Oord *et al.*, 2017) lead to biased gradient estimations, which are accumulated as the sequence becomes longer. The Monte-Carlo-Simulation is unbiased but suffers from high variances. Designing a reasonable control variate for variance reduction is an extremely tricky task (Mnih and Gregor, 2014; Tucker *et al.*, 2017). For this sake, we propose replacing u_i with a continuous code space c learned from the whole dialogue flow.

4.2.2 Continuous Code Space

We define the continuous code space c to follow the Gaussian probability distribution with a diagonal covariance matrix conditioning on the whole dialogue:

$$c \sim p_\phi(c|H_{i-1}, F_i) = \mathcal{N}(\mu_c, \sigma_c^2 \mathbb{I}|H_{i-1}, F_i) \quad (4.2)$$

The dialogue history H_{i-1} is encoded into vector \tilde{H}_{i-1} by a forward hierarchical GRU model E_f as in (Serban *et al.*, 2016). The future conversation, including the current utterance, is encoded into \tilde{F}_i by a backward hierarchical GRU E_b . \tilde{H}_{i-1} and \tilde{F}_i are concatenated and a multi-layer perceptron is built on top of them to estimate the Gaussian mean and covariance parameters. The code space is trained to infer

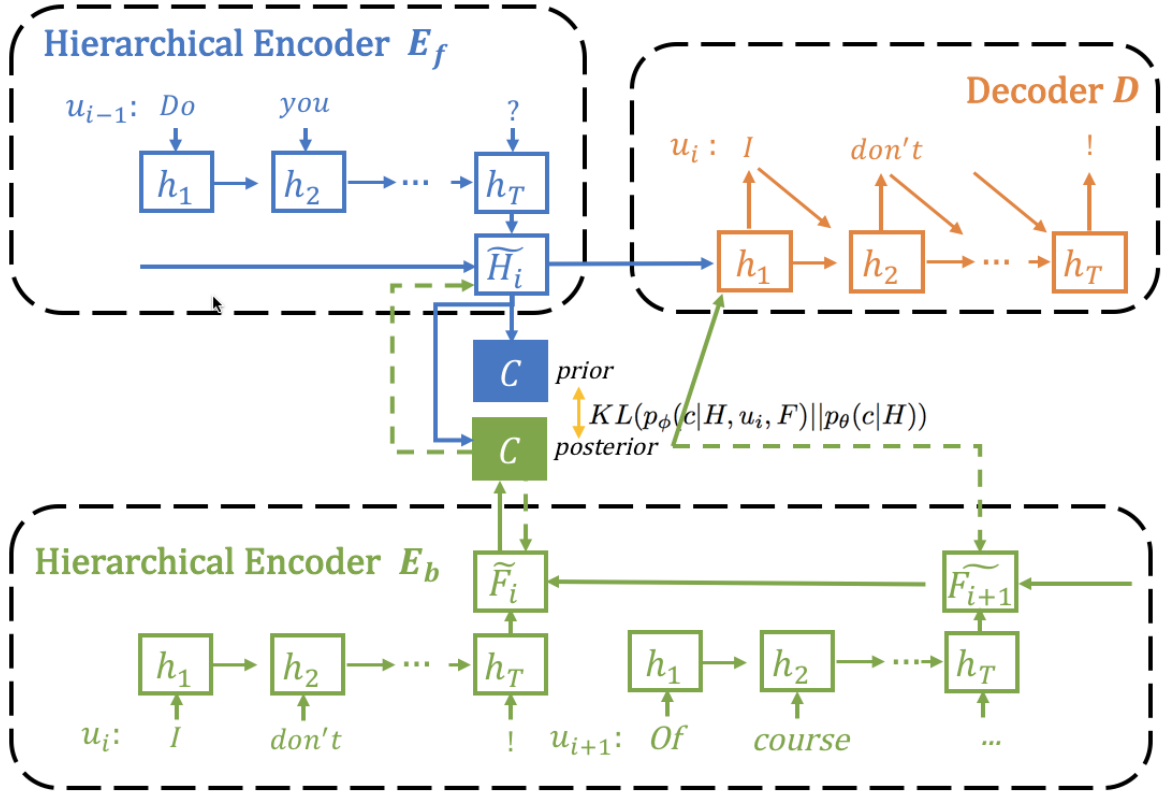


Figure 4.2: Framework of NEXUS Networks. Full line indicates the generative model to generate the continuous code and corresponding responses. Dashed line indicates the inference model where the posterior code is trained to infer the history, current and future utterances. Both parts are simultaneously trained by gradient descent.

the encoded history \tilde{H}_{i-1} and future \tilde{F}_{i+1} . The full optimizing objective is:

$$\begin{aligned}
 \mathcal{L}(c) &= \max_{\phi} \mathbb{E}_{p_{\phi}(H_{i-1}, F_i, c)} [\lambda_1 \log p_{\phi}(\tilde{H}_{i-1} | c) \\
 &\quad + \lambda_2 \log p_{\phi}(\tilde{F}_{i+1} | c)] \\
 p_{\phi}(H_{i-1}, F_i, c) &= p(H_{i-1}, F_i) p_{\phi}(c | H_{i-1}, F_i) \\
 p_{\phi}(\tilde{H}_{i-1} | c) &= \mathcal{N}(\mu_{H_i}, \sigma_{H_i}^2 \mathbb{I} | c) \\
 p_{\phi}(\tilde{F}_{i+1} | c) &= \mathcal{N}(\mu_{F_{i+1}}, \sigma_{F_{i+1}}^2 \mathbb{I} | c)
 \end{aligned} \tag{4.3}$$

where \tilde{H}_{i-1} and \tilde{F}_{i+1} are also assumed to be Gaussian distributed given c with mean and covariance estimated from multi-layer perceptrons. We infer the encoded vectors instead of the original sequences for three reasons. Firstly, inferring dense vectors is parallelizable and computationally much cheaper than autoregressive decoding, especially when the context sequences could be unlimitedly long. Secondly, sequence vectors can capture more holistic semantic-level similarity than individual tokens. Lastly, It can also help alleviate the posterior collapsing issue (Bowman *et al.*, 2016) when training variational inference models on text (Chen *et al.*, 2017;

Shen *et al.*, 2018c), which we will use later. It can be shown that the above objective maximizes a lower bound of $\lambda_1 I(H_{i-1}, c) + \lambda_2 I(c, F_{i+1})$, given the conditional probability $p_\phi(c|H_{i-1}, F_i)$. The proof is a direct extension of the derivation in (Chen *et al.*, 2016), followed by the Data Processing Inequality (Beaudry and Renner, 2012) that the encoding function can only reduce the mutual information. As the sampling process contains only Gaussian continuous variables, the above objective can be trained through the reparameterization trick (Kingma and Welling, 2014), which is a low-variance, unbiased gradient estimator (Burda *et al.*, 2015). After training, samples from $p_\phi(c|H_{i-1}, F_i)$ hold high mutual information with both the history and future context. The next step is then transferring the continuous code space to reasonable discrete natural language utterances.

4.2.3 Decoding from Continuous Space

Our decoder transfers the code space c into the ground-truth utterance u_i by defining the probability distribution $p(u_i|H_{i-1}, c)$, which is implemented as a GRU decoder going through u_i word by word to estimate the output probability. The encoded history H_{i-1} and code space c are concatenated as an extra input at each time step. The loss function for the decoder is then:

$$\begin{aligned} \mathcal{L}(d) &= \max_{\phi} \mathbb{E}_{p_\phi(H_{i-1}, F_i, c)} \log p_\phi(u_i|H_{i-1}, c) \\ p_\phi(H_{i-1}, F_i, c) &= p(H_{i-1}, F_i) p_\phi(c|H_{i-1}, F_i) \end{aligned} \quad (4.4)$$

which can be proved to be the lower bound of the conditional mutual information $I(u_i, c|H_{i-1})$. By maximizing the conditional mutual information, c_i is trained to maintain as much information about the target sequence u_i as possible.

Combining Eq. 4.3 and 4.4, our model until now can be viewed as optimizing a lower bound of the following objective:

$$\begin{aligned} \max_{\phi} \lambda_1 I(H_{i-1}, c) + \lambda_2 I(c, F_{i+1}) + I(u_i, c|H_{i-1}) \\ c \sim p_\phi(c|H_{i-1}, F_i) \end{aligned} \quad (4.5)$$

Compared with the original motivation in Eq. 4.1, we sidestep the non-differentiability problem by replacing u_i with a continuous code space c , then forcing u_i to contain the same information as maintained in c by additionally maximizing the mutual information between them.

Nonetheless, Eq. 4.5 and Eq. 4.1 might lead to different optimums as mutual information does not satisfy the transitive law. In the extreme case, different dimensions of c could individually maintain information about history, current and future conversations and the conversations themselves do not share any dependency relation. To avoid this issue, we restrict the dimension of c to be smaller than that of the encoded vectors. In this case, optimizing Eq. 4.5 will favor utterances having stronger correlations with the surrounding context to achieve a higher total mutual information.

4.2.4 Learnable Prior Distribution for Unknown Future

The last problem is the sampling mechanism of c in Eq. 4.2, which conditions on the ground-truth future conversation. In the testing phase, when we have no access to it, we cannot perform the decoding process as in Eq. 4.4. To allow for decoding with only the history context, we need to learn an appropriate prior distribution $p_\theta(c|H_{i-1})$ for c . In the ideal case, we would like

$$p_\theta(c|H_{i-1}) = \sum_{F_i} p_\phi(c|H_{i-1}, F_i) = p_\phi(c|H_{i-1}) \quad (4.6)$$

However, $p_\phi(c|H_{i-1})$ is intractable as it integrates over all possible future conversations. We apply variational inference on c to maximize the variational lower bound (Jordan *et al.*, 1999):

$$\begin{aligned} \mathcal{L}(p) &= \max_{\theta, \phi} \mathbb{E}_{p_\phi(c|H_{i-1}, F_i)} \log p_\theta(\tilde{F}_i|H_{i-1}, c) \\ &\quad - KL(p_\phi(c|H_{i-1}, F_i) || p_\theta(c|H_{i-1})) \\ p_\theta(\tilde{F}_i|H_{i-1}, c) &\sim \mathcal{N}(\mu_{F_i}, \sigma_{F_i}^2 \mathbb{I} | H_{i-1}, c) \\ p_\theta(c|H_{i-1}) &\sim \mathcal{N}(\mu_{prior}, \sigma_{prior}^2 \mathbb{I} | H_{i-1}) \end{aligned} \quad (4.7)$$

It can be reformulated as maximizing:

$$\begin{aligned} &\mathbb{E}_{p_\phi(c|H_{i-1})} KL(p_\phi(\tilde{F}_i|H_{i-1}, c) || p_\theta(\tilde{F}_i|H_{i-1}, c)) \\ &\quad - KL(p_\phi(c|H_{i-1}) || p_\theta(c|H_{i-1})) \end{aligned} \quad (4.8)$$

We can see it implicitly matches $p_\phi(c|H_{i-1})$ to a tractable Gaussian distribution $p_\theta(c|H_{i-1})$ by minimizing the KL divergence between them. It also functions as a regularizer to prevent overfitting when learning $p_\phi(c|H_{i-1}, F_i)$. In the testing phase, we can sample c from the learned prior distribution $p_\theta(c|H_{i-1})$, then generate a response based on it.

4.2.5 Summary

To sum up, the total objective function of our model is:

$$\mathcal{L} = \mathcal{L}(c) + \mathcal{L}(d) + \mathcal{L}(p) \quad (4.9)$$

Weighting can be added to individual loss functions for better performance, but we find it enough to maintain equal weights and avoid extra hyperparameters. All the parameters are simultaneously updated by gradient descent except for the encoders E_f and E_b , which only accept gradients from $\mathcal{L}(d)$ since otherwise the model can easily learn to encode no information for a lower reconstruction loss in $\mathcal{L}(c)$ and $\mathcal{L}(p)$. An overview of our training procedure is depicted in Fig. 4.2.

4.3 RELATIONSHIP TO EXISTING METHODS

MMI decoding. MMI decoder was proposed by (Li *et al.*, 2016a) and further extended in (Wang *et al.*, 2017b). The basic idea is the same as our model by maximizing the mutual information with the dialogue context. However, the MMI principle is applied only at the testing phase rather than the training phase. As a result, it can only be used to evaluate the quality of a generation by estimating its mutual information with the context. To apply it in a generative task, we have to first sample some candidate responses with the seq2seq model, then rerank them by accounting for the MMI score. Our model differs from it in that we directly estimate the decoding probability thus no post-sampling rerank is needed. Moreover, we further include the future context to strengthen the connection role of the current utterances.

Conditional Variational Autoencoder. The idea of learning an appropriate prior distribution in Eq. 4.7 is essentially a conditional variational autoencoder (Sohn *et al.*, 2015) where the accumulated posterior distribution is trained to stay close to a prior distribution. It has also been applied in dialogue generation (Serban *et al.*, 2017c; Zhao *et al.*, 2017c). However, all the above methods stick to the MLE objective function and do not optimize with respect to the mutual information. As we will show in the experiment, they fail to learn the correlation between the utterance and its surrounding context. The generation diversity of these models comes more from the sampling randomness of the prior distribution rather than from the correct understanding of context correlation. Moreover, they suffer from the posterior collapsing problem (Bowman *et al.*, 2016) and require special tricks like KL-annealing, BOW loss or word drop-out (Shen *et al.*, 2018c). Our model does not have such problems.

Deep Reinforcement Learning Dialogue Generation. (Li *et al.*, 2016d) first considered future success in dialogue generation and applied deep reinforcement learning to encourage more interactive conversations. However, the reward functions are intuitively hand-crafted. The relative weight for each reward needs to be carefully tuned and the training stage is unstable due to the huge search space. In contrast, our model maximizes the mutual information in the continuous space and trains the prior distribution through the reparameterization trick. As a result, our model can be more easily trained with a lower variance. Throughout our experiment, the training process of NEXUS network is rather stable and much less data-hungry. The MMI objective of our model is theoretically more sound and no manually-defined rules need to be specified.

Model	DailyDialog			Twitter		
	Average	Greedy	Extreme	Average	Greedy	Extreme
Greedy	0.443	0.376	0.328	0.510	0.341	0.356
Beam	0.437	0.350	0.369	0.505	0.345	0.352
MMI	0.457	0.371	0.371	0.518	0.353	0.365
RL	0.405	0.329	0.305	0.460	0.349	0.323
VHRED	0.491	0.375	0.313	0.525	0.389	0.372
NEXUS-H	0.479	0.381*	0.385*	0.558*	0.392	0.373
NEXUS-F	0.476	0.383*	0.373	0.549*	0.393	0.386*
NEXUS	0.488	0.392*	0.384*	0.556*	0.397*	0.391*

Table 4.1: Results of embedding-based metrics. * indicates statistically significant difference ($p < 0.05$) from the best baselines. The same mark is used in Table 4.2

4.4 EXPERIMENTS

4.4.1 Dataset and Training Details

We run experiments on the DailyDialog (Li *et al.*, 2017c) and Twitter corpus (Ritter *et al.*, 2011). DailyDialog contains 13118 daily conversations under ten different topics. This dataset is crawled from various websites for English learner to practice English in daily life, which is high-quality, less noisy but relatively smaller. In contrast, the Twitter corpus is significantly larger but contains more noise. We obtain the dataset as used in (Serban *et al.*, 2017c) and filter out tweets that have already been deleted, resulting in about 750,000 multi-turn dialogues. The contents have more informal, colloquial expressions which makes the generation task harder. These two datasets are randomly separated into training/validation/test sets with the ratio of 10:1:1.

In order to keep our model comparable with the state-of-the-art, we keep most parameter values the same as in (Serban *et al.*, 2017c). We build our vocabulary dictionary based on the most frequent 20,000 words for both corpus and map other words to a UNK token. The dimensionality of the code space c is 100. We use a learning rate of 0.001 for DailyDialog and 0.0002 for Twitter corpus. The batch size is fixed to 128. The word vector dimension is 300 and is initialized with the public Word2Vec (Mikolov *et al.*, 2013a) embeddings trained on the Google News Corpus. The probability estimators for the Gaussian distributions are implemented as 3-layer perceptrons with the hyperbolic tangent activation function. As mentioned above, when training NEXUS models, we block the gradient from $\mathcal{L}(c)$ and $\mathcal{L}(p)$ with respect to E_f and E_b to encourage more meaningful encodings. The UNK token is prevented from being generated in the test phase. We implemented all the models with the open-sourced Python library Pytorch (Paszke *et al.*, 2017) and optimized using the Adam optimizer (Kingma and Ba, 2015).

4.4.2 Compared Models

We conduct extensive experiments to compare our model against several representative baselines.

Seq2Seq: Following the same implementation as in (Vinyals and Le, 2015), the seq2seq model serves as a baseline. We try both greedy decoding and beam search (Graves, 2012) with beam size set to 5 when testing.

MMI: We implemented the bidirectional-MMI decoder as in (Li *et al.*, 2016a), which showed better performance over the anti-LM model. The hyperparameter λ is set to 0.5 as suggested. 200 candidates per context are sampled for re-ranking.

VHRED: The VHRED model is essentially a conditional variational autoencoder with hierarchical encoders (Serban *et al.*, 2017c; Zhao *et al.*, 2017c). To alleviate the posterior collapsing problem, we apply the KL-annealing trick and early stop with the step set as 12,000 for the DailyDialog and 75,000 for the Twitter corpus.

RL: Deep reinforcement learning chatbot as in (Li *et al.*, 2016d). We use all the three reward functions mentioned in the paper and keep the relative weights the same as in the original paper. Policy network is initialized with the above-mentioned MMI model.

NEXUS-H: NEXUS network maximizing mutual information only with the history ($\lambda_2 = 0$).

NEXUS-F: NEXUS network maximizing mutual information only with the future ($\lambda_1 = 0$).

NEXUS: NEXUS network maximizing mutual information with both the history and future.

NEXUS-H and NEXUS-F are implemented to help us better analyze the effects of different components in our model. The hyperparameters λ_1 and λ_2 in NEXUS are set to be 0.5 and 1 respectively as we find history vector is consistently easier to be reconstructed than the future vector (4.5.6).

4.4.3 Metric-based Performance

Embedding Score. We conducted three embedding-based evaluations (average, greedy and extrema) (Liu *et al.*, 2016), which map responses into vector space and compute the cosine similarity (Rus and Lintean, 2012). The embedding-based metrics can to a large extent capture the semantic-level similarity between generated responses and ground truth. We represent words using Word2Vec embeddings trained on the Google News Corpus. We also measure the uncertainty of the score by assuming each data point is independently Gaussian distributed. The standard deviation yields the 95% confidence interval (Barany *et al.*, 2007). Table 4.1 reports the embedding scores on both datasets. NEXUS network significantly outperforms the best baseline model in most cases. Notably, NEXUS can absorb the advantages from both NEXUS-H and NEXUS-F. The history and future information seem to help the model from different perspectives. Taking into account both of them does not create a conflict and the combination leads to an overall improvement. RL

Model	DailyDialog			Twitter		
	BLEU-1	BLEU-2	BLEU-3	BLEU-1	BLEU-2	BLEU-3
Greedy	0.394	0.245	0.157	0.340	0.203	0.116
Beam	0.386	0.251	0.163	0.338	0.205	0.112
MMI	0.407	0.269	0.172	0.347	0.208	0.118
RL	0.298	0.186	0.075	0.314	0.199	0.103
VHRED	0.395	0.281	0.190	0.355	0.211	0.124
NEXUS-H	0.418	0.279	0.199*	0.366*	0.212	0.126
NEXUS-F	0.399	0.260	0.167	0.359	0.213	0.123
NEXUS	0.424*	0.276	0.198*	0.363*	0.220*	0.131*

Table 4.2: Results of BLEU score. It is computed based on the smooth BLEU algorithm (Lin and Och, 2004). p-value interval is computed base on the altered bootstrap resampling algorithm (Riezler and Maxwell, 2005)

performs rather poorly on this metric, which is understandable as it does not target the ground-truth responses during training (Li *et al.*, 2016d).

BLEU Score. BLEU is a popular metric that measures the geometric mean of the modified n-gram precision with a length penalty (Papineni *et al.*, 2002). Table 4.2 reports the BLEU 1-3 scores. Compared with embedding-based metrics, the BLEU score quantifies the word-overlap between generated responses and the ground-truth. One challenge of evaluating dialogue generation by BLEU score is the difficulty of accessing multiple references for the one-to-many alignment relation. Following (Sordoni *et al.*, 2015; Zhao *et al.*, 2017c; Shen *et al.*, 2018c), for each context, 10 more candidate references are acquired by using information retrieval methods (see Appendix 4.5.4 for more details). All candidates are then passed to human annotators to filter unsuitable ones, resulting in 6.74 and 5.13 references for DailyDialog and Twitter dataset respectively. The human annotation is costly, so we evaluate it on 1000 sampled test cases for each dataset. As the BLEU score is not the simple mean of individual sentence scores, we compute the 95% significance interval by bootstrap resampling (Koehn, 2004; Riezler and Maxwell, 2005). As can be seen, NEXUS network achieves best or near-best performances with only greedy decoders. NEXUS-H generally outperforms NEXUS-F as the connection with future context is not explicitly addressed by the BLEU score metric. MMI and VHRED bring minor improvements over the seq2seq model. Even when evaluated on multiple references, RL still performs worse than most models.

Connecting the preceding. We define two metrics to evaluate the model’s capability of “connecting the preceding context”: **AdverSuc** and **Neg-PMI**. AdverSuc measures the coherence of generated responses with the provided context by learning an adversarial discriminator (Li *et al.*, 2017a) on the same corpus to distinguish

Model	AdverSuc		Neg-PMI		#Turns		Distinct-1		Distinct-2		Pri	Post	Flu
Greedy	0.21	0.13	47.4	45.8	0.2	0.6	.019	.017	.096	.072	0.45	0.04	0.92
Beam	0.16	0.12	47.2	45.3	0.2	0.7	.026	.019	.103	.086	0.52	0.06	0.90
MMI	0.30	0.19	45.6	43.2	1.1	1.6	.042	.025	.247	.117	0.56	0.13	0.89
RL	0.13	0.11	45.0	42.6	2.3	2.3	.048	.033	.324	.287	0.46	0.15	0.69
VHRED	0.19	0.16	46.8	44.7	1.7	1.1	.255	.106	.431	.311	0.42	0.22	0.92
NEXUS-H	0.36	0.21	44.1	41.8	2.0	1.8	.263	.108	.454	.306	0.66	0.20	0.92
NEXUS-F	0.22	0.12	47.1	45.9	2.6	2.2	.288	.117	.466	.325	0.51	0.31	0.94
NEXUS	0.35	0.18	44.6	41.4	2.8	2.5	.282	.119	.470	.329	0.70	0.33	0.93
GROUND	0.87	0.73	40.5	38.1	4.8	4.0	.390	.215	.522	.495	0.92	0.67	0.97

Table 4.3: Coherence, diversity and human evaluations. Left: DailyDialog results, right: Twitter results

coherent responses from randomly sampled ones. We encode the context and response separately with two different LSTM neural networks and output a binary signal indicating coherent or not⁵. The AdverSuc value is reported as the success rate that the model fools the classifier into believing its false generations ($p(\text{generated} = \text{coherent}) > 0.5$). Neg-PMI measures the negative pointwise mutual information value $-\log p(c|r)/p(c)$ between the generated response r and the dialogue context c . $p(c|r)$ is estimated by training a separate backward seq2seq model. As $p(c)$ is a constant, we ignore it and only report the value of $-\log p(c|r)$. A good model should achieve a higher AdverSuc and a lower Neg-PMI. The results are listed in Table 4.3. We can see there is still a big gap between ground-truth and synthesized responses. As expected, NEXUS-H leads to the most significant improvement. MMI model also performs remarkably well, but it requires post-reranking thus the sampling process is much slower. VHRED and NEXUS-F do not help much here, sometimes even slightly degrade the performance. We also tried removing the history context when computing the posterior distribution in VHRED, the resulting model has similar performance among all metrics, which suggests VHRED itself cannot actually learn the correlation pattern with the preceding context. Surprisingly, though RL explicitly set the coherence score as a reward function, its performance is far from satisfying. We assume RL requires much more data to learn the appropriate policy than other models and the training process suffers from a higher variance. The result is thus hard to be guaranteed.

Connecting the following. We measure the model’s capability of “connecting the following context” from two perspectives: number of the simulated turns and diversity of generated responses. We apply all models to generate multiple turns until a generic response is reached. The set of generic responses is manually

⁵We apply the same architecture as in (Lu *et al.*, 2017b). In our experiment, the discriminator performs reasonably well in the 4 scenarios outlined in (Li *et al.*, 2017a) and thus can be used as a fair evaluation metric.

examined to include all utterances providing only passive dull replies⁶. The number of generated turns can reflect the time that a model can maintain an interactive conversation. The results are reflected in the **#Turns** column in Table 4.3. As in (Li *et al.*, 2016a), we measure the diversity by the percentage of distinct unigrams (**Distinct-1**) and bigrams (**Distinct-2**) in all generated responses. Intuitively a higher score on these three metrics implies a more interactive generation system that can better connect the future context. Again, NEXUS network dominates most fields. NEXUS-F brings more impact than NEXUS-H as it explicitly encourages more interactive turns. Most seq2seq models fail to provide an informative response in the first turn. The MMI-decoder does not change much, possibly because the sampling space is not large enough, a more diverse sampling mechanism (Vijayakumar *et al.*, 2018) might help. NEXUS network can effectively continue the conversation for 2.8 turns for DailyDialog and 2.5 turns for Twitter, which is closest to the ground truth (4.8 and 4.0 turns respectively). It also achieves the best diversity score in both datasets. It is worth mentioning that NEXUS-H also improves over baselines, though not as significantly as NEXUS-F, so NEXUS is not a trade-off but more like an enhanced version from NEXUS-H and NEXUS-F.

In summary, NEXUS network clearly generates higher-quality responses in both coherence and diversity, even in a rather small dataset like DailyDialog. NEXUS-H contributes more to the coherence and NEXUS-F more to the diversity.

4.4.4 Human Evaluation

We also employed crowdsourced judges to provide evaluations for a random sample of 500 items in the DailyDialog test dataset. Participants are asked to assign a binary score to each context-response pair from three perspectives: whether the response coincides with its preceding context (Pri), whether the response is interesting enough for people to continue (Post) and whether the response itself is a fluent natural sentence (Flu). Each sample gets one point if judged as yes and zero otherwise. Each pair is judged by three participants and the score supported by most people is adopted. We also evaluated the inter-annotator consistency by Fleiss’k score (Fleiss, 1971) and obtained k scores of 0.452 for Pri, 0.459 for Post (moderate agreement) and 0.621 for Flu (substantial agreement), which implies most context-response pairs reach a consensus on the evaluation task. We compute the average human score for each model. Unlike metric-based scores, the human evaluation is conducted only on the DailyDialog corpus as it contains less noise and can be more fairly evaluated by human judges. Table 4.3 shows the result in the last three columns. As can be seen, the pri and post human scores are highly correlated with the automatic evaluation metric “coherence” and “#turns”, verifying the validity of these two metrics. As for fluency, there is no significant difference among most models. As we also manually examined, fluency is not a major problem and all models produce mostly well-

⁶We use a simple rule matching method (see Appendix 4.5.5). We manually inspect it on a validation subset and find the accuracy is more than 90%. Similar methods are adopted in (Li *et al.*, 2016d).

Context	Response
Hi , Alice , what are you going to do this sunday ? __eou__	MMI: Oh , that ' s great ! VHRED : Well , it seems that i ' m going to be late . RL : How long he it be ? NEXUS: Nothing really , what ' s up ?
Did you watch the Oscars on Sunday night ? __eou__	MMI: I have no idea . VHRED: That ' s not too bad . RL : It ' s my mum ' s birthday on Sunday . NEXUS: No , nothing much. Could you help me ?
Let ' s go for a drink ! __eou__	MMI: Thanks . VHRED: all right . let me see you then . RL : Thank you inviting me . NEXUS: Sounds good . Where are we going ?
You won ' t want to miss our webinar tomorrow ! __eou__	MMI: lol ! VHRED: That ' s terrific ! RL : Sure, where where ? NEXUS: Thanks for your invitation! I ' m free !
How I read the article ? __eou__	MMI: there!!! VHRED: What ' s good with it ? RL : : I don ' t know what to do with it ! NEXUS: Maybe the force is yourself !

Table 4.4: Examples of context-response pairs. __eou__ denotes end-of-utterance. First three rows are from DailyDialog and the last two rows are from Twitter

formed sentences. Overall, NEXUS network does produce responses that are more acceptable to human judges.

Table 4.4 presents some randomly sampled context-response pairs provided by MMI, VHRED, RL and NEXUS model. We see NEXUS network does generate more interactive outputs than the other three. Though reranked by the bidirectional language model, the MMI decoder still produces quite a few generic responses. VHRED ' s utterances are more diverse, but it only cares about answering to the immediate query and makes no efforts to bring about further topics. Moreover, it also generates more inappropriate responses than the others. RL provides diverse responses but sometimes not fluent or coherent enough. We do observe that NEXUS sometimes generate over-complex questions which are not very natural, as in the second example. But in most cases, it outperforms the others.

4.5 ADDITIONAL INFORMATION

4.5.1 Proof of Eq. 4.3

$$\begin{aligned}
& \lambda_1 I(H, c) + \lambda_2 I(c, F) \\
& \geq \lambda_1 I(\tilde{H}, c) + \lambda_2 I(c, \tilde{F}) \\
& = \lambda_1 \mathbb{E}_{p_\phi(\tilde{H}|c)} \log \frac{p_\phi(\tilde{H}|c)}{p(\tilde{H})} + \lambda_1 \mathbb{E}_{p_\phi(c|\tilde{F})} \log \frac{p_\phi(\tilde{F}|c)}{p(\tilde{F})} \\
& = \lambda_1 \mathbb{E}_{p_\phi(\tilde{H}|c)} \log p_\phi(\tilde{H}|c) + \lambda_1 \mathbb{H}(\tilde{H}) + \lambda_2 \mathbb{E}_{p_\phi(c|\tilde{F})} \log p_\phi(\tilde{F}|c) + \lambda_2 \mathbb{H}(\tilde{F}) \\
& \geq \lambda_1 \mathbb{E}_{p_\phi(\tilde{H}|c)} \log p_\phi(\tilde{H}|c) + \lambda_2 \mathbb{E}_{p_\phi(c|\tilde{F})} \log p_\phi(\tilde{F}|c) \\
& = \lambda_1 \mathbb{E}_{p_\phi(\tilde{H}|c)} \log p_\gamma(\tilde{H}|c) + \lambda_1 KL(p_\phi(\tilde{H}|c) || p_\gamma(\tilde{H}|c)) + \lambda_2 \mathbb{E}_{p_\phi(c|\tilde{F})} \log p_\gamma(\tilde{F}|c) \\
& \quad + \lambda_2 KL(p_\phi(\tilde{F}|c) || p_\gamma(\tilde{F}|c)) \\
& \geq \lambda_1 \mathbb{E}_{p_\phi(\tilde{H}|c)} \log p_\gamma(\tilde{H}|c) + \lambda_2 \mathbb{E}_{p_\phi(c|\tilde{F})} \log p_\gamma(\tilde{F}|c) \\
& = \mathbb{E}_{p_\phi(\tilde{H}u_i F, c)} [\lambda_1 \log p_\gamma(\tilde{H}|c) + \lambda_2 \log p_\gamma(\tilde{F}|c)]
\end{aligned}$$

4.5.2 Proof of Eq. 4.4

$$\begin{aligned}
I(u_i, c|H) &= \mathbb{E}_{p(H)} \mathbb{E}_{p_\phi(u_i c|H)} \log \frac{p_\phi(u_i|Hc)}{p(u_i|H)} \\
&= \mathbb{E}_{p(H)} \mathbb{E}_{p_\phi(u_i c|H)} \log p_\phi(u_i|Hc) + \mathbb{H}(u_i|H) \\
&\geq \mathbb{E}_{p(Hu_i F)} \mathbb{E}_{p_\phi(c|Hu_i F)} \log p_\phi(u_i|Hc) \\
&= \mathbb{E}_{p(Hu_i F)} \mathbb{E}_{p_\phi(c|Hu_i F)} \log p_\gamma(u_i|Hc) + \mathbb{E}_{p_\phi(Hc F)} KL(p_\phi(u_i|Hc) || p_\gamma(u_i|Hc)) \\
&\geq \mathbb{E}_{p(Hu_i F)} \mathbb{E}_{p_\phi(c|Hu_i F)} \log p_\gamma(u_i|Hc)
\end{aligned}$$

4.5.3 Derivation of Eq. 4.8

$$\begin{aligned}
& \mathbb{E}_{p(u_i F|\tilde{H})} [\mathbb{E}_{p_\phi(c|\tilde{H}u_i F)} \log p_\phi(u_i F|c) - KL(p_\phi(c|\tilde{H}u_i F) || p_\theta(c|\tilde{H}))] \\
&= \mathbb{E}_{p(u_i F|\tilde{H})} [\mathbb{E}_{p_\phi(c|\tilde{H}u_i F)} \log \frac{p_\phi(u_i F|c) p_\theta(c|\tilde{H})}{p_\phi(c|\tilde{H}u_i F)}] \\
&= \mathbb{E}_{p(u_i F|\tilde{H})} [\mathbb{E}_{p_\phi(c|\tilde{H}u_i F)} \log \frac{p_\phi(u_i F|c) p_\theta(c|\tilde{H}) p(u_i F|\tilde{H})}{p_\phi(u_i F|\tilde{H}c) p_\phi(c|\tilde{H})}] \\
&= \mathbb{E}_{q_\phi(c|\tilde{H})} KL(p_\phi(u_i F|\tilde{H}c) || p_\phi(u_i F|\tilde{H})) - KL(p_\phi(c|\tilde{H}) || p_\theta(c|\tilde{H})) - \mathbb{H}(u_i F|\tilde{H})
\end{aligned}$$

4.5.4 Information Retrieval Technique for Multiple References

We collected multiple reference responses for each dialogue context in the test set by information retrieval techniques. References are retrieved based on their similarity with the provided context. Responses to the retrieved utterances are used as references. The process of retrieving similar context is as follows: First, we select 1000 candidate utterances using the tf-idf score. These candidates are then mapped to a vector space by summing their contained word vectors. After that, they are reranked based on the average of cosine similarity, Jaccard distance and Euclidean distance with the ground-truth context. The top 10 retrieved responses are passed to human annotators to judge the appropriateness.

4.5.5 Phrases that count as forming dull responses

- 1) i know
- 2) no __eou__(yes __eou__)
- 3) no problem
- 4) lol
- 5) thanks __eou__
- 6) don't know
- 7) don't think
- 8) what ?
- 9) of course
- 10) wtf

Utterances matching one of these phrases are treated as dull responses.

4.5.6 Effect of hyperparameter λ_1/λ_2

Figure 4.3 visualizes the effects of hyperparameters λ_1 and λ_2 . The negative-log-likelihood is decomposed into two parts: decoding cross entropy (CE) as in Eq. 4.4 and KL divergence as in Eq. 4.7. The sum is a lower bound of the true log-likelihood. The optimal ratio is around 0.5 for both datasets, which means only half weights should be given to the history compared with the future context. Two reasons can explain this phenomena. Firstly, future vector is harder to infer than history as it is not explicitly exposed as an input in Eq. 4.3. Secondly, minimizing the KL divergence in Eq. 4.7 pushes the code space to discard information from the future context so that it could vanish to zero. Therefore, more weights should be given to the future context to maintain a balance.

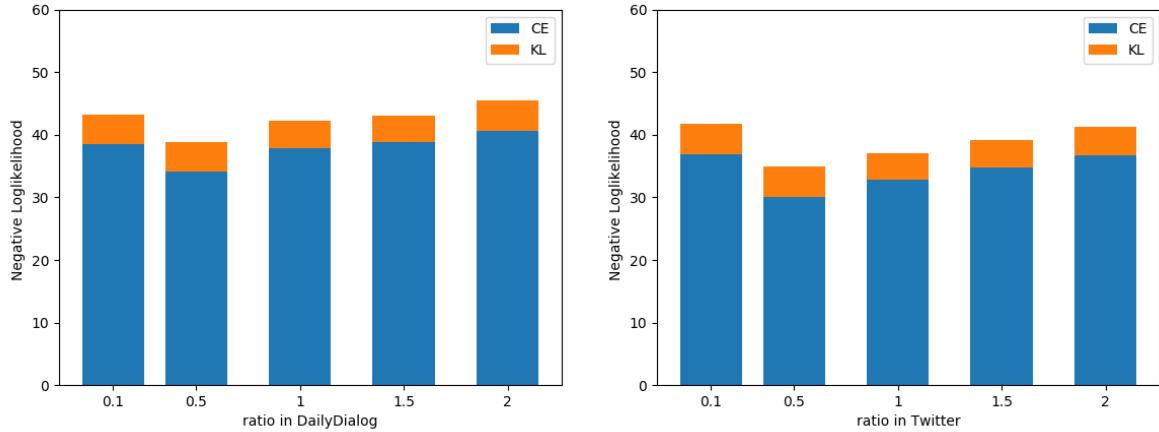


Figure 4.3: Effect of hyperparameter ratio λ_1/λ_2 on two datasets.

4.6 CONCLUSION

In this chapter, we propose “NEXUS Network” to enable more interactive human-computer conversations. The main goal of our model is to strengthen the “nexus” role of the current utterance, connecting both the preceding and the following dialogue context. We compare our model with MMI, reinforcement learning and CVAE-based models. Experiments show that NEXUS network consistently produces higher-quality responses. The model is easier to train, requires no special tricks and demonstrates remarkable generalization capability even in a very small dataset.

Our model can be considered as combining the objective of MMI and CVAE and is compatible with current improving techniques. For example, mutual information can be maximized under a tighter bound using Donsker-Varadhan or f-divergence representation (Donsker and Varadhan, 1983; Nowozin *et al.*, 2016; Belghazi *et al.*, 2018). Extending the code space distribution to more than Gaussian by importance weighted autoencoder (Burda *et al.*, 2015), inverse autoregressive flow (Kingma *et al.*, 2016a) or VamPrior (Tomczak and Welling, 2018) should also help with the performance.

Until now, we explain improved techniques for the vanilla VAE with continuous latent variables on the dialogue generation task. The learned continuous latent variables, however, are non-interpretable. This might not be good for some generation tasks where we hope to have finer-grained control on. In the next chapters we will show how we can use latent variables to stand for discrete, interpretable factors.

Many text generation tasks naturally contain two steps: content selection and surface realization. Current neural encoder-decoder models conflate both steps into a black-box architecture. As a result, the content to be described in the text cannot be explicitly controlled. This paper tackles this problem by decoupling content selection from the decoder. The decoupled content selection is human interpretable, whose value can be manually manipulated to control the content of generated text. The model can be trained end-to-end without human annotations by treating content selection as the latent variable. Different from previous chapters, Latent variables in this chapter has an explicit, interpretable meaning, i.e., content selection of the input. We further propose an effective way to trade-off between performance and controllability with a single adjustable hyperparameter. In both data-to-text and headline generation tasks, our model achieves promising results, paving the way for controllable content selection in text generation (Shen *et al.*, 2019b).⁷

5.1 INTRODUCTION

Many text generation tasks, e.g., data-to-text, summarization and image captioning, can be naturally divided into two steps: content selection and surface realization. The generations are supposed to have two levels of diversity: (1) content-level diversity reflecting multiple possibilities of content selection (what to say) and (2) surface-level diversity reflecting the linguistic variations of verbalizing the selected contents (how to say) (Reiter and Dale, 2000; Nema *et al.*, 2017). Recent neural network models handle these tasks with the encoder-decoder (Enc-Dec) framework (Sutskever *et al.*, 2014; Bahdanau *et al.*, 2015), which simultaneously performs selecting and verbalizing in a black-box way. Therefore, both levels of diversity are entangled within the generation. This entanglement, however, sacrifices the controllability and interpretability, making it difficult to specify the content to be conveyed in the generated text (Qin *et al.*, 2018; Wiseman *et al.*, 2018).

With this in mind, this paper proposes decoupling content selection from the Enc-Dec framework to allow finer-grained control over the generation. Table 5.1 shows an example. We can easily modify the content selection to generate text with various focuses, or sample multiple paraphrases by fixing the content selection.

Though there has been much work dealing with content selection for the Enc-Dec, none of them is able to address the above concerns properly. Current methods can

⁷The source code is available on <https://github.com/chin-gyou/controllable-selection>

Source Sentence: The sri lankan government on Wednesday announced the closure of government schools with immediate effect as a military campaign against tamil separatists escalated in the north of the country.

Selected : sri lankan, closure, schools

Text: sri lanka closes schools .

Selected : sri lankan, Wednesday, closure, schools

Text: sri lanka closes schools on Wednesday.

Selected : sri lankan, closure, schools, military campaign

Text: sri lanka shuts down schools amid war fears.

Selected : sri lankan, announced, closure, schools

Text: sri lanka declares closure of schools.

Table 5.1: Headline generation examples from our model. We can generate text describing various contents by sampling different content selections. The selected source word and its corresponding realizations in the text are highlighted with the same color.

be categorized into the following three classes and have different limits:

1. **Bottom-up:** Train a separate content selector to constrain the attention to source tokens (Gehrmann *et al.*, 2018), but the separate training of selector/generator might lead to discrepancy when integrating them together.
2. **Soft-select:** Learn a soft mask to filter useless information (Mei *et al.*, 2016; Zhou *et al.*, 2017). However, the mask is *deterministic* without any probabilistic variations, making it hard to model the content-level diversity.
3. **Reinforce-select:** Train the selector with reinforcement learning (Chen and Bansal, 2018), which has high training variance and low diversity on content selection.

In this chapter, we treat the content selection as latent variables and train with amortized variational inference (Kingma and Welling, 2014; Mnih and Gregor, 2014). This provides a lower training variance than Reinforce-select. The selector and generator are co-trained within the same objective, the generations are thus more faithful to the selected contents than Bottom-up methods. The selector works by simply masking the attention score in the decoding process without extra burden. Our model is task-agnostic, end-to-end trainable and can be seamlessly inserted into any encoder-decoder architecture. On both the data-to-text and headline generation task, we show our model outperforms others regarding content-level diversity and controllability while maintaining comparable performance. The performance/controllability trade-off can be effectively adjusted by adjusting a single hyperparameter in the training

stage, which constrains an upper bound of the conditional mutual information (CMI) between the selector and generated text (Alemi *et al.*, 2018; Zhao *et al.*, 2018a). A higher CMI leads to stronger controllability with a bit more risk of text disfluency.

In summary, our contributions are **(1)** systematically studying the problem of controllable content selection for Enc-Dec text generation, **(2)** proposing a task-agnostic training framework achieving promising results and **(3)** introducing an effective way to achieve the trade-off between performance and controllability.

5.2 BACKGROUND AND NOTATION

Let X, Y denote a source-target pair. X is a sequence of x_1, x_2, \dots, x_n and can be either some structured data or unstructured text/image depending on the task. Y corresponds to y_1, y_2, \dots, y_m which is a text description of X . The goal of text generation is to learn a distribution $p(Y|X)$ to automatically generate proper text.

The Enc-Dec architecture handles this task with an encode-attend-decode process (Bahdanau *et al.*, 2015; Xu *et al.*, 2015). The encoder first encodes each x_i into a vector h_i . At each time step, the decoder pays attentions to some source embeddings and outputs the probability of the next token by $p(y_t|y_{1:t-1}, C_t)$. C_t is a weighted average of source embeddings:

$$C_t = \sum_i \alpha_{t,i} h_i$$

$$\alpha_{t,i} = \frac{e^{f(h_i, d_t)}}{\sum_j e^{f(h_j, d_t)}} \quad (5.1)$$

d_t is the hidden state of the decoder at time step t . f is a score function to compute the similarity between h_i and d_t (Luong *et al.*, 2015).

5.3 CONTENT SELECTION

Our goal is to decouple the content selection from the decoder by introducing an extra content selector. We hope the content-level diversity can be fully captured by the content selector for a more interpretable and controllable generation process. Following Gehrmann *et al.* (2018); Yu *et al.* (2018), we define content selection as a sequence labeling task. Let $\beta_1, \beta_2, \dots, \beta_n$ denote a sequence of binary selection masks. $\beta_i = 1$ if h_i is selected and 0 otherwise. β_i is assumed to be independent from each other and is sampled from a bernoulli distribution $\mathbf{B}(\gamma_i)$ ⁸. γ_i is the bernoulli parameter, which we estimate using a two-layer feedforward network on top of the source encoder. Text are generated by first sampling β from $\mathbf{B}(\gamma)$ to decide which content to cover, then decode with the conditional distribution $p_\theta(Y|X, \beta)$. The text is

⁸Devlin *et al.* (2019a) have shown that excellent performance can be obtained by assuming such conditionally independence given a sufficiently expressive representation of x , though modelling a richer inter-label dependency is for sure beneficial (Lei *et al.*, 2016; Nallapati *et al.*, 2017).

expected to faithfully convey all selected contents and drop unselected ones. Fig. 5.1 depicts this generation process. Note that the selection is based on the token-level *context-aware* embeddings h and will maintain information from the surrounding contexts. It encourages the decoder to stay faithful to the original information instead of simply fabricating random sentences by connecting the selected tokens. For each source-target pair, the ground-truth selection mask is unknown, so training

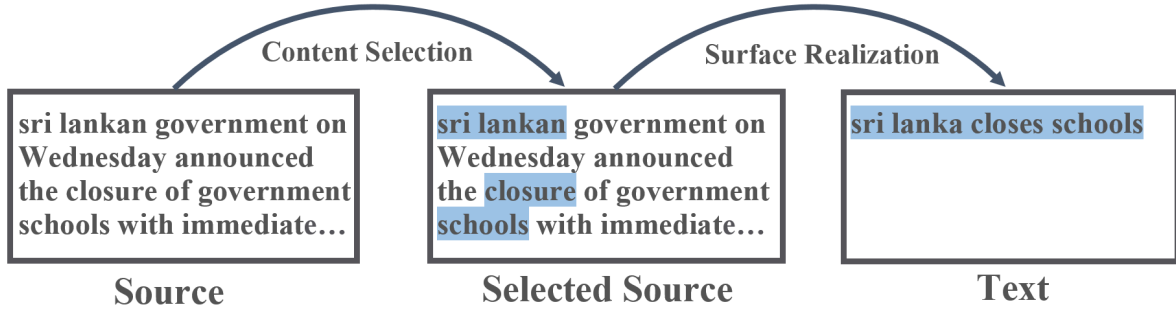


Figure 5.1: Model will select contents based on $\mathbf{B}(\gamma)$, then decode with $p_{\theta}(Y|X, \beta)$. Source-text pairs are available for training, but the ground-truth content selection for each pair is unknown.

is challenging. In the following session, we discuss several training possibilities and introduce the proposed model in detail.

5.3.1 Bottom-up

The most intuitive way is training the content selector to target some heuristically extracted contents. For example, we can train the selector to select overlapped words between the source and target (Gehrmann *et al.*, 2018), sentences with higher tf-idf scores (Li *et al.*, 2018) or identified image objects that appear in the caption (Wang *et al.*, 2017c). A standard encoder-decoder model is independently trained. In the testing stage, the prediction of the content selector is used to hard-mask the attention vector to guide the text generation in a bottom-up way. Though easy to train, Bottom-up generation has the following two problems: (1) The heuristically extracted contents might be coarse and cannot reflect the variety of human languages and (2) The selector and decoder are independently trained towards different objectives thus might not adapt to each other well.

β as Latent Variable: Another way is to treat β as a latent variable and co-train selector and generator by maximizing the marginal data likelihood. By doing so, the selector has the potential to automatically explore optimal selecting strategies best fit for the corresponding generator component.

With this in mind. We design $p_{\theta}(Y|X, \beta)$ by changing the original decoder in the following way: (1) We initialize hidden states of the decoder from a mean pooling over selected contents to inform the decoder which contents to cover and (2)

Unselected contents will be prohibited from being attended to:

$$\begin{aligned} d_0 &= \text{MLP} \left(\frac{1}{n} \left(\sum_i^n \beta_i h_i \right) \right) \\ \alpha_{t,i} &= \frac{e^{f(h_i, d_t)} \beta_i}{\sum_j e^{f(h_j, d_t)} \beta_j} \end{aligned} \tag{5.2}$$

d_0 is the initial decoder hidden state and MLP denotes multi-layer-perceptron.

Since computing the exact marginal likelihood $\log \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} p_\theta(Y|X, \beta)$ requires enumerating over all possible combinations of β (complexity $\mathbf{O}(2^n)$), we need some way to efficiently estimate the likelihood.

5.3.2 Soft-Select

Soft-select falls back on a *deterministic* network to output the likelihood function's first-order Taylor series approximation expanded at $\mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} \beta$:

$$\begin{aligned} &\log \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} p_\theta(Y|X, \beta) \\ &\approx \log[p_\theta(Y|X, \gamma) + \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} (\beta - \gamma) p'_\theta(Y|X, \gamma)] \\ &= \log p_\theta(Y|X, \gamma) \end{aligned}$$

By moving the expectation into the decoding function, we can deterministically compute the likelihood by setting $\beta_i = \gamma_i$, reducing complexity to $\mathbf{O}(1)$. Each attention weight will first be "soft-masked" by γ before being passed to the decoder. soft-select is fully differentiable and can be easily trained by gradient descent. However, this soft-approximation is normally inaccurate, especially when $\mathbf{B}(\gamma)$ has a high entropy, which is common in one-to-many text generation tasks. The gap between $\log \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} p_\theta(Y|X, \beta)$ and $\log p_\theta(Y|X, \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)})$ will be large (Ma *et al.*, 2017; Deng *et al.*, 2018). In practice, this would lead to unrealistic generations when sampling β from the deterministically trained distribution.

5.3.3 Reinforce-Select

Reinforce-select (RS) (Ling and Rush, 2017; Chen and Bansal, 2018) utilizes reinforcement learning to approximate the marginal likelihood. Specifically, it is trained to maximize a lower bound of the likelihood by applying the Jensen inequality:

$$\log \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} p_\theta(Y|X, \beta) \geq \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} \log p_\theta(Y|X, \beta)$$

The gradient to γ is approximated with Monte-Carlo sampling by applying the REINFORCE algorithm (Williams, 1992; Glynn, 1990). To speed up convergence, we pre-train the selector by some distant supervision, which is a common practice in reinforcement learning. REINFORCE is unbiased but has a high variance. Many

research have proposed sophisticated techniques for variance reduction (Mnih and Gregor, 2014; Tucker *et al.*, 2017; Grathwohl *et al.*, 2018). In text generation, the high-variance problem is aggravated because there exists multiple valid selections. Accurately estimating the likelihood becomes difficult. Another issue is its tendency to avoid stochasticity (Raiko *et al.*, 2015), which we will show in Sec 5.5.2 that it results in low content-level diversity.

5.3.4 Variational Reinforce-Select

We propose Variational Reinforce-Select (VRS) which applies variational inference (Kingma and Welling, 2014) for variance reduction. Instead of directly integrating over $\mathbf{B}(\gamma)$, it imposes a proposal distribution q_ϕ for importance sampling. The marginal likelihood is lower bounded by:

$$\begin{aligned}
 & \log \mathbb{E}_{\beta \sim \mathbf{B}(\gamma)} p_\theta(Y|X, \beta) \\
 &= \log \mathbb{E}_{\beta \sim q_\phi} \frac{p_\theta(Y, \beta|X)}{q_\phi(\beta)} \\
 &\geq \mathbb{E}_{\beta \sim q_\phi} \log \frac{p_\theta(Y, \beta|X)}{q_\phi(\beta)} \\
 &= \mathbb{E}_{\beta \sim q_\phi} \log p_\theta(Y|X, \beta) - KL(q_\phi || \mathbf{B}(\gamma))
 \end{aligned} \tag{5.3}$$

By choosing a proper q_ϕ , the bound will be improved and the variance can be largely reduced compared with REINFORCE. If q_ϕ equals the posterior distribution $p_\theta(\beta|X, Y)$, the bound is tight and the variance would be zero (Mnih and Rezende, 2016). We define $q_\phi(\beta|X, Y)$ as a mean-field distribution parameterized by a set of global parameters ϕ to approach the true posterior distribution. ϕ , θ and γ are simultaneously trained by maximizing the last line of Eq. 5.3. $q_\phi(\beta|X, Y)$ also allows us to further perform posterior inference: Given an arbitrary text Y for a source X , we can infer which source contents are included in Y .

In Eq.5.3, the KL divergence term can be computed analytically. As for the independence assumption, it can be summed over each individual β_i . The likelihood term is differentiable to θ but not to ϕ , we estimate the gradient to ϕ in Eq 5.3 by applying the REINFORCE estimator:

$$\begin{aligned}
 & \nabla_\phi \mathbb{E}_{\beta \sim q_\phi} \log p_\theta(Y|X, \beta) = \\
 & \mathbb{E}_{\beta \sim q_\phi} \nabla_\phi \log q_\phi(\beta|X, Y) (\log p_\theta(Y|X, \beta) - B)
 \end{aligned}$$

B is the control variate (Williams, 1992). The optimal B would be (Weaver and Tao, 2001):

$$B^* = \mathbb{E}_{\beta \sim q_\phi} \log p_\theta(Y|X, \beta)$$

which we set as a soft-select approximation:

$$B = \log p_\theta(Y|X, \mathbb{E}_{\beta \sim q_\phi} \beta)$$

We estimate Eq. 5.3.4 with a single sample from q_ϕ for efficiency. Though multiple-sample could potentially further tighten the bound and reduce the variance (Burda *et al.*, 2015; Lawson *et al.*, 2018; Tucker *et al.*, 2019), it brings significant computational overhead, especially in text generation tasks where the whole sentence needs to be decoded.

5.3.5 Degree of Controllability

In practice, when treating content selection as latent variables, the model tends to end up with a trivial solution of always selecting all source tokens (Shen *et al.*, 2018a; Ke *et al.*, 2018b). This behavior is understandable since Eq. 5.2 strictly masks unselected tokens. Wrongly unselecting one token will largely deteriorate the likelihood. Under the maximum likelihood (MLE) objective, this high risk pushes the selector to take a conservative strategy of always keeping all tokens, then the whole model degenerates to the standard Enc-Dec and the selection mask loses effects on the generation. Usually people apply a penalty term to the selecting ratio when optimizing the likelihood:

$$\mathcal{L} + \lambda |(\bar{\gamma} - \alpha)| \quad (5.4)$$

\mathcal{L} is the MLE loss function, $\bar{\gamma}$ is the mean of γ and α is the target selecting ratio. This forces the selector to select the most important α tokens for each source input instead of keeping all of them.

In our VRS model, we can easily adjust the degree of controllability by limiting an upper bound of the conditional mutual information (CMI) $I(\beta, Y|X)$ (Zhao *et al.*, 2018a). Specifically, we can change our objective into:

$$\begin{aligned} \max_{\phi, \theta, \gamma} \mathbb{E}_{\beta \sim q_\phi} \log p_\theta(Y|X, \beta) \\ - \lambda |KL(q_\phi || \mathbf{B}(\gamma)) - \epsilon| \end{aligned} \quad (5.5)$$

λ is a fixed lagrangian multiplier. Eq. 5.5 can be proved equal to maximum likelihood with the constraint $I(\beta, Y|X) = \epsilon$ given proper λ (Alemi *et al.*, 2018). A higher ϵ indicates β has more influences to Y (higher controllability) while always safely selecting all tokens will lead $I(\beta, Y|X) = 0$.⁹ It is preferred over Eq. 5.4 because (a) CMI directly considers the dependency between the selection and *multiple*-possible text while limiting the ratio aims at finding the *single* most salient parts for each source. (b) Unlike CMI, limiting the ratio is coarse. It considers only the total selected size and ignores its internal distribution.

In practice, we can set ϵ to adjust the degree of controllability we want. Later we will show it leads to a trade-off with performance. The final algorithm is detailed in Algorithm 1. To keep fairness, we train RS and VRS with the same control variate and pre-training strategy.¹⁰

⁹We also tried adding a coverage constraint to ensure the decoder covers all the selected tokens (Wen *et al.*, 2015; Wang *et al.*, 2019), but we find it brings no tangible help since a higher CMI can already discourage including redundant tokens into the selection.

¹⁰The only extra parameter of VRS is ϕ which is a simple MLP structure. The actual training

Algorithm 1 Variational Reinforce-Select (VRS)

Parameters: θ, ϕ, γ
 $pretrain \leftarrow \text{TRUE}$
repeat
 Sample X, Y from the corpus;
 Encode X into (h_1, h_2, \dots, h_n) ;
 if $pretrain$ **then**
 Update ϕ with distant supervision;
 Update θ, γ by $\nabla_{\theta, \gamma} \mathbb{E}_{\beta \sim q_\phi} \log p_\theta(Y|X, \beta) - KL(q_\phi || \mathbf{B}(\gamma))$ (Eq. 5.3);
 else
 Update θ, γ, ϕ by $\nabla_{\theta, \gamma, \phi} \mathbb{E}_{\beta \sim q_\phi} \log p_\theta(Y|X, \beta) - \lambda |KL(q_\phi || \mathbf{B}(\gamma)) - \epsilon|$ (Eq. 5.5);
 end if
 $pretrain \leftarrow \text{FALSE}$ if Eq. 5.3 degrades
until convergence and $pretrain$ is False

5.4 RELATED WORK

Most content selection models train the selector with heuristic rules (Hsu *et al.*, 2018; Li *et al.*, 2018; Yu *et al.*, 2018; Gehrmann *et al.*, 2018; Yao *et al.*, 2019; Moryossef *et al.*, 2019), which fail to fully capture the relation between selection and generation. Mei *et al.* (2016); Zhou *et al.* (2017); Lin *et al.* (2018); Li *et al.* (2018) “soft-select” word or sentence embeddings based on a gating function. The output score from the gate is a deterministic vector without any probabilistic variations, so controlling the selection to generate diverse text is impossible. Very few works explicitly define a bernoulli distribution for the selector, then train with the REINFORCE algorithm (Ling and Rush, 2017; Chen and Bansal, 2018), but the selection targets at a high recall regardless of the low precision, so the controllability over generated text is weak. Fan *et al.* (2018a) control the generation by manually concatenating entity embeddings, while our model is much more flexible by explicitly defining the selection probability over all source tokens.

Our work is closely related with learning discrete representations with variational inference (Wen *et al.*, 2017; van den Oord *et al.*, 2017; Kaiser *et al.*, 2018; Lawson *et al.*, 2018), where we treat content selection as the latent representation. Limiting the KL-term is a common technique to deal with the “posterior collapse” problem (Kingma *et al.*, 2016a; Yang *et al.*, 2017b; Shen *et al.*, 2018c). We adopt a similar approach and use it to further control the selecting strategy.

5.5 EXPERIMENTS

For the experiments, we focus on comparing (1) Bottom-up generation (Bo.Up.), (2) soft-select (SS), (3) Reinforce-select (RS) and (4) Variational-Reinforce-select (VRS)

complexity is similar to RS because they both use the REINFORCE algorithm for gradient estimation.

Gigaword	Oracle upper bound of			% unique Generation	% unique Mask	% Effect of Selector	Entropy of Selector
	ROUGE-1	ROUGE-2	ROUGE-L				
Bo.Up.	42.61	22.32	38.37	84.28	95.87	87.91	0.360
SS	33.15	14.63	30.68	82.06	96.23	85.27	0.392
RS	36.62	18.34	34.60	3.01	6.23	48.31	0.018
VRS	54.73	33.28	51.62	89.23	92.51	96.45	0.288
Wikibio	ROUGE-4	BLEU-4	NIST	Generation	Mask	Selector	Selector
Bo.Up.	47.28	49.95	11.06	31.57	77.42	40.78	0.177
SS	41.73	43.94	9.82	63.09	89.42	70.55	0.355
RS	44.07	46.89	10.31	4.55	43.83	10.38	0.105
VRS	52.41	55.03	11.89	57.62	77.83	74.03	0.181

Table 5.2: Diversity of content selection. The % effect of selector is defined as the ratio of unique generation and mask, which reflects the rate that changing the selector will lead to corresponding changes of the generated text.

regarding their performance on content selection. SS and RS are trained with the selecting ratio constraint in Eq. 5.4. For the SS model, we further add a regularization term to encourage the maximum value of γ to be close to 1 as in Mei *et al.* (2016). We first briefly introduce the tasks and important setup, then present the evaluation results.

5.5.1 Tasks and Setup

We test content-selection models on the headline and data-to-text generation task. Both tasks share the same framework with the only difference of source-side encoders.

Headline Generation: We use English Gigaword preprocessed by Rush *et al.* (2015), which pairs first sentences of news articles with their headlines. We keep most settings same as in Zhou *et al.* (2017), but use a vocabulary built by byte-pair-encoding (Sennrich *et al.*, 2016b). We find it speeds up training with superior performance.

Data-to-Text Generation: We use the Wikibio dataset (Lebret *et al.*, 2016). The source is a Wikipedia infobox and the target is a one-sentence biography description. Most settings are the same as in Liu *et al.* (2018), but we use a bi-LSTM encoder for better performance.

Heuristically extracted content: This is used to train the selector for bottom up models and pre-train the RS and VRS model. For wikibio, we simply extract overlapped words between the source and target. In Gigaword, as the headline is more abstractive, we select the closest source word for each target word in the embedding space. Stop words and punctuations are prohibited from being selected.

Choice of α/ϵ : As seen in Sec 5.3.5, we need to set the hyperparameter α for RS/SS and ϵ for VRS. α corresponds to the selecting ratio. We set them as $\alpha = 0.35$ for Wikibio and 0.25 for Gigaword. The value is decided by running a human evaluation to get the empirical estimation. To keep comparison fairness, we tune ϵ to

make VRS select similar amount of tokens with RS. The values we get are $\epsilon = 0.15n$ for Wikibio and $\epsilon = 0.25n$ for Gigaword. n is the number of source tokens.¹¹

5.5.2 Results and Analysis

Ideally we would expect the learned content selector to (1) have reasonable diversity so that text with various contents can be easily sampled, (2) properly control the contents described in the generated text and (3) not hurt performance. The following section will evaluate these three points in order.

Diversity: We first look into the diversity of content selection learned by different models. For each test data, 50 selection masks are randomly sampled from the model’s learned distribution. Greedy decoding is run to generate the text for each mask. We measure the entropy of the selector, proportion of unique selection masks and generated text in the 50 samples. We further define the “effect” of the selector as the ratio of sampled unique text and mask. This indicates how often changing the selection mask will also lead to a change in the generated text. The results are averaged over all test data. Following Rush *et al.* (2015) and Lebrete *et al.* (2016), we measure the quality of generated text with ROUGE-1, 2, L F-score for Gigaword and ROUGE-4, BLEU-4, NIST for Wikibio. As there is only one reference text for each source, we report an oracle upper bound of these scores by assuming an “oracle” that can choose the best text among all the candidates (Mao *et al.*, 2015; Wang *et al.*, 2017c). Namely, out of each 50 sampled text, we pick the one with the maximum metric score. The final metric score is evaluated on these “oracle” picked samples. The intuition is that if the content selector is properly trained, at least one out of the 50 samples should describe similar contents with the reference text, the metric score between it and the reference text should be high. Table 5.2 lists the results. We can have the following observations:

- RS model completely fails to capture the content-level diversity. Its selector is largely deterministic, with a lowest entropy value among all models. In contrast, the selector from SS, VRS and Bo.Up. have reasonable diversity, with over 90% and 75% unique selection masks for Gigaword and Wikibio respectively.
- The selector from VRS has the strongest effect to the generator, especially on the Gigaword data where modifying the content selection changes the corresponding text in more than 95% of the cases. RS has the lowest effect value, which indicates that even with the selecting ratio constraint, its generator still ignores the selection mask to a large extent.
- The oracle metric score of VRS is much higher than the other two. This is beneficial when people want to apply the model to generate a few candidate text then hand-pick the suitable one. VRS has more potential than the other

¹¹ ϵ corresponds to the KL divergence of the selection mask, which scales linearly with the number of source tokens, so we set it proportionally w.r.t. n .

three to contain the expected text. SS performs worst. The gap between the soft approximation and the real distribution, as mentioned before, indeed results in a large drop of performance.

In short, compared with others, the content selector of VRS is (1) *diverse*, (2) *has stronger effect on the text generation* and (3) *with a larger potential of producing an expected text*.

Controllability: We have shown the content selector of VRS is diverse and has strong effect on the text generation. This section aims at examining whether such effect is desirable, i.e., whether the selector is able to properly control the contents described in the text. We measure it based on the self-bleu metric and a human evaluation.

The self-bleu metric measures the controllability by evaluating the “intra-selection” similarity of generated text. Intuitively, by fixing the selection mask, multiple text sampled from the decoder are expected to describe the same contents and thereby should be highly similar to each other. The decoder should only model surface-level diversity without further modifying the selected contents. With this in mind, for each test data, we randomly sample a selection mask from the selector’s distribution, then fix the mask and run the decoder to sample 10 different text. The self-BLEU-1 score (Zhu *et al.*, 2018) on the sampled text is reported, which is the average BLEU score between each text pair. A higher self-BLEU score indicates the sampled text are more similar with each other. The results are shown in Table 5.3. We can see generations from VRS have a clearly higher intra-selection similarity. SS performs even worse than RS, despite having a high effect score in Table 5.2. The selector from SS affects the generation in an undesirable way, which also explain why SS has a lowest oracle metric score though with a high score on content diversity and effect.

Method	Bo.Up.	SS	RS	VRS
Gigaword	46.58	37.20	48.13	61.14
Wikibio	38.30	13.92	25.99	43.81

Table 5.3: Self-Bleu score by fixing selection mask. Higher means better controllability of content selection

We further run a human evaluation to measure the text-content consistency among different models. 100 source text are randomly sampled from the human-written DUC 2004 data for task 1&2 (Over *et al.*, 2007). Bo.Up, SS, RS and VRS are applied to generate the target text by first sampling a selection mask, then run beam search decoding with beam size 10. We are interested in seeing (1) if multiple generations from the same selection mask are paraphrases to each other (intra-consistent) and (2) if generations from different selection masks do differ in the content they described (inter-diverse). The results in Table 5.4 show that VRS significantly outperforms the other two in both intra-consistency and inter-diversity. RS has the lowest score on both because the selector has very weak effects on the

Method	Fluency	intra-consistency	inter-diversity
Reference	0.96	-	-
Enc-Dec	0.83	-	-
Bo.Up.	0.46	0.48	0.61
SS	0.27	0.41	0.54
RS	0.78	0.39	0.47
VRS	0.74	0.72	0.87

Table 5.4: Human evaluation on fluency, intra-consistency and inter-diversity of content selection on DUC 2004.

generation as measured in the last section. Bo.Up and SS lay between them. Overall VRS is able to *maintain the highest content-text consistency* among them.

Method	R-1	R-2	R-L	%Word
Zhou <i>et al.</i> (2017)	36.15	17.54	33.63	100
Enc-Dec	35.92	17.43	33.42	100
SS	20.35	4.78	16.53	24.82
Bo.Up	28.17	10.32	26.68	24.54
RS	35.45	16.38	32.71	25.12
VRS($\epsilon = 0$)-pri	36.42	17.81	33.86	78.63
VRS($\epsilon = 0.25$)-pri	34.26	15.11	31.69	24.36
VRS($\epsilon = 0$)-post	37.14	18.03	34.26	78.66
VRS($\epsilon = 0.25$)-post	56.72	33.24	51.88	24.53

Table 5.5: Gigaword best-select results. Larger ϵ leads to more controllable selector with a bit degrade of performance. (-post) means selecting from the posterior $q_\phi(\beta|X, Y)$, (-pri) is from the prior $\mathbf{B}(\gamma_i)$.

Performance & Trade-off: To see if the selector affects performance, we also ask human annotators to judge the text fluency. The fluency score is computed as the average number of text being judged as fluent. We include generations from the standard Enc-Dec model. Table 5.4 shows the best fluency is achieved for Enc-Dec. Imposing a content selector always affects the fluency a bit. The main reason is that when the controllability is strong, the change of selection will directly affect the text realization so that a tiny error of content selection might lead to unrealistic text. If the selector is not perfectly trained, the fluency will inevitably be influenced. When the controllability is weaker, like in RS, the fluency is more stable because it will not be affected much by the selection mask. For SS and Bo.Up, the drop of fluency is significant because of the gap of soft approximation and the independent training procedure. In general, VRS does properly decouple content selection from the enc-dec architecture, with only tiny degrade on the fluency.

Method	R-4	B-4	NIST	%Word
Liu <i>et al.</i> (2018)	41.65	44.71		100
Enc-Dec	42.07	44.80	9.82	100
SS	5.10	5.73	0.24	35.12
Bo.Up	8.07	9.52	0.42	38.79
RS	42.64	45.08	10.01	34.53
VRS($\epsilon = 0$)-pri	43.01	46.01	10.24	84.56
VRS($\epsilon = 0.15$)-pri	42.13	44.51	9.84	34.04
VRS($\epsilon = 0$)-post	43.84	46.60	10.27	85.34
VRS($\epsilon = 0.15$)-post	49.68	52.26	11.48	34.57

Table 5.6: Wikibio best-select results.

Table 5.5/5.6 further measure the metric scores on Gigaword/Wikibio by decoding text from the best selection mask based on the selector’s distribution (set $\beta_i = 1$ if $\mathbf{B}(\gamma_i) > 0.5$ and 0 otherwise). We include results from VRS model with $\epsilon = 0$, which puts no constraint on the mutual information. We further report the score by generating the best selection mask from the learned posterior distribution $q_\phi(\beta|X, Y)$ for VRS model. Two current SOTA results from Zhou *et al.* (2017) and Liu *et al.* (2018) and the proportion of selected source words for each model are also included. We have the following observations:

- As the value of ϵ decreases, the performance of VRS improves, but the selector loses more controllability because the model tends to over-select contents (over 75% source words selected). The text-content consistency will become low.
- Increasing ϵ sacrifices a bit performance, but still comparable with SOTA. Especially on Wikibio where the performance drop is minor. The reason should be that Wikibio is relatively easier to predict the selection but Gigaword has more uncertainty.
- Increasing ϵ improves the accuracy of the posterior selection. This would be useful when we want to perform posterior inference for some source-target pair.
- Setting $\epsilon = 0$ can actually outperform SOTA seq2seq which keeps all tokens, suggesting it is still beneficial to use the VRS model even if we do not care about the controllability.

Figure 5.2 visualizes how changing the value of ϵ affects the negative log likelihood (NLL), entropy of the selector and self-bleu score, which roughly correlates with performance, diversity and controllability. NLL is evaluated based on the lower bound in Eq 5.3 (Sohn *et al.*, 2015). We can see as ϵ increases, the performance decreases gradually but the content selection gains more diversity and controllability. In practice we can tune the ϵ value to achieve a trade-off.

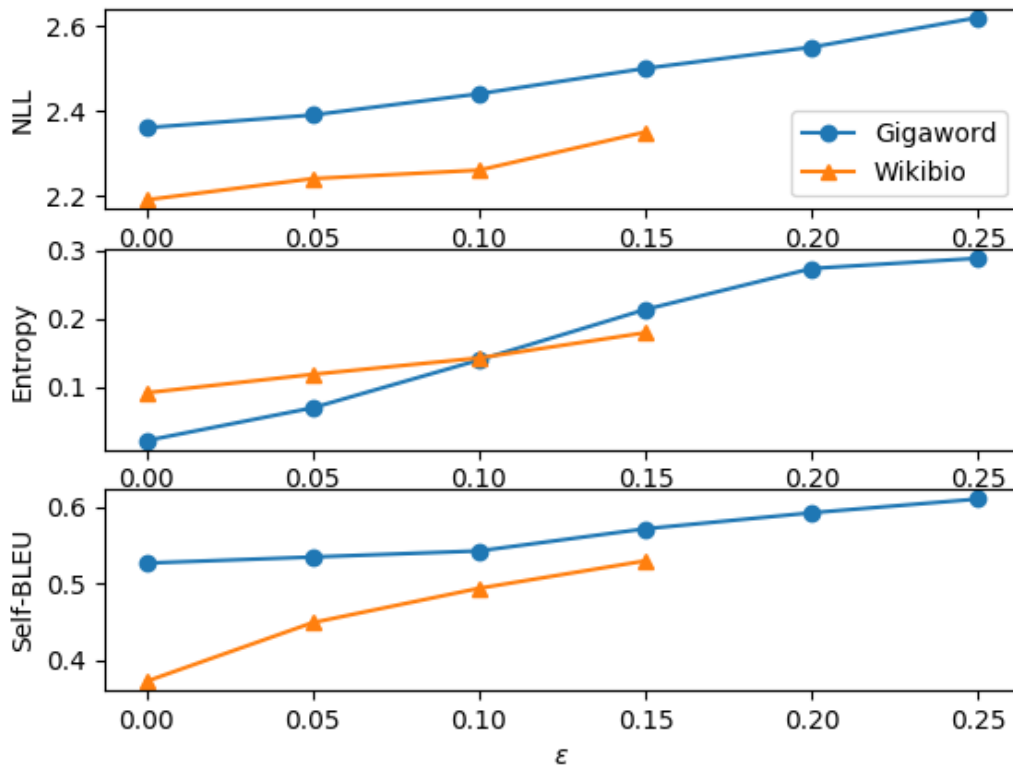


Figure 5.2: Negative log likelihood (NLL), selection entropy and self-BLEU as ϵ changes. NLL and self-bleu on Wikibio are added by 1 for better visualization. Lower NLL suggests higher performance. Higher entropy/self-BLEU means higher diversity/controllability.

Generation Example: Figure 5.3 shows some examples from Gigaword. As can be seen, decodings from the VRS model are largely consistent with each other, in most cases only replacing one or two words with corresponding synonyms. Samples are able to faithfully convey all selected contents. In contrast, generations from SS, Bo.Up. and RS are unpredictable, differing in both selected contents and also the way of saying. SS and Bo.Up also suffer more from the text disfluency. The generations from them are largely uncertain.

Source: indian prime minister p.v. narasimha rao 's promise of more autonomy for troubled kashmir and his plea for early state elections has sparked a violent reaction from provincial moslem and opposition parties .

Samples from SS:

- t1: indian indian calls for end to violence in kashmir .
- t2: indian pm calls for end to violence in afghanistan .
- t3: indian pm calls for boycott of pakistan 's ruling party .

Samples from Bo.Up:

- t1: india promises more autonomous more autonomy .
- t2: indian pm promises autonomy for kashmir autonomy .
- t3: indian pm 's promise sparks violent reaction .

Samples from RS:

- t1: indian pm 's kashmir promises sparks violent reaction.
- t2: indian pm 's promise sparks violent reaction .
- t3: kashmir parties blast pm 's promise .

Samples from VRS:

- t1: indian pm 's promise on kashmir sparks uproar .
- t2: indian pm 's promise on kashmir sparks protests .
- t3: indian pm 's promise for kashmir sparks controversy .

Source: factory orders for manufactured goods rose ## percent in september , the commerce department said here thursday .

Samples from SS:

- t1: u.s. consumer confidence down in january in january.
- t2: u.s. wholesale prices up ## percent in october .
- t3: u.s. jobless rate rises to ## percent in march .

Samples from Bo.Up.:

- t1: september u.s. factory orders up ## percent .
- t2: september u.s. factory orders increase .
- t3: factory orders up in september .

Samples from RS:

- t1: u.s. factory orders up ## percent in september .
- t2: factory orders for manufactured goods rise .
- t3: factory orders up in september from the year .

Samples from VRS:

- t1: september factory orders up ## percent .
- t2: september factory orders rise ## percent .
- t3: september factory orders increase ## pct .

Figure 5.3: Text generation examples from Gigaword. Highlighted words are selected. t1-3 are sampled from the decoder based on the selected content. Generations from VRS are more faithful to selected contents.

5.6 CONCLUSION

In this chapter, we tackle the unaddressed problem of controllable content selection in text generation. We propose a general framework based on variational inference that can be potentially applied to arbitrary tasks. On both the headline generation and data-to-text tasks, our model outperforms state-of-the-art models regarding the diversity and controllability of content selection. We further introduce an effective way to achieve a performance/controllability trade-off, which can be easily tuned to meet specific requirements. The proposed model can be further improved by, for example, using a more flexible content selector or multi-sample estimation, which we leave for future work.

THE last chapter uses latent variables to model the content selection. This chapter goes one step further by modeling the word-level alignment between the input and the output, so that humans can easily interpret how each word is generated. We base our system with the pointer Generator, which has been the de facto standard for modern summarization systems. However, this architecture faces two major drawbacks: Firstly, the pointer is limited to copying the exact words while ignoring possible inflections or abstractions, which restricts its power of capturing richer latent alignment. Secondly, the copy mechanism results in a strong bias towards extractive generations, where most sentences are produced by simply copying from the source text. In this chapter, we address these problems by allowing the model to “edit” pointed tokens instead of always hard copying them. The editing is performed by transforming the pointed word vector into a target space with a learned relation embedding. On three large-scale summarization dataset, we show the model is able to (1) capture more latent alignment relations than exact word matches, (2) improve word alignment accuracy, allowing for better model interpretation and controlling, (3) generate higher-quality summaries validated by both qualitative and quantitative evaluations and (4) bring more abstraction to the generated summaries (Shen *et al.*, 2019c).¹²

6.1 INTRODUCTION

Modern state-of-the-art (SOTA) summarization models are built upon the pointer generator architecture (See *et al.*, 2017). At each decoding step, the model generates a sentinel to decide whether to sample words based on the neural attention (generation mode), or directly copy from an aligned source context (point mode) (Gu *et al.*, 2016; Merity *et al.*, 2017; Yang *et al.*, 2017a). Though outperforming the vanilla attention models, the pointer generator only captures exact word matches. As shown in Fig. 6.1, for abstractive summarization, there exists a large number of syntactic inflections (escalated \rightarrow escalates) or semantic transformations (military campaign \rightarrow war), where the target word also has an explicit grounding in the source context but changes its surface. In standard pointer generators, these words are not covered by the point mode. This largely restricts the application of the pointer generator, especially on highly abstractive dataset where only a few words are exactly copied. Moreover, the hard copy operation biases the model towards extractive summarization, which is undesirable for generating more human-like

¹²The source code is available at <https://github.com/chin-gyou/generalized-PG>.

Source: The **sri lankan** government on Wednesday announced the **closure** of government **schools** with immediate effect **as** a **military campaign** against tamil separatists **escalated** in the north of the country.

... on Wednesday announced the closure of schools ...

Sri lanka **closes** schools as war escalates.

Aligned to “announced”:

Sri lanka announces closure of government schools.

Sri lanka declares closure of government schools.

Blue: prior
Red: posterior

Aligned to “closure”:

Sri Lanka closes government schools.

Sri lanka shuts down government schools

Figure 6.1: Alignment visualization of our model when decoding “closes”. Posterior alignment is more accurate for model interpretation. In contrast, the prior alignment probability is spared to “announced” and “closure”, which can be manually controlled to generate desired summaries. Decoded samples are shown when aligned to “announced” and “closure” respectively. **Highlighted source words** are those that can be directly aligned to a target token in the gold summary.

summaries (Kryściński *et al.*, 2018).

To solve this problem, we propose **Generalized Pointer Generator (GPG)** which replaces the hard copy component with a more general soft “editing” function. We do this by learning a relation embedding to transform the pointed word into a target embedding. For example, when decoding “closes” in Figure 6.1, the model should first point to “closure” in the source, predict a relation to be applied (noun → third person singular verb), then transform “closure” into “closes” by applying the relation transformation. The generalized point mode is encouraged to capture such latent alignment which cannot be identified by the standard pointer generator.

This improved alignment modelling is intriguing in that (a) people can better control the generation by manipulating the alignment trajectory, (b) posterior alignment can be inferred by Bayes’ theorem (Deng *et al.*, 2018; Shankar and Sarawagi, 2019) to provide a better tool for interpretation¹³ and finally (c) explicitly capturing the

¹³The induced alignment offers useful annotations for people to identify the source correspondence

alignment relation should improve generation performance. (Figure 6.1 shows an example of how latent alignment can improve the controllability and interpretation. Pointer generators fail to model such alignment relations that are not exact copies.) To eliminate the OOV problem, we utilize the byte-pair-encoding (BPE) segmentation (Sennrich *et al.*, 2016b) to split rare words into sub-units, which has very few applications in summarization so far (Fan *et al.*, 2018b; Kiyono *et al.*, 2018), though being a common technique in machine translation (Wu *et al.*, 2016; Vaswani *et al.*, 2017; Gehring *et al.*, 2017).

Our experiments are conducted on three summarization datasets: CNN/dailymail (Hermann *et al.*, 2015), English Gigaword (Rush *et al.*, 2015) and XSum (Narayan *et al.*, 2018) (a newly collected corpus for extreme summarization). We further perform human evaluation and examine the word alignment accuracy on the manually annotated DUC 2004 dataset. Overall we find our model provides the following benefits:

1. It can capture richer latent alignment and improve the word alignment accuracy, enabling better controllability and interpretation.
2. The generated summaries are more faithful to the source context because of the explicit alignment grounding.
3. It improves the abstraction of generations because our model allows editing the pointed token instead of always copying exactly.

In the next section, we will first go over the background, then introduce our model and finally present the experiment results and conclusion.

6.2 BACKGROUND

Let X, Y denote a source-target pair where X corresponds to a sequence of words x_1, x_2, \dots, x_n and Y is its corresponding summary y_1, y_2, \dots, y_m . In this section, we introduce two baseline models for automatically generating Y from X :

6.2.1 Seq2seq with Attention

In a seq2seq model, each source token x_i is encoded into a vector h_i . At each decoding step t , the decoder computes an attention distribution a_t over the encoded vectors based on the current hidden state d_t (Bahdanau *et al.*, 2015):

$$a_t = \text{softmax}(f(h_i, d_t)) \quad (6.1)$$

for each target word. News editors can post-edit machine-generated summaries more efficiently with such annotation. For summary readers, it also helps them track back the source context when they are interested in some specific details.

f is a score function to measure the similarity between h_i and d_t . The context vector c_t and the probability of next token are computed as below.

$$\begin{aligned} c_t &= \sum_i h_i a_{t,i} \\ y_t^* &= [d_t \circ c_t]L \\ p_{vocab} &= \text{softmax}(y_t^* W^T) \end{aligned} \tag{6.2}$$

\circ means concatenation and L, W are trainable parameters. We tie the parameters of W and the word embedding matrix as in Press and Wolf (2017); Inan *et al.* (2017). Namely, a target vector y_t^* is predicted, words having a higher inner product with y_t^* will have a higher probability.

6.2.2 Pointer Generator

The pointer generator extends the seq2seq model to support copying source words (Vinyals *et al.*, 2015). At each time step t , the model first computes a generation probability $p_{gen} \in [0, 1]$ by:

$$p_{gen} = \sigma(\text{MLP}_g([d_t \circ c_t])) \tag{6.3}$$

σ is a sigmoid function and MLP_g is a learnable multi-layer perceptron. p_{gen} is the probability of enabling the generation mode instead of the point mode. In the generation mode, the model computes the probability over the whole vocabulary as in Eq. 6.2. In the point mode, the model computes which source word to copy based on the attention distribution a_t from Eq.6.1. The final probability is marginalized over $a_{t,i}$:

$$\begin{aligned} p(y_t) &= p_{gen} p_{vocab}(y_t) + (1 - p_{gen}) \sum_i a_{t,i} \delta(y_t | x_i) \\ \delta(y_t | x_i) &= \begin{cases} 1, & \text{if } y_t = x_i. \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6.4}$$

If we know exactly from which mode each word comes from, *e.g.*, by assuming all co-occurred words are copied, then the marginalization can be omitted (Gulcehre *et al.*, 2016; Wiseman *et al.*, 2017), but normally p_{gen} is treated as a latent variable (Gu *et al.*, 2016; See *et al.*, 2017).

6.3 GENERALIZED POINTER GENERATOR (GPG)

As seen in Eq .6.4, $\delta(y_t | x_i)$ is a 0-1 event that is only turned on when y_t is exactly the same word as x_i . This restricts the expressiveness of the point mode, preventing it from paying attention to inflections, POS transitions or paraphrases. This section explains how we generalize pointer networks to cover these conditions.

Redefine $\delta(y_t | x_i)$: We extend $\delta(y_t | x_i)$ by defining it as a smooth probability distribution over the whole vocabulary. It allows the pointer to edit x_i to a different

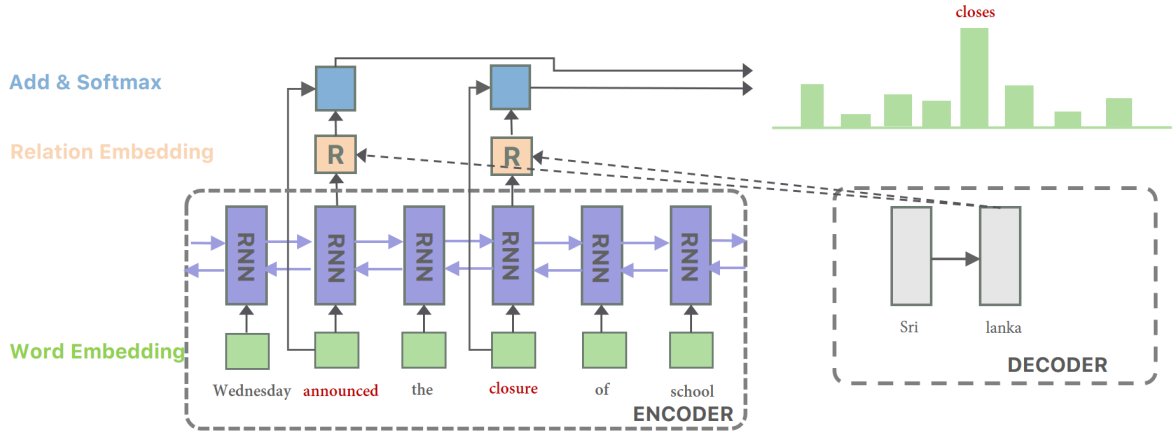


Figure 6.2: Architecture of the generalized pointer. The same encoder is applied to encode the source and target. When decoding “closes”, we first find top-k source positions with the most similar encoded state. For each position, the decoding probability is computed by adding its word embedding and a predicted relation embedding.

word y_t instead of simply copying it. Following Eq. 6.2, we derive $\delta(y_t|x_i)$ by first predicting a target embedding $y_{t,i}^*$, then applying the softmax. The difference is that we derive $y_{t,i}^*$ as the summation of the pointed word embedding \vec{x}_i and a relation embedding $r(d_t, h_t)$:

$$\begin{aligned} y_{t,i}^* &= r(d_t, h_t) + \vec{x}_i \\ \delta(y_t|x_i) &= \text{softmax}(y_{t,i}^* W^T) \end{aligned} \quad (6.5)$$

\vec{x}_i denotes the embedding of the word x_i . $r(d_t, h_t)$ can be any function conditioning on d_t and h_t , which we parameterize with a multi-layer-perceptron in our experiments. The computation of $y_{t,i}^*$ is similar to the classical TransE model (Bordes *et al.*, 2013) where an entity vector is added by a relation embedding to translate into the target entity. The intuition is straightforward: After pointing to x_i , humans usually first decide which relation should be applied (inflection, hypernym, synonym, etc) based on the context $[d_t, h_t]$, then transform x_i to the proper target word y_t . Using addition transformation is backed by the observation that vector differences often reflect meaningful word analogies (Mikolov *et al.*, 2013b; Pennington *et al.*, 2014) (“man” – “king” \approx “woman” – “queen”) and they are effective at encoding a great amount of word relations like hypernym, meronym and morphological changes (Vylomova *et al.*, 2016; Hakami *et al.*, 2018; Allen and Hospedales, 2019). These word relations reflect most alignment conditions in text summarization. For example, humans often change the source word to its hypernym (boy \rightarrow child), to make it more specific (person \rightarrow man) or apply morphological transformations (liked \rightarrow like). Therefore, we assume $\delta(y_t|x_i)$ can be well modelled by first predicting a relation embedding to be applied, then added to \vec{x}_i . If x_i should be exactly copied like in standard pointer generators, the relation embedding is a zero vector

meaning an identity transition. We also tried applying more complex transitions to \vec{x}_i like diagonal mapping (Trouillon *et al.*, 2016), but did not observe improvements. Another option is to estimate $\delta(y_t|x_i)$ directly from (d_t, h_i) by an MLP regardless of \vec{x}_i . However, this leads to poor alignment and performance drop because y_t is not explicitly grounded on x_i ¹⁴.

Estimate Marginal Likelihood: Putting Eq. 6.5 back to Eq. 6.4, the exact marginal likelihood is too expensive for training. The complexity grows linearly with the source text length n and each computation of $\delta(y_t|x_i)$ requires a separate softmax operation. One option is to approximate it by sampling like in hard attention models (Xu *et al.*, 2015; Deng *et al.*, 2017), but the training becomes challenging due to the non-differentiable sampling process. In our work, we take an alternative strategy of marginalizing only over k most likely aligned source words. This top- k approximation is widely adopted when the target distribution is expected to be sparse and only a few modes dominate (Britz *et al.*, 2017; Ke *et al.*, 2018a; Shankar *et al.*, 2018). We believe this is a valid assumption in text summarization since most source tokens have a vanishingly small probability to be transferred into a target word.

For each target word, how to determine the k most likely aligned source words is crucial. An ideal system should always include the gold aligned source word in the top- k selections. We tried several methods and find the best performance is achieved when encoding each source/target token into a vector, then choosing the k source words that are closest to the target word in the encoded vector space. The closeness is measured by the vector inner product¹⁵. The encoded vector space serves like a contextualized word embedding (McCann *et al.*, 2017; Peters *et al.*, 2018). Intuitively if a target word can be aligned to a source word, they should have similar semantic meaning and surrounding context thus should have similar contextualized word embeddings. The new objective is then defined as in Eq. 6.6:

$$\begin{aligned}
 p(y_t) &= p_{gen}p_{vocab}(y_t) + (1 - p_{gen})p_{point}(y_t) \\
 p_{point}(y_t) &= \sum_i a_{t,i}\delta(y_t|x_i) \\
 &\approx \sum_{i; h_i^T e(y_t) \in \text{TopK}} a_{t,i}\delta(y_t|x_i)
 \end{aligned} \tag{6.6}$$

$e(y_t)$ is the encoded vector for y_t . The marginalization is performed only over the k chosen source words. Eq. 6.6 is a lower bound of the data likelihood because it only marginalizes over a subset of X . In general a larger k can tighten the bound to get a more accurate estimation and we analyze the effect of k in Section 6.5.2. Note that the only extra parameters introduced by our model are the multi-layer-perceptron to

¹⁴ h_i can contain context information from surrounding words and thus not necessarily relates to word x_i . It is part of the reason that neural attention has a poor alignment (Koehn and Knowles, 2017). Making it grounded on x_i improves alignment and performance is also observed in machine translation (Nguyen and Chiang, 2018; Kuang *et al.*, 2018)

¹⁵We compared several strategies for choosing the top- k words. Note that the top- k approximation is only used for training, so we can spot the whole target text to decide top- k candidates.

compute the relation embedding $r(d_t, h_i)$. The marginalization in Eq. 6.6 can also be efficiently parallelized. An illustration of the generalized pointer is in Figure 6.2.

6.4 RELATED WORK

Neural attention models (Bahdanau *et al.*, 2015) with the seq2seq architecture (Sutskever *et al.*, 2014) have achieved impressive results in text summarization tasks. However, the attention vector comes from a weighted sum of source information and does not model the source-target alignment in a probabilistic sense. This makes it difficult to interpret or control model generations through the attention mechanism. In practice, people do find the attention vector is often blurred and suffers from poor alignment (Koehn and Knowles, 2017; Kiyono *et al.*, 2018; Jain and Wallace, 2019). Hard alignment models, on the other hand, explicitly models the alignment relation between each source-target pair. Though theoretically sound, hard alignment models are hard to train. Exact marginalization is only feasible for data with limited length (Yu *et al.*, 2016; Aharoni and Goldberg, 2017; Deng *et al.*, 2018; Backes *et al.*, 2018), or by assuming a simple copy generation process (Vinyals *et al.*, 2015; Gu *et al.*, 2016; See *et al.*, 2017). Our model can be viewed as a combination of soft attention and hard alignment, where a simple top-k approximation is used to train the alignment part (Shankar *et al.*, 2018; Shankar and Sarawagi, 2019). The hard alignment generation probability is designed as a relation summation operation to better fit the summarization task. In this way, the generalized copy mode acts as a hard alignment component to capture the direct word-to-word transitions. On the contrary, the generation mode is a standard soft-attention structure to only model words that are purely functional, or need fusion, high-level inference and can be hardly aligned to any specific source context (Daumé III and Marcu, 2005).

6.5 EXPERIMENTS AND RESULTS

In the experiment, we compare seq2seq with attention, standard pointer generators and the proposed generalized pointer generator (GPG). To further analyze the effect of the generalized pointer, we implement a GPG model with only the point mode (GPG-pt) for comparison. We first introduce the general setup, then report the evaluation results and analysis.

6.5.1 General Setup

Dataset: We perform experiments on the CNN/dailymail (Hermann *et al.*, 2015), English Gigaword (Rush *et al.*, 2015) and XSum (Narayan *et al.*, 2018) dataset. CNN/DM contains online news with multi-sentence summaries (We use the non-anonymized version from See *et al.* (2017)). English Gigaword paired the first sentence of news articles with its headline. XSum corpus provides a single-sentence summary for each

BBC long story. We pick these three dataset as they have different properties for us to compare models. CNN/DM strongly favors extractive summarization (Kryściński *et al.*, 2018). Gigaword has more one-to-one word direct mapping (with simple paraphrasing) (Napoles *et al.*, 2012) while XSum needs to perform more information fusion and inference since the source is much longer than the target (Narayan *et al.*, 2018).

Model: We use single-layer bi-LSTM encoders for all models. For comparison, hidden layer dimensions are set the same as in Zhou *et al.* (2017) for Gigaword and See *et al.* (2017) for CNN/DM and XSum. We train with batch size 256 for gigaword and 32 for the other two. The vocabulary size is set to 30k for all dataset. Word representations are shared between the encoder and decoder. We tokenize words with WordPiece segmentation (Wu *et al.*, 2016) to eliminate the OOV problem.

Inference: We decode text using beam search (Graves, 2012) with beam size 10. We apply length normalization to rescale the score. Unlike See *et al.* (2017); Gehrmann *et al.* (2018), we do not explicitly impose coverage penalty since it brings extra hyper-parameters. Instead, for CNN/Dailymail, we use a simple tri-gram penalty (Paulus *et al.*, 2018) to prevent repeated generations. GPG models use an exact marginalization for testing and decoding, while for training and validation we use the top- k approximation mentioned above. The decoder will first decode sub-word ids then map them back to the normal sentence. All scores are reported on the word level and thus comparable with previous results. When computing scores for multi-sentence summaries. The generations are split into sentences with the NLTK sentence tokenizer.

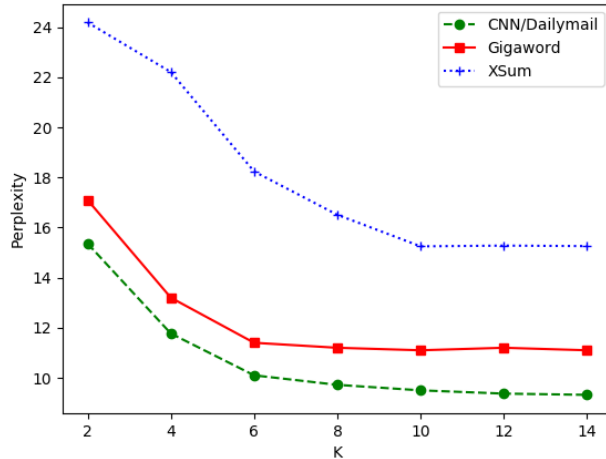
6.5.2 Results and Analysis

The results are presented in the following order: We first study the effect of the hyperparameter k , then evaluate model generations by automatic metrics and look into the generation’s level of abstraction. Finally, we report the human evaluation and word alignment accuracy.

Effect of K : k is the only hyperparameter introduced by our model. Figure 6.3 visualizes the effect of k on the test perplexity. As mentioned in Sec 6.3, a larger k is expected to tighten the estimation bound and improve the performance. The figure shows the perplexity generally decreases as increasing k . The effect on Gigaword and XSum saturates at $k = 6$ and 10 respectively, so we fix such k value for later experiments. For the longer dataset CNN/Dailymail, the perplexity might still decrease a bit afterwards, but the improvement is marginal, so we set $k = 14$ for the memory limit.

Automatic Evaluation: The accuracy is evaluated based on the standard metric ROUGE (Lin, 2004) and the word perplexity on the test data. We report the ROUGE-1, ROUGE-2 and ROUGE-L F-score measured by the official script. Table 6.1, 6.2 and 6.3 lists the results for CNN/Dailymail, Gigaword and XSum respectively. Statistically significant results are underlined¹⁶. On the top two rows of each table,

¹⁶Results on the Gigaword test set is not significant due to the small test size (1951 article-summary

Figure 6.3: Test perplexity when increasing k

Method	R-1	R-2	R-L	PPL
Point.Gen.+Cov*	39.53	17.28	36.38	
Bottom Up [†]	41.22	18.68	38.34	
seq2seq	39.79	17.37	36.34	17.49
Point.Gen.	40.03	17.52	36.77	12.36
GPG-ptr	<u>40.54</u>	18.05	37.19	<u>10.23</u>
GPG	40.95	<u>18.01</u>	37.46	9.37

Table 6.1: ROUGE score on CNN/Dailymail. * marks results from See *et al.* (2017), and [†] from Gehrmann *et al.* (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$.

we include two results taken from current state-of-the-art word-based models. They are incomparable with our model because of the different vocabulary, training and decoding process, but we report them for completeness. Lower rows are results from our implemented models. Pointer generators bring only slight improvements over the seq2seq baseline. This suggests that after eliminating the OOV problem, the naive seq2seq with attention model can already implicitly learn most copy operations by itself. GPG models outperform seq2seq and pointer generators on all dataset. The improvement is more significant for more abstractive corpus Gigaword and XSum, indicating our model is effective at identifying more latent alignment relations.

Notably, even the pure pointer model (GPG-ptr) without the generation mode outperforms standard pointer generators in CNN/DM and Gigaword, implying most target tokens can be generated by aligning to a specific source word. The finding is consistent with previous research claiming CNN/DM summaries are

pairs).

Method	R-1	R-2	R-L	PPL
seq2seq*	34.04	15.95	31.68	
DRGD [†]	36.27	17.57	33.62	
seq2seq	36.01	17.52	33.60	18.92
Point.Gen.	36.14	17.68	33.56	14.90
GPG-ptr	37.14	19.05	34.67	12.32
GPG	37.23	19.02	34.66	11.41

Table 6.2: ROUGE score on Gigaword. * marks results from the word-based seq2seq implementation of Zhou *et al.* (2017), and [†] from Li *et al.* (2017b).

Method	R-1	R-2	R-L	PPL
Point.Gen*	29.70	9.21	23.24	
T-CONVS2S*	31.89	11.54	25.75	
seq2seq	31.90	11.15	25.48	22.87
Point.Gen.	31.87	11.20	25.42	17.83
GPG-ptr	31.49	11.02	25.37	18.62
GPG	<u>33.11</u>	<u>12.55</u>	<u>26.57</u>	<u>15.28</u>

Table 6.3: ROUGE score on XSum. * marks results from Narayan *et al.* (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$.

Models	% of NNs in CNN/Dailymail				% of NNs in Gigaword			% of NNs in XSum		
	NN-1	NN-2	NN-3	NN-S	NN-1	NN-2	NN-3	NN-1	NN-2	NN-3
Seq2seq	0.38	3.56	7.98	54.97	16.15	52.84	73.76	27.05	76.54	92.07
Point.Gen.	0.04	1.51	4.29	35.82	13.99	47.79	68.53	19.45	66.68	84.59
GPG-ptr	0.17	2.05	5.08	41.64	14.05	48.09	70.70	20.03	69.54	87.14
GPG	0.35	2.91	5.66	49.24	15.14	52.07	72.73	24.16	71.93	87.94
Reference	9.08	46.71	67.99	97.78	48.26	84.53	94.43	32.24	84.12	95.92

Table 6.4: Proportion of novel n-grams (NN-1,2,3) and sentences (NN-S) on generated summaries. GPG generate more novel words compared with standard pointer generators, though still slightly lower than seq2seq.

largely extractive (Zhang *et al.*, 2018a; Kryściński *et al.*, 2018). Though the Gigaword headline dataset is more abstractive, most words are simple paraphrases of some specific source word, so pure pointer GPG-ptr can work well. This is different from the XSum story summarization dataset where many target words require high-level abstraction or inference and cannot be aligned to a single source word, so combining the point and generation mode is necessary for a good performance.

The word perplexity results are consistent over all dataset ($GPG < GPG\text{-}ptr < Point.Gen. < seq2seq$). The reduction of perplexity does not necessarily indicate an increase for the ROUGE score, especially for pointer generators. This might

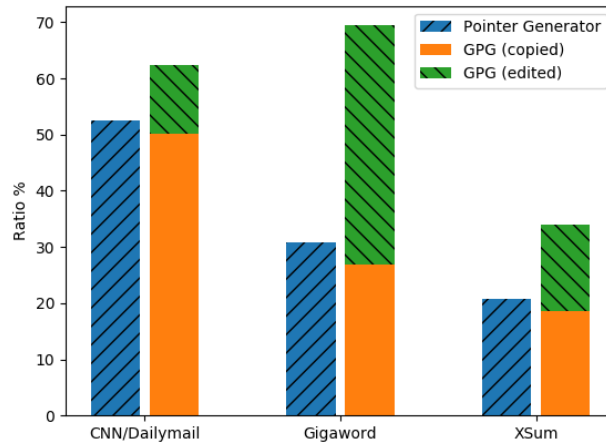


Figure 6.4: Pointing Ratio of the standard pointer generator and GPG (evaluated on the test data). GPG enables the point mode more often, but quite a few pointed tokens are edited rather than simply copied.

attribute to the different probability computation of pointer generators, where the probability of copied words are only normalized over the source words. This brings it an inherent advantage over other models where the normalization is over the whole 30k vocabularies.

Level of Abstraction: In Tab. 6.4, we look into how abstractive the generated summaries are by calculating the proportion of novel unigram, bigram and trigrams that do not exist in the corresponding source text. On CNN/DM, as the generations contain multiple sentences, we further report the proportion of novel sentences (obtained with NLTK sent_tokenize).

Tab. 6.4 reflects the clear difference in the level of abstraction (seq2seq > GPG > GPG-ptr > Point.Gen.). Though the seq2seq baseline generated most novel words, many of them are hallucinated facts (see Fig 6.6), as has also been noted in See *et al.* (2017). The abstraction of GPG model is close to seq2seq and much higher than copy-based pointer generators. We believe it comes from their ability of “editing” pointed tokens rather than simple copying them.

To examine the pointing behavior of the GPG model, we visualize the average pointing ratio on three dataset in Fig. 6.4. The pointing ratio can be considered as the chance that a word is generated from the point mode instead of the generation mode. We compute it as $(1 - p_{gen}) \sum_i a_{t,i} \delta(y_t | x_i) / p(y_t)$, averaged over all target tokens in the test data. For the GPG model, we further split it into the copy ratio (words that are exactly copied) and the editing ratio (words that are edited). We find the GPG model enables the point mode more frequently than standard pointer generators, especially on the Gigaword dataset (40% more). This also explains why a pure pointer model is more effective for Gigaword and CNN/DM. More than 60% target tokens can be generated from the point mode, while for XSum the ratio is less than

Article: (...) and bild reported that the (...)
Summary: (...) and bild report that the (...) [0.964]
Article: (...) thousands more death row prisoner (...)
Summary: (...) thousands more deaths row (...) [0.981]
Article: (...) surviving relatives of a woman who (...)
Summary: (...) family of woman who (...) [0.984]
Article: jordan 's crown prince (...)
Summary: jordanian crown prince (...) [0.993]
Article: a middle-aged man and a young girl (...) [0.814]
Summary: a man and a child died (...)
Article: (...) , was abducted in 2012 (...)
Summary: (...) was kidnapped in (...) [0.924]

Figure 6.5: Examples of summaries produced by GPG. Each two samples from CNN/DM, Gigaword and XSum (up to down). **bold** denotes novel words and their pointed source tokens. Bracketed numbers are the pointing probability ($1 - p_{gen}$) during decoding.

40%. Quite a few pointing operation includes text rewriting (**green bar** in Fig. 6.4). This could explain why our model is able to generate more novel words.

A few examples are displayed in Fig 6.5. We find our model frequently changes the tense (reported \rightarrow report), singular/plural (death \rightarrow deaths) or POS tag (jordan \rightarrow jordanian) of words. Sometimes it also paraphrases (relatives \rightarrow family) or abstracts a noun to its hypernym (girl \rightarrow child). The word editing might be wrong though. For example, “death row prisoner” is wrongly changed to “deaths row prisoner” in the second example, possibly because this phrase is rarely seen so that the model made an error by mistaking “death” as the main subject after “thousands more”¹⁷.

Human evaluation: We further perform a human evaluation to assess the model generations. We focus on evaluating the fluency and faithfulness since the ROUGE score often fails to quantify them (Schluter, 2017; Cao *et al.*, 2018). 100 random source-target pairs are sampled from the human-written DUC 2004 data for task 1&2 (Over *et al.*, 2007). Models trained on Gigaword are applied to generate corresponding summaries. The gold targets, together with the model generations are randomly shuffled then assigned to 10 human annotators. Each pair is evaluated by three different people and the most agreed score is adopted. Each pair is assigned a 0-1 score to indicate (1) whether the target is fluent in grammar, (2) whether the target

¹⁷It also reveals a limit of GPG model in that it only models token-level alignment. For phrases like death row prisoner, it cannot align it based on its compositional meaning.

	Fluency	Faithfulness	o/1
seq2seq	0.83	0.61	0.53
Point.Gen.	0.78	0.65	0.55
GPG-ptr	0.79	0.78	0.67
GPG	0.82	0.74	0.69
Gold	0.96	0.92	0.96

Table 6.5: Human evaluation results on DUC 2004. o/1 is the score for the o/1 Turing test.

	seq2seq	Point.Gen.	GPG
Prec	0.361	0.435 (0.512)	0.533 (0.628)

Table 6.6: Word Alignment Precision on DUC 2004. Number in bracket is the posterior alignment precision.

faithfully conveys the source information without hallucination and (3) whether the target is considered human-generated or machine-generated (like a o/1 Turing test). The averaged score is reported in Table 6.5. All models generally achieve high scores in fluency, but generations from GPG models are more faithful to the source information thereby have a larger chance of fooling people into believe they’re human-generated (over 0.1 higher score on the o/1 Turing test). This can be explained by GPG’s capability at capturing more latent alignments. As shown in Figure 6.4, GPG generates over half of the target words by its point mode. Words are generated by explicitly grounding on some source context instead of fabricating freely.

Fig. 6.6 compares some generation snippets. As can be observed, seq2seq models tend to freely synthesize wrong facts not grounded on the source text, especially on the more difficult XSum dataset. In the last example, seq2seq only capture the subject “odom” and some keywords “police”, “basketball” then start to freely fabricate random facts. Pointer generators are slightly better as it is trained to directly copy keyword from the source. However, once it starts to enter the generation mode (“of british” in example 2 and “has been arrested” in example 3), the generation also loses control. GPG largely alleviates the problems because it can point to an aligned source word, then transform it by a learned relation embedding. The explicit alignment modelling encourages the model to stay close to the source information.

Alignment Accuracy: We also manually annotate the word alignment on the same 100 DUC 2004 pairs. Following Daumé III and Marcu (2005), words are allowed to be aligned with a specific source word, phrase or a “null” anchor meaning that it cannot be aligned with any source word. The accuracy is only evaluated on the target words with a non-null alignment. For each target token, the most attended source word is considered as alignment (Ghader and Monz, 2017). For the pointer generator and GPG, we also induce the posterior alignment by applying the Bayes’

theorem. We report the alignment precision (Och and Ney, 2000) in Table 6.6, i.e., an alignment is considered as valid if it matches one of the human annotated ground truth.

The results show that GPG improves the alignment precision by 0.1 compared with the standard pointer generator. The posterior alignment is more accurate than the prior one (also reflected in Figure 6.1), enabling better human interpretation.

Article:	(...) marseille prosecutor brice robin told cnn that " so far no videos were used in the crash investigation . " he added , " a person who has such a video needs to immediately give it to the investigators . " robin 's comments follow claims by two magazines , german daily bild and french paris match (...)
Seq2seq:	marseille prosecutor brice robin tells cnn that " so far no videos were used in the crash investigation "
Point.Gen:	robin 's comments follow claims by two magazines , german daily bild and french (..)
GPG:	" so far no videos were used in the crash investigation , " prosecutor brice robin says (..)
Article:	surviving relatives of a woman who claimed she was raped ## years ago by the british queen 's representative in australia are seeking to withdraw a lawsuit against him , after the case drew widespread publicity in australia .
Seq2seq:	family of british queen 's representative in australia seeking to withdraw lawsuit against him .
Point.Gen:	surviving relatives of british queen 's representative seeking to withdraw lawsuit against him .
GPG:	family of woman who claimed she was victim of british queen 's representative seeks to withdraw lawsuit .
Article:	police were called to love ranch brothel in crystal , nevada , after he was found unresponsive on tuesday . the american had to be driven to hospital (...) mr odom , 35 , has played basketball for (...) lakers and clippers . he (...) was suspended from the nba for violating its anti-drug policy (...) was named nba sixth man of the year (...)
Seq2seq:	basketball legend odom odom has died at the age of 83 , police have confirmed .
Point.Gen:	a former nba sixth man has been arrested on suspicion of anti-drug offences in the us state of california .
GPG:	the american basketball association (lakers) star lamar odom has been found unconscious in the us state of nevada .

Figure 6.6: Examples of generated summaries. Examples are taken from CNN/DM, Gigaword and XSum (from up to down). Darker means higher pointing probability.

6.6 CONCLUSION

In this chapter, we propose generalizing the pointer generator to go beyond exact copy operation. At each decoding step, the decoder can either generate from the vocabulary, copy or edit some source words by estimating a relation embedding. Experiments on abstractive summarization show the generalized model generates more abstract summaries yet faithful to the source information. The generalized pointer is able to capture richer latent alignment relationship beyond exact copies. This helps improve the alignment accuracy, allowing better model controllability and interpretation.

We believe the generalized pointer mechanism could have potential applications in many fields where tokens are not exactly copied. By integrating off-the-shelf knowledge bases to clearly model the transition relation embedding, it should further improve the interpretability and might be especially helpful under low-resource settings, which we leave for future work.

A major drawback of the proposed approach is that it can only model word-level alignment. In many cases, a more natural alignment relationship is at the segment-level (phrases or word spans). In the next chapter, we will show it is possible to use latent variables to jointly model segment-level and word-level alignment with a tractable marginal likelihood.

THE neural attention model has achieved great success in data-to-text generation tasks. Though usually excelling at producing fluent text, it suffers from the problem of information missing, repetition and “hallucination”. Due to the black-box nature of the neural attention architecture, avoiding these problems in a systematic way is non-trivial. To address this concern, we propose to explicitly segment target text into fragment units and align them with their data correspondences. The segmentation and correspondence are jointly learned as latent variables without any human annotations. The major different with the last chapter is that we can model both segment and word level alignment. This makes it appealing for specific tasks like data-to-text where the input is naturally segmented into several records. We further impose a soft statistical constraint to regularize the segmental granularity. The resulting architecture maintains the same expressive power as neural attention models, while being able to generate fully interpretable outputs with several times less computational cost. On both E2E and WebNLG benchmarks, we show the proposed model consistently outperforms its neural attention counterparts (Shen *et al.*, 2020).

7.1 INTRODUCTION

Data-to-text generation aims at automatically producing natural language descriptions of structured database (Reiter and Dale, 1997). Traditional statistical methods usually tackle this problem by breaking the generation process into a set of local decisions that are learned separately (Belz, 2008; Angeli *et al.*, 2010; Kim and Mooney, 2010; Oya *et al.*, 2014). Recently, neural attention models conflate all steps into a single end-to-end system and largely simplify the training process (Mei *et al.*, 2016; Lebrete *et al.*, 2016; Shen *et al.*, 2017c, 2018b; Su *et al.*, 2019b; Chang *et al.*, 2020). However, the black-box conflation also renders the generation uninterpretable and hard to control (Wiseman *et al.*, 2018; Shen *et al.*, 2019b). Verifying the generation correctness in a principled way is non-trivial. In practice, it often suffers from the problem of information missing, repetition and “hallucination” (Dušek *et al.*, 2018, 2020).

In this work, we propose to explicitly exploit the *segmental structure* of text. Specifically, we assume the target text is formed from a sequence of segments. Every segment is the result of a two-stage decision: (1) Select a proper data record to be described and (2) Generate corresponding text by *paying attention only to the selected data record*. This decision is repeated until all desired records have been realized.

Source data:

Name[Clowns], PriceRange[more than £30],
EatType[pub], FamilyFriendly[no]

Generation:

①Name → ②(Clowns)
③FamilyFriendly → ④(is a child-free)
⑤PriceRange → ⑥(, expensive)
⑦EatType → ⑧(pub.)

Figure 7.1: Generation from our model on the E2E dataset. Decoding is performed segment-by-segment. Each segment realizes one data record. ①~⑧ mark the decision order in the generation process.

Figure 7.1 illustrates this process.

Compared with neural attention, the proposed model has the following advantages: (1) We can monitor the corresponding data record for every segment to be generated. This allows us to easily control the output structure and verify its correctness¹⁸. (2) Explicitly building the correspondence between segments and data records can potentially reduce the hallucination, as noted in (Wu *et al.*, 2018; Deng *et al.*, 2018) that hard alignment usually outperforms soft attention. (3) When decoding each segment, the model pays attention only to the selected data record instead of averaging over the entire input data. This largely reduces the memory and computational costs¹⁹.

To train the model, we *do not* rely on any human annotations for the segmentation and correspondence, but rather marginalize over all possibilities to maximize the likelihood of target text, which can be efficiently done within polynomial time by dynamic programming. This is essentially similar to traditional methods of inducing segmentation and alignment with semi-markov models (Daumé III and Marcu, 2005; Liang *et al.*, 2009). However, they make strong independence assumptions thus perform poorly as a generative model (Angeli *et al.*, 2010). In contrast, the transition and generation in our model condition on *all previously generated text*. By integrating an autoregressive neural network structure, our model is able to capture unbounded dependencies while still permitting tractable inference. The training process is stable as it does not require any sampling-based approximations. We further add a soft statistical constraint to control the segmentation granularity via posterior regularization (Ganchev *et al.*, 2010). On both the E2E and WebNLG benchmarks, our model is able to produce significantly higher-quality outputs while being several

¹⁸For example, we can perform a similar constrained decoding as in Balakrishnan *et al.* (2019) to rule out outputs with undesired patterns.

¹⁹Coarse-to-fine attention (Ling and Rush, 2017; Deng *et al.*, 2017) was proposed for the same motivation, but they resort to reinforcement learning which is hard to train, and the performance is sacrificed for efficiency.

times computationally cheaper. Due to its fully interpretable segmental structure, it can be easily reconciled with heuristic rules or hand-engineered constraints to control the outputs.

7.2 RELATED WORK

Data-to-text generation is traditionally dealt with using a pipeline structure containing content planning, sentence planning and linguistic realization (Reiter and Dale, 1997). Each target text is split into meaningful fragments and aligned with corresponding data records, either by hand-engineered rules (Kukich, 1983; McKeown, 1992) or statistical induction (Liang *et al.*, 2009; Koncel-Kedziorski *et al.*, 2014; Qin *et al.*, 2018). The segmentation and alignment are used as supervision signals to train the content and sentence planner (Barzilay and Lapata, 2005; Angeli *et al.*, 2010). The linguistic realization is usually implemented by template mining from the training corpus (Kondadadi *et al.*, 2013; Oya *et al.*, 2014). Our model adopts a similar pipeline generative process, but integrates all the sub-steps into a single end-to-end trainable neural architecture. It can be considered as a neural extension of the PCFG system in Konstas and Lapata (2013), with a more powerful transition probability considering inter-segment dependence and a state-of-the-art attention-based language model as the linguistic realizer. Wiseman *et al.* (2018) tried a similar neural generative model to induce templates. However, their model only captures loose data-text correspondence and adopts a weak markov assumption for the segment transition probability. Therefore, it underperforms the neural attention baseline as for generation. Our model is also in spirit related to recent attempts at separating content planning and surface realization in neural data-to-text models (Zhao *et al.*, 2018b; Puduppully *et al.*, 2019; Moryossef *et al.*, 2019; Ferreira *et al.*, 2019). Nonetheless, all of them resort to *manual annotations or hand-engineered rules applicable only for a narrow domain*. Our model, instead, automatically learn the optimal content planning via exploring over exponentially many segmentation/correspondence possibilities.

There have been quite a few neural alignment models applied to tasks like machine translation (Wang *et al.*, 2018a; Deng *et al.*, 2018), character transduction (Wu *et al.*, 2018; Shankar and Sarawagi, 2019) and summarization (Yu *et al.*, 2016; Shen *et al.*, 2019c). Unlike word-to-word alignment, we focus on learning the alignment between data records and text segments. Some works also integrate neural language models to jointly learn the segmentation and correspondence, e.g., phrase-based machine translation (Huang *et al.*, 2018), speech recognition (Wang *et al.*, 2017a) and vision-grounded word segmentation (Kawakami *et al.*, 2019). Data-to-text naturally fits into this scenario since each data record is normally verbalized in one continuous text segment.

7.3 BACKGROUND: DATA-TO-TEXT

Let X, Y denote a source-target pair. X is structured data containing a set of records and Y corresponds to y_1, y_2, \dots, y_m which is a text description of X . The goal of data-to-text generation is to learn a distribution $p(Y|X)$ to automatically generate proper text describing the content of the data.

The neural attention architecture handles this task with an encode-attend-decode process (Bahdanau *et al.*, 2015). The input X is processed into a sequence of x_1, x_2, \dots, x_n , normally by flattening the data records (Wiseman *et al.*, 2017). The encoder encodes each x_i into a vector h_i . At each time step, the decoder attends to encoded vectors and outputs the probability of the next token by $p(y_t|y_{1:t-1}, A_t)$. A_t is a weighted average of source vectors:

$$A_t = \sum_i \alpha_{t,i} h_i$$

$$\alpha_{t,i} = \frac{e^{f(h_i, d_t)}}{\sum_j e^{f(h_j, d_t)}} \quad (7.1)$$

d_t is the hidden state of the decoder at time step t . f is a score function to compute the similarity between h_i and d_t (Luong *et al.*, 2015).

7.4 APPROACH

Suppose the input data X contains a set of records r_1, r_2, \dots, r_K . Our assumption is that the target text $y_{1:m}$ can be segmented into a sequence of fragments. Each fragment corresponds to one data record. As the ground-truth segmentation and correspondence are not available, we need to enumerate over all possibilities to compute the likelihood of $y_{1:m}$. Denote by \mathcal{S}_y the set containing all valid segmentation of $y_{1:m}$. For any valid segmentation $s_{1:\tau_s} \in \mathcal{S}_y$, $\pi(s_{1:\tau_s}) = y_{1:m}$, where π means concatenation and τ_s is the number of segments. For example, let $m = 5$ and $\tau_s = 3$. One possible segmentation would be $s_{1:\tau_s} = \{\{y_1, y_2, \$\}, \{y_3, \$\}, \{y_4, y_5, \$\}\}$. $\$$ is the end-of-segment symbol and is removed when applying the π operator. We further define $c(*)$ to be the corresponding data record(s) of $*$. The likelihood of each text is then computed by enumerating over all possibilities of $s_{1:\tau_s}$ and $c(s_{1:\tau_s})$:

$$p(y_{1:m}|X) = \sum_{s_{1:\tau_s} \in \mathcal{S}_y} p(s_{1:\tau_s}|X)$$

$$= \sum_{s_{1:\tau_s} \in \mathcal{S}_y} \prod_{o=1}^{\tau_s} \sum_{c(s_o)=r_1}^{r_K} p(s_o|\pi(s_{<o}), c(s_o))$$

$$\times p(c(s_o)|\pi(s_{<o}), c(s_{<o})) \quad (7.2)$$

Every segment is generated by first selecting the data record based on the *transition probability* $p(c(s_o)|\pi(s_{<o}), c(s_{<o}))$, then generating tokens based on the word genera-

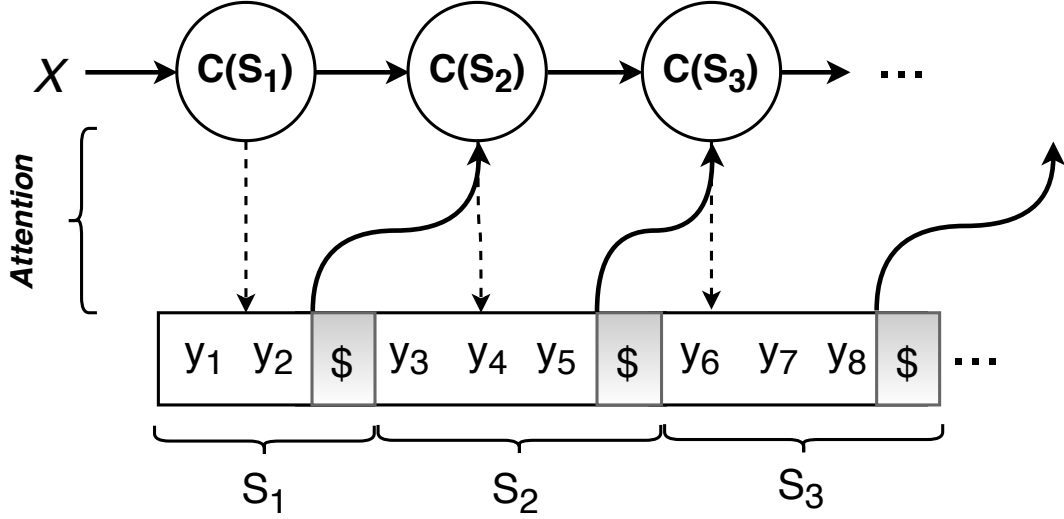


Figure 7.2: Generation process of our approach. Segment end symbol \$ is ignored when updating the state of the decoder. *Solid arrows* indicate the transition model and *dashed arrows* indicate the generation model. Every segment s_o is generated by attending only to the corresponding data record $c(s_o)$.

tion probability $p(s_o | \pi(s_{<o}), c(s_o))$. Figure 7.2 illustrates the generation process of our model.

Generation Probability. We base the generation probability on the same decoder as in neural attention models. The only difference is that *the model can only pay attention to its corresponding data record*. The attention scores of other records are masked out when decoding s_o :

$$\alpha_{t,i} = \frac{e^{f(h_i, d_t)} \mathbb{1}(x_i \in c(s_o))}{\sum_j e^{f(h_j, d_t)} \mathbb{1}(x_j \in c(s_o))}$$

where $\mathbb{1}$ is the indicator function. This forces the model to learn proper correspondences and enhances the connection between each segment and the data record it describes.

Following the common practice, we define the output probability with the pointer generator (See *et al.*, 2017; Wiseman *et al.*, 2017):

$$\begin{aligned} p_{gen} &= \sigma(\text{MLP}_g([d_t \circ A_t])) \\ p_{vocab} &= \text{softmax}(W_1 d_t + W_2 A_t) \\ p_{\theta}(y_t | y_{<t}) &= p_{gen} p_{vocab}(y_t) \\ &\quad + (1 - p_{gen}) \sum_{i: y_t = x_i} \alpha_{t,i} \end{aligned}$$

d_t is the decoder's hidden state at time step t . \circ denotes vector concatenation. A_t is the context vector. MLP indicates multi-layer perceptron and σ normalizes the

score between $(0, 1)$. W_1 and W_2 are trainable matrices. p_{gen} is the probability that the word is generated from a fixed vocabulary distribution p_{vocab} instead of being copied. The final decoding probability $p_\theta(y_t)$ is marginalized over p_{vocab} and the copy distribution. The generation probability of s_o factorizes over the words within it and the end-of-segment token:

$$p(s_o | \pi(s_{<o}), c(s_o)) = p_\theta(\$ | y_{1:t}) \prod_{y_t \in s_o} p_\theta(y_t | y_{<t})$$

Transition Probability. We make a mild assumption that $c(s_o)$ is dependent only on $c(s_{o-1})$ and $\pi(s_{1:o-1})$ but irrelevant of $c(s_{<o-1})$, which is a common practice when modelling alignment (Och *et al.*, 1999; Yu *et al.*, 2016; Shankar and Sarawagi, 2019). The transition probability is defined as:

$$\begin{aligned} p(c(s_o) = r_i | c(s_{<o}), \pi(s_{<o})) \\ \approx p(c(s_o) = r_i | c(s_{o-1}), \pi(s_{<o})) \\ \propto f(r_i)^T [M^T A_{s_{o-1}} + N^T d_{s_{o-1}}] \end{aligned} \quad (7.3)$$

A softmax layer is finally applied to the above equation to normalize it as a proper probability distribution. $f(r_i)$ is a representation of r_i , which is defined as a max pooling over all the word embeddings contained in r_i . $A_{s_{o-1}}$ is the attention context vector when decoding the last token in s_{o-1} , defined as in Equation 7.1. It carries important information from $c(s_{o-1})$ to help predict $c(s_o)$. $d_{s_{o-1}}$ is the hidden state of the neural decoder which goes through all history tokens $\pi(s_{1:o-1})$. M, N are trainable matrices to project $A_{s_{o-1}}$ and $d_{s_{o-1}}$ into the same dimension as $f(r_i)$.

We further add one constraint to prohibit *self-transition*, which can be easily done by zeroing out the transition probability in Equation 7.3 when $c(s_o) = c(s_{o-1})$. This forces the model to group together text describing the same data record.

Since Equation 7.3 conditions on all previously generated text, it is able to capture more complex dependencies as in semi-markov models (Liang *et al.*, 2009; Wiseman *et al.*, 2018).

Null Record. In our task, we find some frequent phrases, e.g., “it is”, “and”, tend to be wrongly aligned with some random records, similar to the garbage collection issue in statistical alignment (Brown *et al.*, 1993). This hurt the model interpretability. Therefore, we introduce an additional null record r_0 to attract these non-content phrases. The context vector when aligned to r_0 is a zero vector so that the decoder will decode words based solely on the language model without relying on the input data.

Training. Equation 7.2 contains exponentially many combinations to enumerate over. Here we show how to efficiently compute the likelihood with the forward algorithm in dynamic programming (Rabiner, 1989). We define the forward variable $\alpha(i, j) = p(y_{1:i}, c(y_i) = j | X)$. With the base $\alpha(1, j) = p(y_1 | c(y_1) = j)$. The recursion

goes as follows for $i = 1, 2, \dots, m - 1$:

$$\begin{aligned} \alpha(i+1, j) &= \sum_{p=1}^i \sum_{q=r_0}^{r_K} \alpha(p, q) \\ &\times p(c(y_{p+1}) = j | c(y_p) = q, y_{1:p}) \\ &\times p(y_{p+1:i+1} | c(y_{p+1:i+1}) = q, y_{1:p}) \\ &\times p(\$ | c(y_{p+1:i+1}) = q, y_{1:i+1}) \end{aligned} \quad (7.4)$$

The final likelihood of the target text can be computed as $p(y_{1:m} | X) = \sum_{j=r_0}^{r_K} \alpha(m, j)$. As the forward algorithm is fully differentiable, we maximize the log-likelihood of the target text by backpropagating through the dynamic programming. The process is essentially equivalent to the generalized EM algorithm (Eisner, 2016). By means of the modern automatic differentiation tools, we avoid the necessity to calculate the posterior distribution manually (Kim *et al.*, 2018a).

To speed up training, we set a threshold L to the maximum length of a segment as in Liang *et al.* (2009); Wiseman *et al.* (2018). This changes the complexity in Equation 7.4 to a constant $O(LK)$ instead of scaling linearly with the length of the target text. Moreover, as pointed out in Wang *et al.* (2017a), the computation for the longest segment can be reused for shorter segments. We therefore first compute the generation and transition probability for the whole sequence in one pass. The intermediate results are then cached to efficiently proceed the forward algorithm without any re-computation.

One last issue is the numerical precision, it is important to use the log-space binary operations to avoid underflow (Kim *et al.*, 2017).

Near[riverside], Food[French], EatType[pub], Name[Cotto]	
1. [Near]Near[the]Null[riverside]Near[is a]Null[French]Food	[pub]EatType[called]Null[Cotto]Name[.]Null
2. [Near the riverside]Near[is]Null[a French]Food[pub]EatType	[called Cotto]Name[.]Null
3. [Near the riverside]Near[is a French]Food[pub]EatType	[called Cotto .]Name
4. [Near the riverside]Near[is a French pub]Food	[called Cotto .]Name

Table 7.1: Segmentation with various granularities. 1 is too fine-grained while 4 is too coarse. We expect a segmentation like 2 or 3 to better control the generation.

Segmentation Granularity. There are several valid segmentations for a given text. As shown in Table 7.1, when the segmentation (example 1) is too fine-grained, controlling the output information becomes difficult because the content of one data record is realized in separate pieces ²⁰. When it is too coarse, the alignment might

²⁰The finer-grained segmentation might be useful if the focus is on modeling the detailed discourse structure instead of the information accuracy (Reed *et al.*, 2018; Balakrishnan *et al.*, 2019), which we

become less accurate (as in Example 4, “pub” is wrongly merged with previous words and aligned together to the “Food” record). In practice, we expect the segmentation to stay with accurate alignment yet avoid being too brokenly separated. To control the granularity as we want, we utilize posterior regularization (Ganchev *et al.*, 2010) to constrain the expected number of segments for each text ²¹, which can be calculated by going through a similar forward pass as in Equation 7.4 (Eisner, 2002). Most computation is shared without significant extra burden. The final loss function is:

$$-\log \mathbb{E}_{\mathcal{S}_y} p(s_{1:\tau_s} | X) + \max(|\mathbb{E}_{\mathcal{S}_y} \tau_s - \eta|, \gamma) \quad (7.5)$$

$\log \mathbb{E}_{\mathcal{S}_y} p(s_{1:\tau_s} | X)$ is the log-likelihood of target text after marginalizing over all valid segmentations. $\mathbb{E}_{\mathcal{S}_y} \tau_s$ is the expected number of segments and η, γ are hyperparameters. We use the max-margin loss to encourage $\mathbb{E}_{\mathcal{S}_y} \tau_s$ to stay close to η under a tolerance range of γ .

Decoding. The segment-by-segment generation process allows us to easily constrain the output structure. Undesirable patterns can be rejected before the whole text is generated. We adopt three simple constraints for the decoder:

1. Segments must not be empty.
2. The same data record cannot be realized more than once (except for the null record).
3. The generation will not finish until all data records have been realized.

Constraint 2 and 3 directly address the information repetition and missing problem. When segments are incrementally generated, the constraints will be checked against for validity. Note that adding the constraints hardly incur any cost, the decoding process is still finished *in one pass*. No post-processing or reranking is needed.

Computational Complexity. Suppose the input data has M records and each record contains N tokens. The computational complexity for neural attention models is $O(MN)$ at each decoding step where the whole input is retrieved. Our model, similar to chunkwise attention (Chiu and Raffel, 2018) or coarse-to-fine attention (Ling and Rush, 2017), reduces the cost to $O(M + N)$, where we select the record in $O(M)$ at the beginning of each segment and attend only to the selected record in $O(N)$ when decoding every word. For larger input data, our model can be significantly cheaper than neural attention models.

leave for future work.

²¹We can also utilize some heuristic rules to help segmentation. For example, we can prevent breaking syntactic elements obtained from an external parser (Yang *et al.*, 2019) or match entity names with handcrafted rules (Chen *et al.*, 2018). The interpretability of the segmental structure allows easy combination with these rules. We focus on a general *domain-agnostic* method in this paper, though heuristic rules might bring further improvement under certain cases.

7.5 EXPERIMENT SETUP

Dataset. We conduct experiments on the E2E (Novikova *et al.*, 2017b) and WebNLG (Gardent *et al.*, 2017) datasets. E2E is a crowd-sourced dataset containing 50k instances in the restaurant domain. The inputs are dialogue acts consisting of three to eight slot-value pairs. WebNLG contains 25k instances describing entities belonging to fifteen distinct DBpedia categories. The inputs are up to seven RDF triples of the form (*subject, relation, object*).

Implementation Details. We use a bi-directional LSTM encoder and uni-directional LSTM decoder for all experiments. Input data records are concatenated into a sequence and fed into the encoder. We choose the hidden size of encoder/decoder as 512 for E2E and 256 for WebNLG. The word embedding is with size 100 for both datasets and initialized with the pre-trained Glove embedding²² (Pennington *et al.*, 2014). We use a drop out rate of 0.3 for both the encoder and decoder. Models are trained using the Adam optimizer (Kingma and Ba, 2015) with batch size 64. The learning rate is initialized to 0.01 and decays an order of magnitude once the validation loss increases. All hyperparameters are chosen with grid search according to the validation loss. Models are implemented based on the open-source library PyTorch (Paszke *et al.*, 2019). We set the hyperparameters in Eq. 7.5 as $\eta = K, \gamma = 1$ (recall that K is the number of records in the input data). The intuition is that every text is expected to realize the content of all K input records. It is natural to assume every text can be roughly segmented into K fragments, each corresponding to one data record. A deviation of $K \pm 1$ is allowed for noisy data or text with complex structures.

Metrics. We measure the quality of system outputs from three perspectives: (1) *word-level overlap* with human references, which is a commonly used metric for text generation. We report the scores of BLEU-4 (Papineni *et al.*, 2002), ROUGE-L (Lin, 2004), Meteor (Banerjee and Lavie, 2005) and CIDEr (Vedantam *et al.*, 2015). (2) *human evaluation*. Word-level overlapping scores usually correlate rather poorly with human judgements on fluency and information accuracy (Reiter and Belz, 2009; Novikova *et al.*, 2017a). Therefore, we passed the input data and generated text to human annotators to judge if the text is fluent by grammar (scale 1-5 as in Belz and Reiter (2006)), contains wrong fact inconsistent with input data, repeats or misses information. We report the *averaged score* for fluency and *definite numbers* for others. The human is conducted on a sampled subset from the test data. To ensure the subset covers inputs with all possible number of records ($K \in [3, 8]$ for E2E and $K \in [1, 7]$ for WebNLG), we sample 20 instances for every possible K . Finally, we obtain 120 test cases for E2E and 140 for WebNLG²³. (3) *Diversity of outputs*. Diversity is an important concern for many real-life applications. We measure it by the number of

²²nlp.stanford.edu/data/glove.6B.zip

²³The original human evaluation subset of WebNLG is randomly sampled, most of the inputs contain less than 3 records, so we opt for a new sample for a thorough evaluation.

Metrics Models	Word Overlap				Human Evaluation				Diversity	
	BLEU	R-L	Meteor	CIDEr	Fluent	Wrong	Repeat	Miss	Dist-1	Dist-3
Slug	0.662	0.677	0.445	2.262	4.94	5	0	17	74	507
DANGNT	0.599	0.663	0.435	2.078	4.97	0	0	21	61	301
TUDA	0.566	0.661	0.453	1.821	4.98	0	0	10	57	143
N_Temp	0.598	0.650	0.388	1.950	4.84	19	3	35	119	795
PG	0.638	0.677	0.449	2.123	4.91	15	1	29	133	822
Ours	0.647	0.683	0.453	2.222	4.96	0	1	15	127	870
Ours (+R)	0.645	0.681	0.452	2.218	4.95	0	0	13	133	881
Ours (+RM)	0.651	0.682	0.455	2.241	4.95	0	0	3	135	911

Table 7.2: Automatic and human evaluation results on E2E dataset. **SLUG**, **DANGNT**, **TUDA** and **N_TEMP** are from previous works and the other models are our own implementations.

Metrics Models	Word Overlap				Human Evaluation				Diversity	
	BLEU	R-L	Meteor	CIDEr	Fluent	Wrong	Repeat	Miss	Dist-1	Dist-3
Melbourne	0.450	0.635	0.376	2.814	4.16	42	22	37	3167	13,744
UPF-FORGe	0.385	0.609	0.390	2.500	4.08	29	6	28	3191	12,509
PG	0.452	0.652	0.384	2.623	4.13	43	26	42	3,218	13,403
Ours	0.453	0.656	0.388	2.610	4.23	26	19	31	3,377	14,516
Ours (+R)	0.456	0.657	0.390	2.678	4.28	18	2	24	3,405	14,351
Ours (+RM)	0.461	0.654	0.398	2.639	4.26	23	4	5	3,457	14,981

Table 7.3: Automatic and human evaluation results on WebNLG dataset. **MELBOURNE** and **UPFUPF-FORGE** are from previous works and the other models are our own implementations.

unique unigrams and trigrams over system outputs, as done in Dušek *et al.* (2020).

7.6 RESULTS

In this section, we first show the effects of the granularity regularization we proposed, then compare model performance on two datasets and analyze the performance difference. Our model is compared against the neural attention-based pointer generator (**PG**) which does not explicit learn the segmentation and correspondence. To show the effects of the constrained decoding, we run our model with only the first constraint to prevent empty segments (denoted by **ours** in experiments), with the first two constraints to prevent repetition (denoted by **ours (+R)**), and with all constraints to further reduce information missing (denoted by **ours (+RM)**).

Segmentation Granularity. We show the effects of the granularity regularization (§7.4, Segmentation Granularity) in Fig 7.3. When varying the model size, the segmentation granularity changes much if no regularization is imposed. Intuitively if the generation module is strong enough (larger hidden size), it can accurately estimate the sentence likelihood itself without paying extra cost of switching between segments, then it tends to reduce the number of transitions. Vice versa, the number

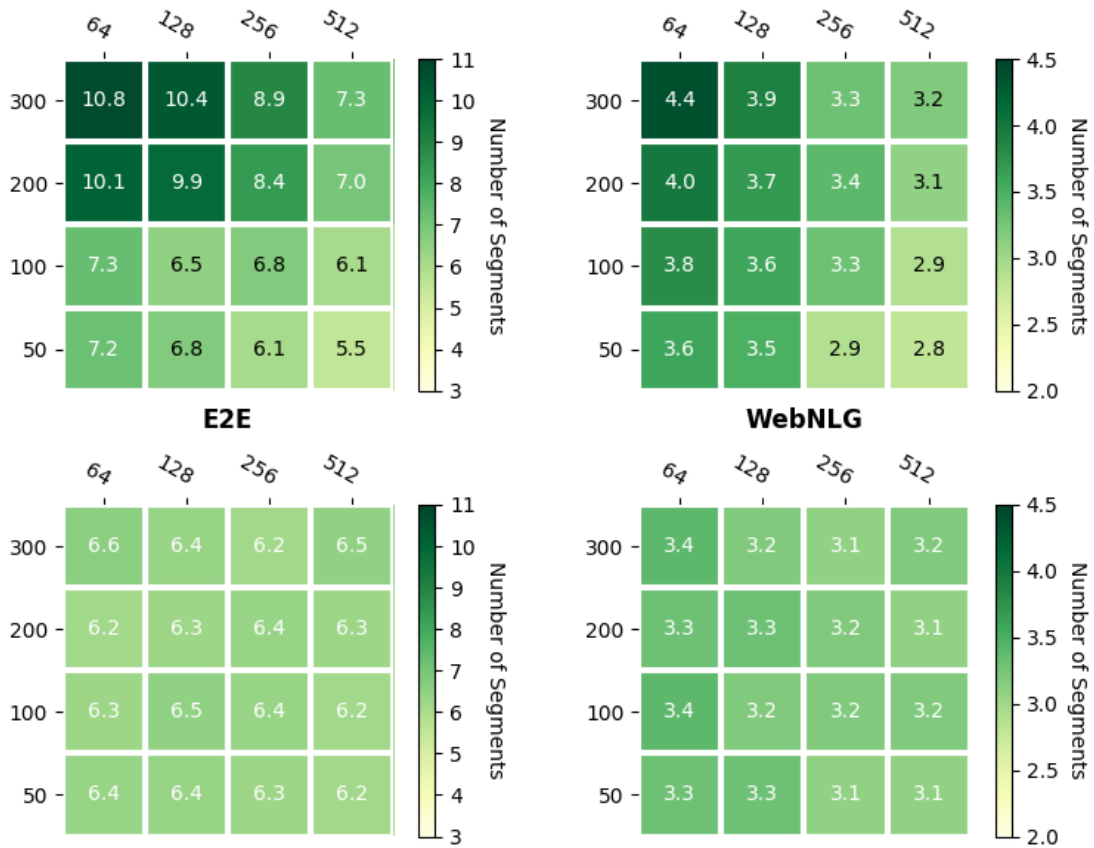


Figure 7.3: Average expected number of segments with varying hyperparameters. x-axis is the encoder/decoder hidden size and y-axis is the word embedding size. Upper two figures are without the granularity regularization and the bottom two are with regularization.

of transitions will grow if the transition module is stronger (larger embedding size). With the regularization we proposed, the granularity remains what we want regardless of the hyperparameters. We can thereby freely decide the model capacity without worrying about the difference of segmentation behavior.

Results on E2E. On the E2E dataset, apart from our implementations, we also compare against outputs from the **SLUG** (Juraska *et al.*, 2018), the overall winner of the E2E challenge (seq2seq-based), **DANGNT** (Nguyen and Tran, 2018), the best grammar rule based model, **TUDA** (Puzikov and Gurevych, 2018), the best template based model, and the autoregressive neural template model (**N_TEMP**) from Wiseman *et al.* (2018). SLUG uses a heuristic slot aligner based on a set of handcrafted rules and combine a complex pipeline of data augmentation, selection, model ensemble and reranker, while our model has a simple end-to-end learning paradigm with no special delexicalizing, training or decoding tricks. Table 7.2 reports the evaluated results. Seq2seq-based models are more diverse than rule-

	<div>Egg Harbor Township, New Jersey <u>isPartOf</u> New Jersey</div> <div>Atlantic City International Airport <u>location</u> Egg Harbor Township, New Jersey</div> <div>Egg Harbor Township, New Jersey <u>isPartOf</u> Atlantic County, New Jersey</div> <div>Atlantic City International Airport <u>Location Identifier</u> "KACY" ICAO</div> <div>Egg Harbor Township, New Jersey <u>country</u> United States</div>
PG	Atlantic City International Airport is located in Egg Harbor Township , New Jersey , United States . It is located in Egg Harbor Township , New Jersey .
Ours	<div>KACY is the ICAO location identifier of Atlantic City International Airport ,</div> <div>which is located at Egg Harbor Township , New jersey , in the United States]</div> <div>. The ICAO location identifier of Atlantic City International Airport is KACY .</div>
Ours (+R)	<div>KACY is the ICAO location identifier of Atlantic City International Airport ,</div> <div>which is located at Egg Harbor Township , New jersey , in the United States]</div> <div>.</div>
Ours (+RM)	<div>KACY is the ICAO location identifier of Atlantic City International Airport , which is located at</div> <div>Egg Harbor Township , New jersey , in the United States</div> <div>. The Egg Harbor Township is a part of Atlantic County , New Jersey .</div> <div>Egg Harbor Township is a part of New Jersey .</div>

Figure 7.4: Example generations from WebNLG. Relation types are underlined and repeated generations are **bolded**. Segments and corresponding records in our model are marked in the same color. By adding explicit constraints to the decoding process, repetition and missing issues can be largely reduced. (better viewed in color)

based models at the cost of higher chances of making errors. As rule-based systems are by design always faithful to the input information, they made zero wrong facts in their outputs. Most models do not have the fact repetition issue because of the relatively simple patterns in the E2E dataset. therefore, adding the (+R) constraint only improves the performance minorly. The (+RM) constraint reduces the number of information missing to 3 without hurting the fluency. All the 3 missing cases are because of the wrong alignment between the period and one data record, which can be easily fixed by defining a simple rule. We put the error analysis in §7.A. N_Temp performs worst among all seq2seq-based systems because of the restrictions we mentioned in §7.2. As also noted by the author, it trades-off the generation quality for interpretability and controllability. In contrast, our model, despite relying on no heuristics or complex pipelines, *made zero wrong facts with the lowest information missing rate, even surpassing rule-based models*. It also maintains interpretable and controllable without sacrificing the generation quality.

Results on WebNLG. Table 7.3 reports the results evaluated on the WebNLG dataset. We also include results from **MELBOURNE**, a seq2seq-based system achieving highest scores on automatic metrics in the WebNLG challenge and **UPF-FORGE**, a classic grammar-based system that wins in the human evaluation WebNLG contains significantly more distinct types of attributes than E2E, so the chance of making

Input:	[name the mill]	[eattype restaurant]	[food english]	[pricerange moderate]	[customerrating 1 out of 5]	[area riverside] ...
PG:	the mill is a low - priced restaurant in the city centre that delivers take - away . it is located near café rouge.					
Input:	[name the mill]	[eattype restaurant]	[food english]	[pricerange moderate]	[customerrating 1 out of 5]	[areariverside] ...
Ours:	[the mill]	[restaurant]	[near café rouge]	[in riverside]	[serves english food]	[at moderate prices][. it is kid friendly and]...

Table 7.4: (E2E) Attention map when decoding the word “low” in the PG model and “moderate” in our model. Hallucinated contents are **bolded**. The PG model wrongly attended to other slots thereby “hallucinated” the content of “low-priced”. Our model always attends to one single slot instead of averaging over the whole inputs, the chance of hallucination is largely reduced.

errors or repetitions increases greatly. Nevertheless, our model still *performs on-par on automatic metrics with superior information adequacy and output diversity*. The (+R) decoding constraint becomes important since the outputs in WebNLG are much longer than those in E2E, neural network models have problems tracking the history generation beyond certain range. Models might repeat facts that have been already generated long back before. The (+R) constraint effectively reduces the repetition cases from 19 to 2. These 2 cases are intra-segment repetitions and failed to be detected since our model can only track inter-segment constraints (examples are in §7.A). The (+RM) constraint brings down the information missing cases to 5 with slightly more wrong and repeated facts compared with (+R). Forcing models to keep generating until covering all records will inevitably increase the risk of making errors.

Discussions. In summary, our models generates *most diverse outputs, achieves similar or better performances in word-overlap automatic metrics while significantly reduces the*

information hallucination, repetition and missing problems. An example of hallucination is shown in Table 7.4. The standard PG model “hallucinated” the contents of “low-priced”, “in the city center” and “delivers take-away”. The visualized attention maps reveal that it failed to attend properly when decoding the word “low”. The decoding is driven mostly by language models instead of the contents of input data. In contrast, as we explicitly align each segment to one slot, the attention distribution of our model is *concentrated on one single slot rather than averaged over the whole input*, the chance of hallucinating is therefore largely reduced.

Figure 7.4 shows some example generations from WebNLG. Without adding the decoding constraints, PG and our model both suffer from the problem of information repetition and missing. However, the interpretability of our model enables us to easily avoid these issues by constraining the segment transition behavior. For the attention-based PG model, there exists no simple way of applying these constraints. We can also explicitly control the output structure similar to Wiseman *et al.* (2018), examples are shown in §7.B.

7.7 CONCLUSION

In this work, we exploit the segmental structure in data-to-text generation. The proposed model significantly alleviates the information hallucination, repetition and missing problems without sacrificing the fluency and diversity. It is end-to-end trainable, domain-independent and allows explicit control over the structure of generated text. As our model is interpretable in the correspondence between segments and input records, it can be easily combined with hand-engineered heuristics or user-specific requirements to further improve the performance.

7.A ERROR ANALYSIS

We analyze common errors below.

Missing: Even with the coverage decoding constraint, the model can still occasionally miss information. We show one example in Table 7.5. The segments cover all input records, but the segment aligned to “familyfriendly” only generates a period symbol. This happens 3 times on E2E and twice on WebNLG. On the other 3 cases of missing on WebNLG, some segments only generate one end-of-sentence symbol. Both conditions can be easily fixed by some simple filtering rules.

Repeating: There are still some repeating cases on the WebNLG dataset. Table 7.6 shows one example. “amsterdam-centrum is part of amsterdam” is repeated twice within a segment. As our constraint decoding can only prevent inter-segment repetition, it cannot fully avoid the repetition problem resulting from the intra-segment errors of RNNs.

Input:	name the phoenix	eattype pub	food french	pricerange £20 - 25	customer rating high	area riverside
	familyfriendly yes	near crowne plaza hotel				
Output:	the phoenix	pub is located in riverside	near crowne plaza hotel	it serves french food	in the £20 -	
	25 price range	it has a high customer rating	.			

Table 7.5: Example of missing in E2E. The “familyfriendly” is wligned to the period symbol.

Input:	Amsterdam ground AFC Ajax (amateurs)	Eberhard van der Laan leader Amsterdam	Amsterdam-Centrum part Amsterdam
Output:	amsterdam-centrum is part of amsterdam and amsterdam-centrum is part of amsterdam , the country where eberhard van der laan is the leader and the ground of afc ajax (amateurs) is located.		

Table 7.6: (E2E) Example of repetition in WebNLG. The phrase “amsterdam-centrum is part of amsterdam” is repeated twice.

7.B CONTROLLING OUTPUT STRUCTURE

As our model learns interpretable correspondence of each segment, it can control the output structures same as in Wiseman *et al.* (2018). Table 7.7 shows example generations by sampling diverse segment structures.

Input:	name the phoenix	eatype pub	food french	pricerange £20 - 25	customerrating high	area riverside
	familyfriendly yes	near crowne plaza hotel				
Output1:	the phoenix	pub is located in riverside	near crowne plaza hotel	.	it serves french food	in the £20 - 25 price range
		.	it has a high customer rating	.		
Output2:	the phoenix	is located in riverside	near crowne plaza hotel	.	it is a family - friendly	french pub with the price range of £20 - 25
		.	it has a high customer rating	.		
Output3:	located in riverside	near crowne plaza hotel	,	the phoenix	is a french pub	with a high customer rating and a price range of £20 - 25
		.	It is family - friendly	.		

Table 7.7: Example of generations with diverse structures.

THE previous chapters all inject latent variables between the input and output to improve the diversity of interpretability. This chapter shows that latent-variable models are also a powerful tool when we have no parallel input-output pairs. We focus again on the task of open-domain dialogue generation where neural network-based sequence-to-sequence (seq2seq) models strongly suffer from the low-diversity problem. As bland and generic utterances usually dominate the frequency distribution in our daily chitchat, avoiding them to generate more interesting responses requires complex data filtering, sampling techniques or modifying the training objective. In this chapter, we propose a new perspective to diversify dialogue generation by leveraging *non-conversational* text. Compared with bilateral conversations, non-conversational text are easier to obtain, more diverse and cover a much broader range of topics. We collect a large-scale non-conversational corpus from multi sources including forum comments, idioms and book snippets. We further present a training paradigm to effectively incorporate these text via iterative back translation. The resulting model is tested on two conversational datasets and is shown to produce significantly more diverse responses without sacrificing the relevance with context (Su *et al.*, 2020).

8.1 INTRODUCTION

Seq2seq models have achieved impressive success in a wide range of text generation tasks. In open-domain chitchat, however, people have found the model tends to strongly favor short, generic responses like “I don’t know” or “OK” (Vinyals and Le, 2015; Shen *et al.*, 2017c). The reason lies in the extreme one-to-many mapping relation between every context and its potential responses (Zhao *et al.*, 2017c; Su *et al.*, 2018). Generic utterances, which can be in theory paired with most context, usually dominate the frequency distribution in the dialogue training corpus and thereby pushes the model to blindly produce these safe, dull responses (Su *et al.*, 2019b).

Current solutions can be roughly categorized into two classes: (1) Modify the seq2seq itself to bias toward diverse responses (Li *et al.*, 2016a; Shen *et al.*, 2019b). However, the model is still trained on the *limited dialogue corpus* which restricts its power at covering broad topics in open-domain chitchat. (2) Augment the training corpus with extra information like structured world knowledge, personality or emotions (Li *et al.*, 2016b; Dinan *et al.*, 2019), which requires *costly human annotation*.

In this work, we argue that training only based on conversational corpus can greatly constrain the usability of an open-domain chatbot system since many topics

Conversational Text	
Context (Translation)	暗恋的人却不喜欢我 The one I have a crush on doesn't like me.
Response	摸摸头 Head pat.
Non-Conversational Text	
Forum Comments	暗恋这碗酒，谁喝都会醉啊 Crush is an alcoholic drink, whoever drinks it will get intoxicated.
Idiom	何必等待一个没有结果的等待 Why wait for a result without hope
Book Snippet	真诚的爱情之路永不会是平坦的 The course of true love never did run smooth (From <i>A Midsummer Night's Dream</i>)

Table 8.1: A daily dialogue and non-conversational text from three sources. The contents of non-conversational text can be potentially utilized to enrich the response generation.

are not easily available in the dialogue format. With this in mind, we explore a cheap way to diversify dialogue generation by utilizing large amounts of *non-conversational text*. Compared with bilateral conversations, non-conversational text covers a much broader range of topics, and can be easily obtained without further human annotation from multiple sources like forum comments, idioms and book snippets. More importantly, non-conversational text are usually *more interesting and contentful* as they are written to convey some specific personal opinions or introduce a new topic, unlike in daily conversations where people often *passively* reply to the last utterance. As can be seen in Table 8.1, the response from the daily conversation is a simple comfort of "Head pat". Non-conversational text, on the contrary, exhibit diverse styles ranging from casual wording to poetic statements, which we believe can be potentially utilized to enrich the response generation.

To do so, we collect a large-scale corpus containing over 1M non-conversational utterances from multiple sources. To effectively integrate these utterances, we borrow the back translation idea from unsupervised neural machine translation (Sennrich *et al.*, 2016a; Lample *et al.*, 2018b) and treat the collected utterances as unpaired responses. We first pre-train the forward and backward transduction model on the parallel conversational corpus. The forward and backward model are then iteratively tuned to find the optimal mapping relation between conversational context and non-conversational utterances (Cotterell and Kreutzer, 2018). By this means, the content of non-conversational utterances is gradually distilled into the dialogue generation model (Kim and Rush, 2016), enlarging the space of generated responses to cover not only the original dialogue corpus, but also the wide topics reflected in the non-conversational utterances.

We test our model on two popular Chinese conversational datasets weibo (Shang *et al.*, 2015) and douban (Wu *et al.*, 2017). We compare our model against retrieval-based systems, style-transfer methods and several seq2seq variants which also target the diversity of dialogue generation. Automatic and human evaluation show that our model significantly improves the responses' diversity both semantically and

syntactically without sacrificing the relevance with context, and is considered as most favorable judged by human evaluators ²⁴.

8.2 RELATED WORK

The tendency to produce generic responses has been a long-standing problem in seq2seq-based open-domain dialogue generation (Vinyals and Le, 2015; Li *et al.*, 2016a). Previous approaches to alleviate this issue can be grouped into two classes.

The first class resorts to modifying the seq2seq architecture itself. For example, Shen *et al.* (2018b); Zhang *et al.* (2018c) changes the training objective to mutual information maximization and rely on continuous approximations or policy gradient to circumvent the non-differentiable issue for text. Li *et al.* (2016d); Serban *et al.* (2017b) treat open-domain chitchat as a reinforcement learning problem and manually define some rewards to encourage long-term conversations. There is also research that utilizes latent variable sampling (Serban *et al.*, 2017c; Shen *et al.*, 2018c, 2019c), adversarial learning (Li *et al.*, 2017a; Su *et al.*, 2018), replaces the beam search decoding with a more diverse sampling strategy (Li *et al.*, 2016c; Holtzman *et al.*, 2020) or applies reranking to filter generic responses (Li *et al.*, 2016a; Wang *et al.*, 2017b). All of the above are still trained on the original dialogue corpus and thereby cannot generate out-of-scope topics.

The second class seeks to bring in extra information into existing corpus like structured knowledge (Zhao *et al.*, 2018b; Ghazvininejad *et al.*, 2018; Dinan *et al.*, 2019), personal information (Li *et al.*, 2016b; Zhang *et al.*, 2018b) or emotions (Shen *et al.*, 2017d; Zhou *et al.*, 2018). However, corpus with such annotations can be extremely costly to obtain and is usually limited to a specific domain with small data size. Some recent research started to do dialogue style transfer based on personal speeches or TV scripts (Niu and Bansal, 2018; Gao *et al.*, 2019; Su *et al.*, 2019a). Our motivation differs from them in that we aim at enriching general dialogue generation with abundant non-conversational text instead of being constrained on one specific type of style.

Back translation is widely used in unsupervised machine translation (Sennrich *et al.*, 2016a; Lample *et al.*, 2018a; Artetxe *et al.*, 2018) and has been recently extended to similar areas like style transfer (Subramanian *et al.*, 2019), summarization (Zhao *et al.*, 2019) and data-to-text (Chang *et al.*, 2020). To the best of our knowledge, it has never been applied to dialogue generation yet. Our work treats the context and non-conversational text as unpaired source-target data. The back-translation idea is naturally adopted to learn the mapping between them. The contents of non-conversational text can then be effectively utilized to enrich the dialogue generation.

Resources	Size	Avg. length
Comments	781,847	21.0
Idioms	51,948	18.7
Book Snippets	206,340	26.9

Table 8.2: Statistics of Non-Conversational Text.

8.3 DATASET

We would like to collect non-conversational utterances that stay close with daily-life topics and can be potentially used to augment the response space. The utterance should be neither too long nor too short, similar with our daily chitchats. Therefore, we collect data from the following three sources:

1. Forum comments. We collect comments from zhihu ²⁵, a popular Chinese forums. Selected comments are restricted to have more than 10 likes and less than 30 words ²⁶.
2. Idioms. We crawl idioms, famous quotes, proverbs and locutions from several websites. These phrases are normally highly-refined and graceful, which we believe might provide a useful augmentation for responses.
3. Book Snippets. We select top 1,000 favorite novels or prose from wechat read ²⁷. Snippets highlighted by readers, which are usually quintessential passages, and with the word length range 10-30 are kept.

We further filter out sentences with offensive or discriminative languages by phrase matching against a large blocklist. The resulting corpus contains over 1M utterances. The statistics from each source are listed in Table 8.2.

8.4 APPROACH

Let $\mathcal{D} = \{(X_1, Y_1), (X_2, Y_2), \dots, (X_N, Y_N)\}$ denote the parallel conversational corpus. X_i is the context and Y_i is the corresponding response. $\mathcal{D}_T = \{T_1, T_2, \dots, T_M\}$ denotes our collected corpus where T_i is a non-conversational utterance. As the standard seq2seq model trained only on \mathcal{D} tends to generate over-generic responses, our purpose is to diversify the generated responses by leveraging the non-conversational corpus \mathcal{D}_T , which are semantically and syntactically much richer than responses

²⁴Code and dataset available at <https://github.com/chin-gyou/Div-Non-Conv>

²⁵<https://www.zhihu.com>

²⁶The posts are usually very long, describing a specific social phenomenon or news event, so building parallel conversational corpus from post-comment pairs is difficult. Nonetheless, these high-liked comments are normally high-quality themselves and can be used to augment the response space.

²⁷<https://wereal.qq.com/>

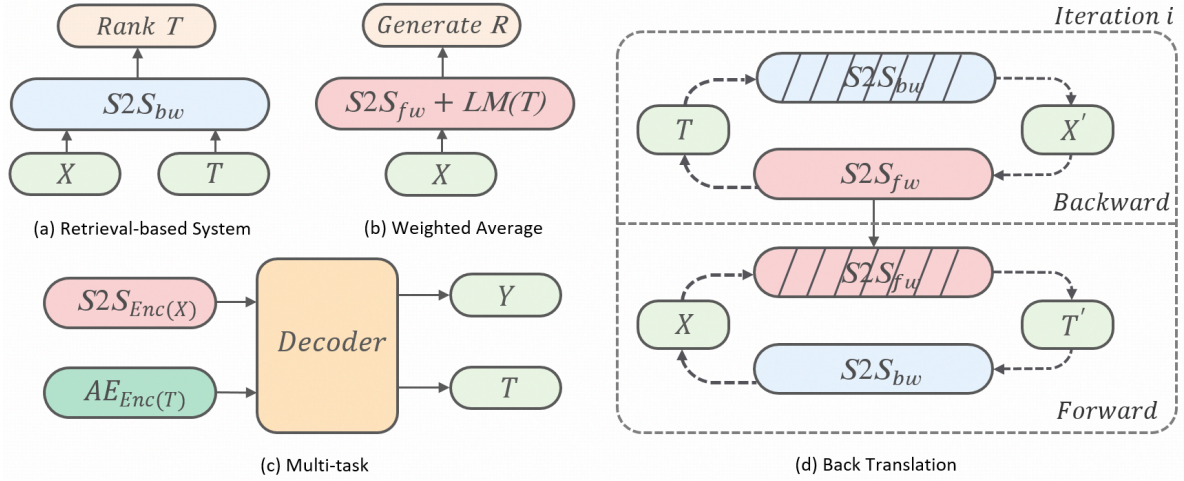


Figure 8.1: Comparison of four approaches leveraging the non-conversational text. $S2S_{fw}$, $S2S_{bw}$ and LM indicate the forward, backward seq2seq and language model respectively. (d) visualizes the process of one iteration for the back translation approach. Striped components are not updated in each iteration.

contained in \mathcal{D} . In the following section, we first go through several baseline systems, then introduce our proposed method based on back translation.

8.4.1 Retrieval-based System

The first approach we consider is a retrieval-based system that considers all sentences contained in \mathcal{D}_T as candidate responses. As the proportion of generic utterances in \mathcal{D}_T is much lower than that in \mathcal{D} , the diversity will be largely improved. Standard retrieval algorithms based on context-matching (Wu *et al.*, 2017; Bartl and Spanakis, 2017) fail to apply here since non-conversational text does not come with its corresponding context. Therefore, we train a backward seq2seq model on the parallel conversational corpus \mathcal{D} to maximize $p(X_i|Y_i)$. The score assigned by the backward model, which can be seen as an estimation of the point-wise mutual information, is used to rank the responses (Li *et al.*, 2016a) ²⁸.

The major limitation of the retrieval-based system is that it can only produce responses from a finite set of candidates. The model can work well only if an appropriate response already exists in the candidate bank. Nonetheless, due to the large size of the non-conversational corpus, this approach is a very strong baseline.

²⁸The backward seq2seq model measures the context relevance better than forward models since the latter highly biases generic utterances (Li *et al.*, 2016a; Zhang *et al.*, 2018c)

8.4.2 Weighted Average

The second approach is to take a weighted average score of a seq2seq model trained on \mathcal{D} and a language model trained on \mathcal{D}_T when decoding responses. The idea has been widely utilized on domain adaptation for text generation tasks (Koehn and Schroeder, 2007; Wang *et al.*, 2017b; Niu and Bansal, 2018). In our scenario, basically we hope the generated responses could share the diverse topics and styles of the non-conversational text, yet stay relevant with the dialogue context. The seq2seq model $S2S$ is trained on \mathcal{D} as an indicator of how relevant each response is with the context. A language model \mathcal{L} is trained on \mathcal{D}_T to measure how the response matches the domain of \mathcal{D}_T . The decoding probability for generating word w at time step t is assigned by:

$$p_t(w) = \alpha S2S_t(w) + (1 - \alpha) L_t(w) \quad (8.1)$$

where α is a hyperparameter to adjust the balance between the two. Setting $\alpha = 1$ will make it degenerate into the standard seq2seq model while $\alpha = 0$ will totally ignore the dialogue context.

8.4.3 Multi-task

The third approach is based on multi-task learning. A seq2seq model is trained on the parallel conversational corpus \mathcal{D} while an autoencoder model is trained on the non-parallel monologue data \mathcal{D}_T . Both models share the decoder parameters to facilitate each other. The idea was first experimented on machine translation in order to leverage large amounts of target-side monolingual text (Luong *et al.*, 2016; Sennrich *et al.*, 2016a). Luan *et al.* (2017) extended it to conversational models for speaker-role adaptation. The intuition is that by tying the decoder parameters, the seq2seq and autoencoder model can learn a shared latent space between the dialogue corpus and non-conversational text. When decoding, the model can generate responses with features from both sides.

8.4.4 Back Translation

Finally, we consider the back translation technique commonly used for unsupervised machine translation (Artetxe *et al.*, 2018; Lample *et al.*, 2018a). The basic idea is to first *initialize* the model properly to provide a good starting point, then iteratively perform *backward* and *forward* translation to learn the correspondence between context and unpaired non-conversational utterances.

Initialization. Unlike unsupervised machine translation, the source and target side in our case come from the same language, and we already have a parallel conversational corpus \mathcal{D} , so we can get rid of the careful embedding alignment and autoencoding steps as in Lample *et al.* (2018b). For the initialization, we simply train

a forward and backward seq2seq model on \mathcal{D} . The loss function is:

$$\mathbb{E}_{X_i, Y_i \sim \mathcal{D}} - \log P_f(Y_i|X_i) - \log P_b(X_i|Y_i) \quad (8.2)$$

where P_f and P_b are the decoding likelihood defined by the forward and backward seq2seq model respectively. We optimize Eq. 8.2 until convergence. Afterwards, the forward and backward seq2seq can learn the backbone mapping relation between a context and its response in a conversational structure.

Backward. After the initialization, we use the backward seq2seq to create pseudo parallel training examples from the non-conversational text \mathcal{D}_T . The forward seq2seq is then trained on the pseudo pairs. The objective is to minimize:

$$\begin{aligned} \mathbb{E}_{T_i \sim \mathcal{D}_T} - \log P_f(T_i|b(T_i)) \\ b(T_i) = \arg \max_u P_b(u|T_i) \end{aligned} \quad (8.3)$$

where we approximate the $\arg \max$ function by using a beam search decoder to decode from the backward model $P_b(u|T_i)$. Because of the non-differentiability of the $\arg \max$ operator, the gradient is only passed through P_f but not P_b ²⁹.

As P_b is already well initialized by training on the parallel corpus \mathcal{D} , the back-translated pseudo pair $\{b(T_i), T_i\}$ can roughly follow the typical human conversational patterns. Training P_f on top of them will encourage the forward decoder to generate utterances in the domain of T_i while maintaining coherent as a conversation.

Forward. The forward translation follows a similar step as back translation. The forward seq2seq P_f translates context into a response, which in return form a pseudo pair to train the backward model P_b . The objective is to minimize:

$$\begin{aligned} \mathbb{E}_{X_i \sim \mathcal{D}} - \log P_b(X_i|f(X_i)) \\ f(X_i) = \arg \max_v P_f(v|X_i) \end{aligned} \quad (8.4)$$

where the $\arg \max$ function is again approximated with a beam search decoder and the gradient is only backpropagated through P_b . Though X_i has its corresponding Y_i in \mathcal{D} , we drop Y_i and instead train on forward translated pseudo pairs $\{X_i, f(X_i)\}$. As P_f is trained by leveraging data from \mathcal{D}_T , $f(X_i)$ can have superior diversity compared with Y_i .

The encoder parameters are shared between the forward and backward models while decoders are separate. The backward and forward translation are iteratively performed to close the gap between P_f and P_b (Hoang *et al.*, 2018; Cotterell and Kreutzer, 2018). The effects of non-conversational text are strengthened after each iteration. Eventually, the forward model will be able to produce diverse responses covering the wide topics in \mathcal{D}_T . Algorithm 2 depicts the training process.

²⁹As also noted in Lample *et al.* (2018b), backpropagating further through P_b brings no improvement.

Algorithm 2 Model Training Process

Initialization: Train by minimizing Eq. 8.2 until convergence;
repeat
 Backward: Train by minimizing Eq. 8.3 until convergence
 Forward: Train by minimizing Eq. 8.4 until convergence
until Model converges

8.5 EXPERIMENTS

8.5.1 Datasets

We conduct our experiments on two Chinese dialogue corpus Weibo (Shang *et al.*, 2015) and Douban (Wu *et al.*, 2017). Weibo ³⁰ is a popular Twitter-like microblogging service in China, on which a user can post short messages, and other users make comment on a published post. The post-comment pairs are crawled as short-text conversations. Each utterance has 15.4 words on average and the data is split into train/valid/test subsets with 4M/40k/10k utterance pairs. Douban ³¹ is a Chinese social network service where people can chat about different topics online. The original data contains 1.1M multi-turn conversations. We split them into two-turn context-response pairs, resulting in 10M train, 500k valid and 100K test samples.

8.5.2 General Setup

For all models, we use a two-layer LSTM (Hochreiter and Schmidhuber, 1997) encoder/decoder structure with hidden size 500 and word embedding size 300. Models are trained with Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of 0.15. We set the batch size as 256 and use the gradients clipping of 5. We build out vocabulary with character-based segmentation for Chinese. For non-Chinese tokens, we simply split by space and keep all unique tokens that appear at least 5 times. Utterances are cut down to at most 50 tokens and fed to every batch. We implement our models based on the OpenNMT toolkit (Klein *et al.*, 2017) and other hyperparameters are set as the default values.

8.5.3 Compared Models

We compare our model with the standard seq2seq and four popular variants which were proposed to improve the diversity of generated utterances. All of them are trained only on the parallel conversational corpus:

Standard. The standard seq2seq with beam search decoding (size 5).

³⁰<http://www.weibo.com/>

³¹<https://www.douban.com/group>

MMI. The maximum mutual information decoding which reranks the decoded responses with a backward seq2seq model (Li *et al.*, 2016a). The hyperparameter λ is set to 0.5 as suggested. 200 candidates per context are sampled for re-ranking

Diverse Sampling. The diverse beam search strategy proposed in Vijayakumar *et al.* (2018) which explicitly controls for the exploration and exploitation of the search space. We set the number of groups as 5, $\lambda = 0.3$ and use the Hamming diversity as the penalty function as in the paper.

Nucleus Sampling. Proposed in Holtzman *et al.* (2020), it allows for diverse sequence generations. Instead of decoding with a fixed beam size, it samples text from the dynamic nucleus. We use the default configuration and set $p = 0.9$.

CVAE. The conditional variational autoencoder (Serban *et al.*, 2017c; Zhao *et al.*, 2017c) which injects diversity by imposing stochastic latent variables. We use a latent variable with dimension 100 and utilize the KL-annealing strategy with step 350k and a word drop-out rate of 0.3 to alleviate the posterior collapse problem (Bowman *et al.*, 2016).

Furthermore, we compare the 4 approaches mentioned in §8.4 which incorporate the collected non-conversational text:

Retrieval-based. (§8.4.1) Due to the large size of the non-conversational corpus, exact ranking is extremely slow. Therefore, we first retrieve top 200 matched text with elastic search based on the similarity of Bert embeddings (Devlin *et al.*, 2019b). Specifically, we pass sentences through Bert and derive a fixed-sized vector by averaging the outputs from the second-to-last layer (May *et al.*, 2019)³². The 200 candidates are then ranked with the backward score³³.

Weighted Average. (§8.4.2) We set $\lambda = 0.5$ in eq. 8.1, which considers context relevance and diversity with equal weights.

Multi-task. (§8.4.3) We concatenate each context-response pair with a non-conversational utterance and train with a mixed objective of seq2seq and autoencoding (by sharing the decoder).

Back Translation. (§8.4.4) We perform the iterative backward and forward translation 4 times for both datasets. We observe the forward cross entropy loss converges after 4 iterations.

³²<https://github.com/hanxiao/bert-as-service>

³³This makes it similar to MMI reranking, whose 200 candidates are from seq2seq decodings instead of top-matched non-conversational utterances.

Metrics Model	Weibo					Douban				
	BLEU-2	Dist-1	Dist-2	Ent-4	Adver	BLEU-2	Dist-1	Dist-2	Ent-4	Adver
Standard	0.0165	0.018	0.050	5.04	0.30	0.0285	0.071	0.206	7.55	0.19
MMI	0.0161	0.025	0.069	5.98	0.42	0.0263	0.143	0.363	7.60	0.31
Diverse	0.0175	0.019	0.054	6.20	0.38	0.0298	0.130	0.358	7.51	0.25
Nucleus	0.0183	0.027	0.074	7.41	0.43	0.0312	0.141	0.402	7.93	0.30
CVAE	0.0171	0.023	0.061	6.63	0.36	0.0287	0.169	0.496	7.80	0.29
Retrieval	0.0142	0.198	0.492	12.5	0.13	0.0276	0.203	0.510	13.3	0.17
Weighted	0.0152	0.091	0.316	9.26	0.22	0.0188	0.172	0.407	8.73	0.14
Multi	0.0142	0.128	0.348	8.98	0.27	0.0110	0.190	0.389	8.26	0.16
BT (Iter=1)	0.0180	0.046	0.171	7.64	0.19	0.0274	0.106	0.313	8.16	0.15
BT (Iter=4)	0.0176	0.175	0.487	11.2	0.35	0.0269	0.207	0.502	11.0	0.25
Human	-	0.171	0.452	9.23	0.88	-	0.209	0.514	11.3	0.85

Table 8.3: Automatic evaluation on Weibo and Douban datasets. Upper areas are models trained only on the conversational corpus. Middle areas are baseline models incorporating the non-conversational corpus. Bottom areas are our model with different number of iterations. Best results in every area are **bolded**.

8.6 RESULTS

As for the experiment results, we report the automatic and human evaluation in §8.6.1 and §8.6.2 respectively. Detailed analysis are shown in §8.6.3 to elaborate the differences among model performances and some case studies.

8.6.1 Automatic Evaluation

Evaluating dialogue generation is extremely difficult. Metrics which measure the word-level overlap like BLEU (Papineni *et al.*, 2002) have been widely used for dialogue evaluation. However, these metrics do not fit into our setting well as we would like to diversify the response generation with an external corpus, the generations will inevitably differ greatly from the ground-truth references in the original conversational corpus. Though we report the BLEU score anyway and list all the results in Table 8.3, it is worth mentioning that the BLEU score itself is by no means a reliable metric to measure the quality of dialogue generations.

Diversity. Diversity is a major concern for dialogue generation. Same as in (Li *et al.*, 2016a), we measure the diversity by the ratio of distinct unigrams (**Dist-1**) and bigrams (**Dist-2**) in all generated responses. As the ratio itself ignores the frequency distribution of n-grams, we further calculate the entropy value for the empirical distribution of n-grams (Zhang *et al.*, 2018c). A larger entropy indicates more diverse distributions. We report the entropy of four-grams (**Ent-4**) in Table 8.3. Among models trained only on the conversational corpus, the standard seq2seq performed worst as expected. All different variants improved the diversity more or less. Nucleus sampling and CVAE generated most diverse responses, especially

Nucleus who wins on 6 out of the 8 metrics. By incorporating the non-conversational corpus, the diversity of generated responses improves dramatically. The retrieval-based system and our model perform best, in most cases even better than human references. This can happen as we enrich the response generation with external resources. The diversity would be more than the original conversational corpus. Weighted-average and multi-task models are relatively worse, though still greatly outperforming models trained only on the conversational corpus. We can also observe that our model improves over standard seq2seq only a bit after one iteration. As more iterations are added, the diversity improves gradually.

Relevance. Measuring the context-response relevance automatically is tricky in our case. The typical way of using scores from forward or backward models as in Li and Jurafsky (2017) is not suitable as our model borrowed information from extra resources. The generated responses are out-of-scope for the seq2seq model trained on only on the conversational corpus and thus would be assigned very low scores. Apart from the BLEU-2 score, we further evaluate the relevance by leveraging an adversarial discriminator (Li *et al.*, 2017a). As has been shown in previous research, discriminative models are generally less biased to high-frequent utterances and more robust against their generative counterparts (Lu *et al.*, 2017a; Luo *et al.*, 2018). The discriminator is trained on the parallel conversational corpus distinguish correct responses from randomly sampled ones. We encode the context and response separately with two different LSTM neural networks and output a binary signal indicating relevant or not ³⁴. The relevance score is defined as the success rate that the model fools the adversarial classifier into believing its generations (**Adver** in Table 8.3). The retrieval-based model, who generates the most diverse generations, achieve the lowest score as for relevance with context. The restriction that it can only select from a set of fixed utterances do affect the relevance a lot ³⁵. Note that *the discriminator is also trained on the same bilateral conversational corpus, putting our model into a naturally disadvantageous place due to the incorporation of out-of-scope non-conversational text*. Nonetheless, our model still achieves competitive relevance score even compared with models trained only on the conversational corpus. This suggests our model does learn the proper patterns in human conversations instead of randomly synthesizing diverse generations.

8.6.2 Human Evaluation

Apart from automatic evaluations, we also employed crowdsourced judges to evaluate the quality of generations for 500 contexts of each dataset. We focus on evaluating the generated responses regarding the (1) relevance: if they coincide with the context

³⁴In our experiment, the discriminator performs reasonably well in the 4 scenarios outlined in Li *et al.* (2017a) and thus can be considered as a fair evaluation metric.

³⁵The fact that we only rank on 200 most similar utterances might also affect. We tried increasing the size to 1,000 but observe no tangible improvement. The candidate size required for a decent relevance score can be unbearably large.

Metrics Model	Weibo			Douban		
	Rel	Inter	Flu	Rel	Inter	Flu
STANDARD	0.32	0.11	0.76	0.26	0.13	0.82
NUCLEUS	0.46	0.19	0.78	0.38	0.21	0.83
RETRIEVAL	0.12	0.35	-	0.09	0.32	-
WEIGHTED	0.19	0.14	0.52	0.15	0.17	0.46
MULTI	0.25	0.21	0.70	0.22	0.23	0.66
BT (ITER=4)	0.43	0.37	0.77	0.39	0.48	0.80

Table 8.4: Human Evaluation Results

Context	一直单身怎么办 (Being always single, what should I do?)
Response	勇敢一点多去加好友啊 (Be brave and add more people to friends.)
Generation	[Iteration 0]: 不知道该怎么办 (I don't know what to do.)
	[Iteration 1]: 单身不可怕, 单身不可怕 (Being single is nothing, being single is nothing.)
	[Iteration 4]: 斯人若彩虹, 遇上方知有 (Every once in a while you find someone who's iridescent, and when you do, nothing will ever compare.)

Table 8.5: Example of response generation in different iterations.

(**Rel**), (2) interestingness: if they are interesting for people to continue the conversation (**Inter**) and (3) fluency: whether they are fluent by grammar (**Flu**)³⁶. Each sample gets one point if judged as yes and zero otherwise. Each pair is judged by three participants and the score supported by most people is adopted. The averaged scores are summarized in Table 8.4. We compare the standard seq2seq model, nucleus sampling which performs best among all seq2seq variants, and the four approaches leveraging the non-conversational text. All models perform decently well as for fluency except the weighted average one. The scores for diversity and relevance generally correlate well with the automatic evaluations. Overall the back-translation model are competitive with respect to fluency and relevance, while generating much more interesting responses to human evaluators. It also significantly outperforms the other three baseline approaches in its capability to properly make use of the non-conversational corpus.

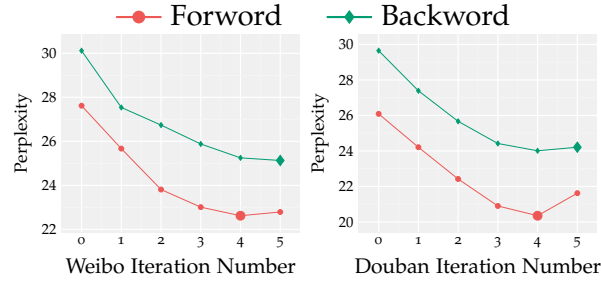


Figure 8.2: Change of validation loss across iterations.

8.6.3 Analysis

Effect of Iterative Training. To show the importance of the iterative training paradigm, we visualize the change of the validation loss in Figure 8.2³⁷. The forward validation loss is computed as the perplexity of the forward seq2seq on the pseudo context-response pairs obtained from the backward model, vice versa for backward loss. It approximately quantifies the KL divergence between them two (Kim and Rush, 2016; Cotterell and Kreutzer, 2018). As the iteration goes, the knowledge from the backward model is gradually distilled into the forward model. The divergence between them reaches the lowest point at iteration 4, where we stop our model. Table 8.5 further displays examples for different iterations. Iteration 0 generates mostly generic responses. Iteration 1 starts to become more diverse but still struggle with fluency and relevance. In the final iteration, it can learn to incorporate novel topics from the non-conversational text yet maintaining the relevance with context.

Diversity of Generation. We find the back translation model can generate *both semantically and syntactically* novel responses. Some examples are shown in Table 8.6. To find semantically novel responses, we segment them into phrases and find those containing novel phrases that do not exist on the conversational corpus. As in the first example of Table 8.6, the word 胖若两人 only exists in the non-conversational corpus. The model successfully learnt its semantic meaning and adopt it to generate novel responses. It is also common that the model learns frequent syntax structures from the non-conversational corpus. In the second example, it learnt the pattern of “To ... is easy, to ... is hard”, which appeared frequently in the non-conversational corpus, and utilized it to produce novel responses with the same structure. Note that both generations from the BT model *never appear exactly in the non-conversational corpus*. It must generate them by correctly understanding the meaning of the phrase components instead of memorizing the utterances verbally.

³⁶We do not evaluate the retrieval-based model for the fluency score as the retrieved utterances are fluent by construct.

³⁷Iteration 0 means before the iteration starts but after the initialization stage, equal to a standard seq2seq.

CXT	最近又长胖了 Fleshing out again recently.
NS	我也是这样的 Me too.
BT	哈哈莫非已经 胖若两人 了 hahaha already as fat as two people?
CXT	爱一个人真的不能跟她表白吗? Why loving someone but cannot confess?
NS	不一定的 Not necessarily.
BT	爱一个人 不难，难的是 放下一个人。 To love is easy, to give up is hard.

Table 8.6: Context (CXT), example generations from nucleus sampling (NS) and back-translation (BT). Novel words and syntax patterns are **highlighted**.

8.7 CONCLUSION

We propose a novel way of diversifying dialogue generation by leveraging non-conversational text. To do so, we collect a large-scale corpus from forum comments, idioms and book snippets. By training the model through iterative back translation, it is able to significantly improve the diversity of generated responses both semantically and syntactically. We compare it with several strong baselines and find it achieved the best overall performance. The model can be potentially improved by filtering the corpus according to different domains, or augmenting with a retrieve-and-rewrite mechanism, which we leave for future work.

DEEP learning has reshaped the research of natural language processing with GPU parallel computing and large-scale databases. The task of text generation, which stands at the core place of natural language processing and machine intelligence, has been well studied and boosted with the population of recurrent neural networks and transformers. Recently, large-scale self-supervised pre-training further pushes the model limits of text generation and reaches or even surpasses human performance in many tasks (Devlin *et al.*, 2019b). Nonetheless, text generation is still far from being a solved problem. Specifically, deep neural networks tend to generate text with low diversity and lack in controllability and interpretability due to their blackbox nature. They also require extensive supervised training data in order to perform competitively.

9.1 SUMMARY OF THESIS

In this thesis, we focus on addressing the above mentioned shortcomings of deep neural networks through an integration of latent-variable models. The text generation process is defined with a set of probabilistic latent variables, through which we could diversify, interpret or control the text to be generated. It can also provide us with a principled way of utilizing non-paired text, enabling efficient unsupervised or semi-supervised learning. The integration of latent variables complicates the training as the exact log-likelihood is usually intractable. Nonetheless, we show the training can be done efficiently with well-designed approximation techniques like top-k sampling, dynamic programming or variational inference. In detail, we conclude this thesis with the following three points:

- latent-variable models can effectively diversify text generation by imposing an additional sampling process. We present algorithms to achieve this without sacrificing other attributes of generated text, while adding negligible training overhead (Chapter 3, 4).
- By specifying the latent variable distribution to follow some human priors. It is able to automatically learn interpretable generation process. The text can be controlled via manipulating the latent variables (Chapter 5, 6, 7).
- Non-paired text can be leveraged to boost the text generation model by treating the missing correspondence as latent variables. The optimization can be effectively done as in the EM algorithm (Chapter 8).

9.2 CHALLENGES

However, latent-variable models still have many challenges when applied to text generation. A crucial unresolved challenge, and usually the core point of all latent variable applications, is the **training instability issue**. As we mentioned in Chapter 2, the model can easily be trapped in a local optimum where latent variables are simply ignored, such that the whole model degrades into the standard seq2seq. Many approaches have been proposed to alleviate this problem, but a principled solution is not yet demonstrated. Most of them revolve it through some heuristic-based techniques, like to weight the learning objective, add handcrafted regularizations or careful initialization. The choice of them is highly data and task dependent, requiring a lot of human intuitions and designs. For example, in our work of controllable content selection (Chapter 5), we adopt the mutual information constraint to force meaningful info encoded into the latent variable. However, the set of the mutual information value is non-trivial. We set it by running human annotations to get a rough estimation, though an ideal model should be able to automate this process. In practice, even with careful design and initialization, the model might still behave unexpectedly while learning latent variables contrary to human intuitions. From one point, as the variables to be learnt are latent, errors made in the training process tend to be magnified, and can never be corrected back by help from explicit supervisions. From the other point, machines, in their strengths and weaknesses, are far different from human minds. Annotations from human supervisions, which serve as the ground truth for the human world, are not necessarily optimal for machines. Therefore, we can often see the cases where latent variables increase interpretability yet sacrifice the performance, as interpretability in the human sense might not be the optimal solution for machines. Before we design our own latent-variable models, we should think beforehand what we want to learn in the latent variables and what our final goal is.

Do we expect the machines to perform similar to human minds to enable manual interpretation, or should we let machines learn their own way of thinking, to explore the most effective latent variables, even though uninterpretable to humans?

In the first case, proper supervision from humans is always helpful. We can use this signal to guide the latent variable learning in a semi-supervised setting. The second case is more challenging, as we are also uncertain about the format and meaning to be encoded into the latent variables. The current major solutions are intuition-based. We hope future works from the machine learning and natural language processing community come up with some principled, general-applicable metrics or algorithms to contribute to a more stable and robust latent-variable model.

Indeed, the recent wide applications of highly-complicated Transformer models further aggregate the challenge. The internal components of the neural network becomes more uninterpretable to humans. The representations at each position can be affected by its neighboring context, and each layer in the representation learn its

different features. Building an interpretable latent-variable model on top of them becomes increasingly infeasible.

How do we effectively incorporate latent-variable models with Transformers? Or even, is latent-variable model and interpretability really needed? If Transformers can achieve state-of-the-art performance in their own magic way by simply using more resources and training data, is interpretability still an indispensable feature we want?

Exploring the answers to these questions would be an interesting future direction. The value of latent-variable models is general and will clearly not fade away due to the more complex Transformer architectures, but studying the combination of them is by all means important. After all, latent-variable models should be validated in the most powerful modern neural architectures.

The last point we would like to highlight is the re-utilization challenge. All the works mentioned in the thesis are task-dependent. To enable controllable text generation for different requirement, e.g., sentiment, content, style, etc, we need to define different probabilistic distributions and latent process. A model trained for one cannot be utilized for other applications at all. This is a large burden in real-life applications. User requirement changes along with time, and there are numerous latent factors and combinations of them that we might want to control. Listing the complete requirement from the beginning, and devising a comprehensive latent-variable model is unrealistic. Therefore, it brings us a new topic that is worth studying:

how do we effectively utilize trained models to adapt to newly introduced latent variables?

Keskar *et al.* (2019) illustrate how to enable controllable generation from large-scaled pretrained language models in a lightweight way. However, their method only applies in a supervised way and all the variables are essentially not latent. They also only experimented with simple discrete controlling variables in a non-conditional setting, and did not shed light on how structured, finer-grained controlling can be fulfilled with conditional input. Re-utilizing pretrained model in practical applications has yet to be explored.

LIST OF FIGURES

1.1	Illustration of a typical encoder-decoder architecture for text generation. Figure taken from (Xie, 2017).	3
1.2	Graphical model of deep latent-variable model. θ is parameterized by deep neural networks	5
2.1	Illustration of optimization technique for latent-variable models	10
2.2	Graphical model of the pointer generator.	12
2.3	Graphical model of the HMM word alignment model. h_t^d is the hidden state of the decoder at time t which encapsulates all the history words y_1, \dots, y_{t-1} . Figure taken from (Wu <i>et al.</i> , 2018)	13
2.4	Illustration of the tree decoder for machine translation. Before translating to the target language, a syntactic tree is first generated, based on which the final output is produced from a tree-decoder. Figure taken from (Wang <i>et al.</i> , 2018b).	15
2.5	Graphical Model of unconditional Latent-variable Model	17
2.6	Inverse Autoregressive Flow. It consists of an initial sample z drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of z , each with a simple Jacobian determinants.	25
2.7	Illustration of Generative Adversarial Networks. The discriminator and the generator compete with each other. In the end, the generator will be able to generate real-like samples to fool the discriminator. . .	26
2.8	Asymmetry of KL divergence	32
2.9	Comparison of one-sided label smoothing and instance noise, p_Y denotes real data distribution and q_θ is the generated distribution . . .	35
2.10	Illustration of unrolled gan	36
2.11	Gradient of original GAN and Wasserstein GAN	37
2.12	Model structure of adversarially regularized autoencoder	40
2.13	Model structure of Professor forcing	41
2.14	Graphical Model of Back Translation	42
2.15	Effects of the temperature in Gumbel-softmax	44
2.16	Illustration of Reinforcement Learning	47
2.17	Change of KL divergence term	49
2.18	Effects of word keep rate. The KL divergence grows as fewer words are kept since the model must reply on more informative z to reconstruct the text.	50
2.19	Effects of additional loss	51
3.1	Architecture for collaborative variational encoder-decoder. \oplus denotes concatenation of information. $M_1(AE)$ and $M_2(CVAE)$ are represented in brown and green respectively.	58

3.2	Up: adversarial encoder-decoder, down: adversarial encoder-decoder after replacing GAN with VAE, Full line rectangle: GAN and CVAE phase, Dotted line rectangle: AE phase	59
3.3	T-SNE visualization of sampled latent variables. left: VHRED, right: CO+SS. Red dots correspond to samples from prior distribution, while the blue dots correspond to samples from posterior distribution. Viewable in color mode only.	64
4.1	A conversation in real life	69
4.2	Framework of NEXUS Networks. Full line indicates the generative model to generate the continuous code and corresponding responses. Dashed line indicates the inference model where the posterior code is trained to infer the history, current and future utterances. Both parts are simultaneously trained by gradient descent.	72
4.3	Effect of hyperparameter ratio λ_1/λ_2 on two datasets.	84
5.1	Model will select contents based on $\mathbf{B}(\gamma)$, then decode with $p_\theta(Y X, \beta)$. Source-text pairs are available for training, but the ground-truth content selection for each pair is unknown.	88
5.2	Negative log likelihood (NLL), selection entropy and self-BLEU as ϵ changes. NLL and self-bleu on Wikibio are added by 1 for better visualization. Lower NLL suggests higher performance. Higher entropy/self-BLEU means higher diversity/controllability.	98
5.3	Text generation examples from Gigaword. Highlighted words are selected. t1-3 are sampled from the decoder based on the selected content. Generations from VRS are more faithful to selected contents.	99
6.1	Alignment visualization of our model when decoding "closes". Posterior alignment is more accurate for model interpretation. In contrast, the prior alignment probability is spared to "announced" and "closure", which can be manually controlled to generate desired summaries. Decoded samples are shown when aligned to "announced" and "closure" respectively. Highlighted source words are those that can be directly aligned to a target token in the gold summary.	102
6.2	Architecture of the generalized pointer. The same encoder is applied to encode the source and target. When decoding "closes", we first find top-k source positions with the most similar encoded state. For each position, the decoding probability is computed by adding its word embedding and a predicted relation embedding.	105
6.3	Test perplexity when increasing k	109
6.4	Pointing Ratio of the standard pointer generator and GPG (evaluated on the test data). GPG enables the point mode more often, but quite a few pointed tokens are edited rather than simply copied.	111
6.5	Examples of summaries produced by GPG. Each two samples from CNN/DM, Gigaword and XSum (up to down). bold denotes novel words and their pointed source tokens. Bracketed numbers are the pointing probability $(1 - p_{gen})$ during decoding.	112

6.6	Examples of generated summaries. Examples are taken from CNN/DM, Gigaword and XSum (from up to down). Darker means higher pointing probability.	115
7.1	Generation from our model on the E2E dataset. Decoding is performed segment-by-segment. Each segment realizes one data record. ①~⑧ mark the decision order in the generation process.	118
7.2	Generation process of our approach. Segment end symbol \$ is ignored when updating the state of the decoder. <i>Solid arrows</i> indicate the transition model and <i>dashed arrows</i> indicate the generation model. Every segment s_o is generated by attending only to the corresponding data record $c(s_o)$	121
7.3	Average expected number of segments with varying hyperparameters. x-axis is the encoder/decoder hidden size and y-axis is the word embedding size. Upper two figures are without the granularity regularization and the bottom two are with regularization.	127
7.4	Example generations from WebNLG. Relation types are <u>underlined</u> and repeated generations are bolded . Segments and corresponding records in our model are marked in the same color. By adding explicit constraints to the decoding process, repetition and missing issues can be largely reduced. (better viewed in color)	128
8.1	Comparison of four approaches leveraging the non-conversational text. $S2S_{fw}$, $S2S_{bw}$ and LM indicate the forward, backward seq2seq and language model respectively. (d) visualizes the process of one iteration for the back translation approach. Striped component are not updated in each iteration. .	139
8.2	Change of validation loss across iterations.	147

LIST OF TABLES

Tab. 2.1	Sentences generated by GAN with soft relaxation	45
Tab. 3.1	Metric Results, left: Dailydialog, right: switchboard	63
Tab. 3.2	Embedding Results, left: dailydialog, right: switchboard	64
Tab. 3.3	Examples of context-response pairs for the neural network models. <code>__eou__</code> denotes end-of-utterance and indicates the start of a new turn.	65
Tab. 3.4	Human Judgements for models trained on Dailydialog corpus, F refers to fluent, C refers to coherence and D refers to diversity. .	66
Tab. 4.1	Results of embedding-based metrics. * indicates statistically significant difference ($p < 0.05$) from the best baselines. The same mark is used in Table 4.2	76
Tab. 4.2	Results of BLEU score. It is computed based on the smooth BLEU algorithm (Lin and Och, 2004). p-value interval is computed base on the altered bootstrap resampling algorithm (Riezler and Maxwell, 2005)	78
Tab. 4.3	Coherence, diversity and human evaluations. Left: DailyDialog results, right: Twitter results	79
Tab. 4.4	Examples of context-response pairs. <code>__eou__</code> denotes end-of-utterance. First three rows are from DailyDialog and the last two rows are from Twitter	81
Tab. 5.1	Headline generation examples from our model. We can generate text describing various contents by sampling different content selections. The selected source word and its corresponding realizations in the text are highlighted with the same color.	86
Tab. 5.2	Diversity of content selection. The % effect of selector is defined as the ratio of unique generation and mask, which reflects the rate that changing the selector will lead to corresponding changes of the generated text.	93
Tab. 5.3	Self-Bleu score by fixing selection mask. Higher means better controllability of content selection	95
Tab. 5.4	Human evaluation on fluency, intra-consistency and inter-diversity of content selection on DUC 2004.	96
Tab. 5.5	Gigaword best-select results. Larger ϵ leads to more controllable selector with a bit degrade of performance. (-post) means selecting from the posterior $q_\phi(\beta X, Y)$, (-pri) is from the prior $\mathbf{B}(\gamma_i)$	96
Tab. 5.6	Wikibio best-select results.	97

Tab. 6.1	ROUGE score on CNN/Dailymail. * marks results from See <i>et al.</i> (2017), and [†] from Gehrmann <i>et al.</i> (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$	109
Tab. 6.2	ROUGE score on Gigaword. * marks results from the word-based seq2seq implementation of Zhou <i>et al.</i> (2017), and [†] from Li <i>et al.</i> (2017b).	110
Tab. 6.3	ROUGE score on XSum. * marks results from Narayan <i>et al.</i> (2018). Underlined values are significantly better than Point.Gen. with $p = 0.05$	110
Tab. 6.4	Proportion of novel n-grams (NN-1,2,3) and sentences (NN-S) on generated summaries. GPG generate more novel words compared with standard pointer generators, though still slightly lower than seq2seq.	110
Tab. 6.5	Human evaluation results on DUC 2004. 0/1 is the score for the 0/1 Turing test.	113
Tab. 6.6	Word Alignment Precision on DUC 2004. Number in bracket is the posterior alignment precision.	113
Tab. 7.1	Segmentation with various granularities. 1 is too fine-grained while 4 is too coarse. We expect a segmentation like 2 or 3 to better control the generation.	123
Tab. 7.2	Automatic and human evaluation results on E2E dataset. SLUG , DANGNT , TUDA and N_TEMP are from previous works and the other models are our own implementations.	126
Tab. 7.3	Automatic and human evaluation results on WebNLG dataset. MELBOURNE and UPFUPF-FORGE are from previous works and the other models are our own implementations.	126
Tab. 7.4	(E2E) Attention map when decoding the word "low" in the PG model and "moderate" in our model. Hallucinated contents are bolded . The PG model wrongly attended to other slots thereby "hallucinated" the content of "low-priced". Our model always attends to one single slot instead of averaging over the whole inputs, the chance of hallucination is largely reduced.	129
Tab. 7.5	Example of missing in E2E. The "familyfriendly" is wigned to the period symbol.	131
Tab. 7.6	(E2E) Example of repeatition in WebNLG. The phrase "amsterdam-centrum is part of amsterdam" is repeated twice.	132
Tab. 7.7	Example of generations with diverse structures.	133
Tab. 8.1	A daily dialogue and non-conversational text from three sources. The contents of non-conversational text can be potentially utilized to enrich the response generation.	136
Tab. 8.2	Statistics of Non-Conversational Text.	138

Tab. 8.3	Automatic evaluation on Weibo and Douban datasets. Upper areas are models trained only on the conversational corpus. Middle areas are baseline models incorporating the non-conversational corpus. Bottom areas are our model with different number of iterations. Best results in every area are bolded	144
Tab. 8.4	Human Evaluation Results	146
Tab. 8.5	Example of response generation in different iterations.	146
Tab. 8.6	Context (CXT), example generations from nucleus sampling (NS) and back-translation (BT). Novel words and syntax patterns are highlighted .148	

BIBLIOGRAPHY

- M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng (2016). TensorFlow: A System for Large-Scale Machine Learning, in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation 2016*. Cited on page 62.
- R. Aharoni and Y. Goldberg (2017). Morphological Inflection Generation with Hard Monotonic Attention, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2017*. Cited on page 107.
- A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy (2018). Fixing a Broken ELBO, in *Proceedings of the 35th International Conference on Machine Learning 2018*. Cited on pages 5, 87, and 91.
- C. Allen and T. Hospedales (2019). Analogies Explained: Towards Understanding Word Embeddings, *International Conference on Machine Learning (ICML)*. Cited on page 105.
- G. Angeli, P. Liang, and D. Klein (2010). A simple domain-independent probabilistic approach to generation, in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing 2010*. Cited on pages 5, 117, 118, and 119.
- M. Arjovsky and L. Bottou (2017). Towards principled methods for training generative adversarial networks, *International Conference on Learning Representations(ICLR)*. Cited on page 31.
- M. Arjovsky, S. Chintala, and L. Bottou (2017). Wasserstein generative adversarial networks, in *International Conference on Machine Learning (ICLR) 2017*. Cited on pages 36 and 37.
- M. Artetxe, G. Labaka, E. Agirre, and K. Cho (2018). Unsupervised neural machine translation, *International Conference on Learning Representations(ICLR)*. Cited on pages 137 and 140.
- M. Backes, P. Berrang, M. Humbert, X. Shen, and V. Wolf (2018). Simulating the large-scale erosion of genomic privacy over time, *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 15(5), pp. 1405–1412. Cited on page 107.
- D. Bahdanau, K. Cho, and Y. Bengio (2015). Neural machine translation by jointly learning to align and translate, *International Conference on Learning Representations(ICLR)*. Cited on pages 2, 85, 87, 103, 107, and 120.

- A. Balakrishnan, J. Rao, K. Upasani, M. White, and R. Subba (2019). Constrained Decoding for Neural NLG from Compositional Representations in Task-Oriented Dialogue, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics 2019*. Cited on pages 118 and 123.
- S. Banerjee and A. Lavie (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization 2005*. Cited on page 125.
- C. Bannard and C. Callison-Burch (2005). Paraphrasing with bilingual parallel corpora, in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics 2005*. Cited on page 1.
- I. Barany, V. Vu, *et al.* (2007). Central limit theorems for Gaussian polytopes, *The Annals of Probability*, vol. 35(4), pp. 1593–1621. Cited on page 77.
- A. Bartl and G. Spanakis (2017). A retrieval-based dialogue system utilizing utterance and context embeddings, in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) 2017*. Cited on page 139.
- R. Barzilay and M. Lapata (2005). Collective content selection for concept-to-text generation, in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing 2005*. Cited on page 119.
- N. J. Beaudry and R. Renner (2012). An intuitive proof of the data processing inequality, *Quantum Information & Computation*, vol. 12(5-6), pp. 432–441. Cited on page 73.
- M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, D. Hjelm, and A. Courville (2018). Mutual Information Neural Estimation, in *Proceedings of the 35th International Conference on Machine Learning 2018*. Cited on page 84.
- A. Belz (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models, *Natural Language Engineering*, vol. 14(4), pp. 431–455. Cited on page 117.
- A. Belz and E. Reiter (2006). Comparing automatic and human evaluation of NLG systems, in *11th Conference of the European Chapter of the Association for Computational Linguistics 2006*. Cited on page 125.
- S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer (2015). Scheduled sampling for sequence prediction with recurrent neural networks, in *Advances in Neural Information Processing Systems 2015*. Cited on page 60.
- Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin (2003). A neural probabilistic language model, *Journal of machine learning research*, vol. 3(Feb), pp. 1137–1155. Cited on page 53.

- D. Berthelot, T. Schumm, and L. Metz (2017). Began: Boundary equilibrium generative adversarial networks, *arXiv preprint arXiv:1703.10717*. Cited on page 38.
- D. M. Blei, A. Y. Ng, and M. I. Jordan (2003). Latent dirichlet allocation, *Journal of machine Learning research*, vol. 3(Jan), pp. 993–1022. Cited on page 4.
- A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko (2013). Translating embeddings for modeling multi-relational data, in *Advances in neural information processing systems 2013*. Cited on page 105.
- S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio (2016). Generating Sentences from a Continuous Space, in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning 2016*. Cited on pages 4, 5, 47, 49, 50, 53, 57, 62, 72, 75, and 143.
- D. Britz, M. Guan, and M.-T. Luong (2017). Efficient Attention using a Fixed-Size Memory Representation, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on page 106.
- P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer (1993). The mathematics of statistical machine translation: Parameter estimation, *Computational linguistics*, vol. 19(2), pp. 263–311. Cited on pages 4 and 122.
- Y. Burda, R. B. Grosse, and R. Salakhutdinov (2015). Importance Weighted Autoencoders, *CoRR*, vol. abs/1509.00519. Cited on pages 73, 84, and 91.
- Z. Cao, F. Wei, W. Li, and S. Li (2018). Faithful to the original: Fact aware neural abstractive summarization, in *Thirty-Second AAAI Conference on Artificial Intelligence 2018*. Cited on page 112.
- E. Chang, D. I. Adelani, X. Shen, and V. Demberg (2020). Unsupervised Pidgin Text Generation By Pivoting English Data and Self-Training, *International Conference on Learning Representation workshop on African NLP*. Cited on pages 117 and 137.
- M. Chen, G. Lampouras, and A. Vlachos (2018). Sheffield at E2E: structured prediction approaches to end-to-end language generation, *arxiv*. Cited on page 124.
- S. F. Chen and J. Goodman (1996). An empirical study of smoothing techniques for language modeling, in *Proceedings of the 34th annual meeting on Association for Computational Linguistics 1996*. Cited on page 2.
- X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in *Advances in Neural Information Processing Systems 2016*. Cited on pages 39 and 73.

- X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel (2017). Variational Lossy Autoencoder, *International Conference on Learning Representations(ICLR)*. Cited on pages 5, 52, 55, 57, and 72.
- Y.-C. Chen and M. Bansal (2018). Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2018*. Cited on pages 86, 89, and 92.
- R. Child, S. Gray, A. Radford, and I. Sutskever (2019). Generating long sequences with sparse transformers, *arXiv preprint arXiv:1904.10509*. Cited on page 3.
- C.-C. Chiu and C. Raffel (2018). Monotonic chunkwise attention, *International Conference on Learning Representations(ICLR)*. Cited on page 124.
- K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*. Cited on page 69.
- J. Clarke and M. Lapata (2010). Discourse constraints for document compression, *Computational Linguistics*, vol. 36(3), pp. 411–441. Cited on page 1.
- R. Cotterell and J. Kreutzer (2018). Explaining and Generalizing Back-Translation through Wake-Sleep, *arXiv preprint arXiv:1806.04402*. Cited on pages 43, 136, 141, and 147.
- A. Creswell and A. A. Bharath (2016). Task specific adversarial cost function, *arXiv preprint arXiv:1609.08661*. Cited on page 29.
- R. Dale, I. Anisimoff, and G. Narroway (2012). HOO 2012: A report on the preposition and determiner error correction shared task, in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP 2012*. Cited on page 1.
- H. Daumé III and D. Marcu (2005). Induction of word and phrase alignments for automatic document summarization, *Computational Linguistics*, vol. 31(4), pp. 505–530. Cited on pages 5, 107, 113, and 118.
- A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39(1), pp. 1–22. Cited on page 10.
- Y. Deng, A. Bakhtin, M. Ott, A. Szlam, and M. Ranzato (2020). Residual Energy-Based Models for Text Generation, in *International Conference on Learning Representations 2020*. Cited on page 3.
- Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush (2017). Image-to-markup generation with coarse-to-fine attention, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70 2017*. Cited on pages 106 and 118.

- Y. Deng, Y. Kim, J. Chiu, D. Guo, and A. Rush (2018). Latent alignment and variational attention, in *Advances in Neural Information Processing Systems 2018*. Cited on pages 4, 9, 89, 102, 107, 118, and 119.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova (2019a). Bert: Pre-training of deep bidirectional transformers for language understanding, *The North American Chapter of the Association for Computational Linguistics*. Cited on page 87.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova (2019b). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) 2019*. Cited on pages 143 and 149.
- E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston (2019). Wizard of wikipedia: Knowledge-powered conversational agents, *International Conference on Learning Representations(ICLR)*. Cited on pages 135 and 137.
- L. Dinh, D. Krueger, and Y. Bengio (2014). NICE: Non-linear independent components estimation, *International Conference on Learning Representation, Big Learn workshop*. Cited on page 23.
- J. Donahue, P. Krähenbühl, and T. Darrell (2017). Adversarial feature learning, *International Conference on Learning Representations(ICLR)*. Cited on page 39.
- M. D. Donsker and S. S. Varadhan (1983). Asymptotic evaluation of certain Markov process expectations for large time. IV, *Communications on Pure and Applied Mathematics*, vol. 36(2), pp. 183–212. Cited on page 84.
- A. Dosovitskiy and T. Brox (2016). Generating images with perceptual similarity metrics based on deep networks, in *Advances in Neural Information Processing Systems 2016*. Cited on page 39.
- V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville (2017). Adversarially learned inference, *International Conference on Learning Representations(ICLR)*. Cited on page 39.
- O. Dušek, J. Novikova, and V. Rieser (2018). Findings of the E2E NLG Challenge, in *Proceedings of the 11th International Conference on Natural Language Generation 2018*. Cited on page 117.
- O. Dušek, J. Novikova, and V. Rieser (2020). Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge, *Computer Speech & Language*, vol. 59, pp. 123–156. Cited on pages 3, 117, and 126.
- J. Eisner (2002). Parameter estimation for probabilistic finite-state transducers, in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics 2002*. Cited on page 124.

- J. Eisner (2016). Inside-outside and forward-backward algorithms are just backprop (tutorial paper), in *Proceedings of the Workshop on Structured Prediction for NLP 2016*. Cited on pages 11 and 123.
- H. Elder, J. Foster, J. Barry, and A. OConnor (2019). Designing a Symbolic Intermediate Representation for Neural Surface Realization, in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation 2019*. Cited on page 15.
- A. Fan, D. Grangier, and M. Auli (2018a). Controllable Abstractive Summarization, in *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation 2018*. Cited on page 92.
- A. Fan, D. Grangier, and M. Auli (2018b). Controllable Abstractive Summarization, in *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation 2018*. Cited on page 103.
- W. Fedus, I. Goodfellow, and A. M. Dai (2018). MaskGAN: Better Text Generation via Filling in the _____, in *International Conference on Learning Representations 2018*. Cited on page 3.
- T. C. Ferreira, C. van der Lee, E. van Miltenburg, and E. Krahmer (2019). Neural data-to-text generation: A comparison between pipeline and end-to-end architectures, *Conference on Empirical Methods in Natural Language Processing*. Cited on page 119.
- J. L. Fleiss (1971). Measuring nominal scale agreement among many raters., *Psychological bulletin*, vol. 76(5), p. 378. Cited on page 80.
- B. J. Frey (1998). *Graphical models for machine learning and digital communication*, MIT press. Cited on page 4.
- B. J. Frey, G. E. Hinton, and a. P. Dayan (1996). Does the wake-sleep algorithm produce good density estimators?, in *Advances in neural information processing systems 1996*. Cited on page 4.
- K. Ganchev, J. Gillenwater, B. Taskar, *et al.* (2010). Posterior regularization for structured latent variable models, *Journal of Machine Learning Research*, vol. 11(Jul), pp. 2001–2049. Cited on pages 118 and 124.
- X. Gao, Y. Zhang, S. Lee, M. Galley, C. Brockett, J. Gao, and B. Dolan (2019). Structuring Latent Spaces for Stylized Response Generation, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) 2019*. Cited on page 137.
- C. Gardent, A. Shimorina, S. Narayan, and L. Perez-Beltrachini (2017). The webnlg challenge: Generating text from rdf data, in *Proceedings of the 10th International Conference on Natural Language Generation 2017*. Cited on page 125.

- J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin (2017). Convolutional Sequence to Sequence Learning, in *International Conference on Machine Learning 2017*. Cited on page 103.
- S. Gehrmann, Y. Deng, and A. Rush (2018). Bottom-Up Abstractive Summarization, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 86, 87, 88, 92, 108, 109, and 158.
- M. Germain, K. Gregor, I. Murray, and H. Larochelle (2015). MADE: masked autoencoder for distribution estimation, in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15) 2015*. Cited on page 24.
- H. Ghader and C. Monz (2017). What does Attention in Neural Machine Translation Pay Attention to?, in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers) 2017*. Cited on page 113.
- M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley (2018). A knowledge-grounded neural conversation model, in *Thirty-Second AAAI Conference on Artificial Intelligence 2018*. Cited on page 137.
- P. W. Glynn (1990). Likelihood Ratio Gradient Estimation for Stochastic Systems, *Commun. ACM*, vol. 33(10), pp. 75–84. Cited on page 89.
- J. Godfrey and E. Holliman (). SWITCHBOARD-1 Release 2, 1997, *Linguistic Data Consortium, Philadelphia*. Cited on page 61.
- E. Goldberg, N. Driedger, and R. I. Kittredge (1994). Using natural-language processing to produce weather forecasts, *IEEE Expert*, vol. 9(2), pp. 45–53. Cited on page 1.
- I. Goodfellow (2016). NIPS 2016 tutorial: Generative adversarial networks, *Advances in neural information processing systems tutorial*. Cited on pages 4 and 31.
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial nets, in *Advances in neural information processing systems 2014*. Cited on pages 25, 27, 28, and 59.
- W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud (2018). Backpropagation through the Void: Optimizing control variates for black-box gradient estimation, in *International Conference on Learning Representations 2018*. Cited on page 90.
- A. Graves (2012). Sequence Transduction with Recurrent Neural Networks, *CoRR*, vol. abs/1211.3711. Cited on pages 62, 77, and 108.
- J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher (2018a). Non-Autoregressive Neural Machine Translation, in *International Conference on Learning Representations 2018*. Cited on page 3.

- J. Gu, D. J. Im, and V. O. Li (2018b). Neural machine translation with gumbel-greedy decoding, *Association for the Advancement of Artificial Intelligence*, pp. 5125–5132. Cited on page 70.
- J. Gu, Z. Lu, H. Li, and V. O. Li (2016). Incorporating Copying Mechanism in Sequence-to-Sequence Learning, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2016*. Cited on pages 3, 11, 101, 104, and 107.
- C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio (2016). Pointing the Unknown Words, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2016*. Cited on pages 3 and 104.
- I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville (2017). Improved training of wasserstein gans, *Advances in Neural Information Processing Systems*. Cited on page 38.
- J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang (2017). Long Text Generation via Adversarial Training with Leaked Information, *Association for the Advancement of Artificial Intelligence*. Cited on page 46.
- H. Hakami, K. Hayashi, and D. Bollegala (2018). Why does PairDiff work?-A Mathematical Analysis of Bilinear Relational Compositional Operators for Analogy Detection, in *Proceedings of the 27th International Conference on Computational Linguistics 2018*. Cited on page 105.
- D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma (2016a). Dual learning for machine translation, in *Advances in Neural Information Processing Systems 2016*. Cited on page 70.
- J. He, X. Wang, G. Neubig, and T. Berg-Kirkpatrick (2020). A Probabilistic Formulation of Unsupervised Text Style Transfer, in *International Conference on Learning Representations 2020*. Cited on pages 4 and 43.
- K. He, X. Zhang, S. Ren, and J. Sun (2016b). Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition 2016*. Cited on page 3.
- X. He, G. Haffari, and M. Norouzi (2018). Sequence to Sequence Mixture Model for Diverse Machine Translation, in *Proceedings of the 22nd Conference on Computational Natural Language Learning 2018*. Cited on page 12.
- K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom (2015). Teaching machines to read and comprehend, in *Advances in Neural Information Processing Systems 2015*. Cited on pages 103 and 107.
- I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner (2017). beta-VAE: Learning basic visual concepts with a constrained

- variational framework, in *In Proceedings of the International Conference on Learning Representations (ICLR) 2017*. Cited on page 62.
- G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal (1995). The "wake-sleep" algorithm for unsupervised neural networks, *Science*, vol. 268(5214), p. 1158. Cited on pages 18, 43, and 56.
- R. D. Hjelm, A. P. Jacob, T. Che, K. Cho, and Y. Bengio (2017). Boundary-Seeking Generative Adversarial Networks, *arXiv preprint arXiv:1702.08431*. Cited on page 45.
- V. C. D. Hoang, P. Koehn, G. Haffari, and T. Cohn (2018). Iterative Back-Translation for Neural Machine Translation, in *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation 2018*. Cited on pages 42 and 141.
- S. Hochreiter and J. Schmidhuber (1997). Long Short-Term Memory, *Neural Computation*, vol. 9(8), pp. 1735–1780. Cited on pages 3 and 142.
- A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi (2020). The Curious Case of Neural Text Degeneration, in *International Conference on Learning Representations 2020*. Cited on pages 3, 137, and 143.
- W.-T. Hsu, C.-K. Lin, M.-Y. Lee, K. Min, J. Tang, and M. Sun (2018). A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2018*. Cited on page 92.
- Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing (2017). Controllable Text Generation, *International Conference on Machine Learning*. Cited on page 43.
- Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing (2018). On Unifying Deep Generative Models, *International Conference on Learning Representation*. Cited on page 39.
- P.-S. Huang, C. Wang, S. Huang, D. Zhou, and L. Deng (2018). Towards neural phrase-based machine translation, *International Conference on Learning Representations (ICLR)*. Cited on page 119.
- F. Huszár (2015). How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?, *arXiv preprint arXiv:1511.05101*. Cited on page 29.
- F. Huszár (2017). Variational Inference using Implicit Distributions, *arXiv preprint arXiv:1702.08235*. Cited on page 38.
- H. Inan, K. Khosravi, and R. Socher (2017). Tying word vectors and word classifiers: A loss framework for language modeling, *International Conference on Learning Representations (ICLR)*. Cited on pages 3 and 104.
- R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton (1991). Adaptive mixtures of local experts, *Neural computation*, vol. 3(1), pp. 79–87. Cited on page 11.

- S. Jain and B. C. Wallace (2019). Attention is not Explanation, *The North American Chapter of the Association for Computational Linguistics*. Cited on page 107.
- E. Jang, S. Gu, and B. Poole (2017). Categorical reparameterization with gumbel-softmax, *International Conference on Learning Representations(ICLR)*. Cited on pages 43, 70, and 71.
- M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul (1999). An introduction to variational methods for graphical models, *Machine learning*, vol. 37(2), pp. 183–233. Cited on pages 16, 55, and 74.
- J. Juraska, P. Karagiannis, K. Bowden, and M. Walker (2018). A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) 2018*. Cited on page 127.
- L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer (2018). Fast Decoding in Sequence Models Using Discrete Latent Variables, in *Proceedings of the 35th International Conference on Machine Learning 2018*. Cited on pages 71 and 92.
- K. Kawakami, C. Dyer, and P. Blunsom (2019). Unsupervised Word Discovery with Segmental Neural Language Models, *Annual Meeting of the Association for Computational Linguistics*. Cited on page 119.
- N. R. Ke, A. G. A. P. GOYAL, O. Bilaniuk, J. Binas, M. C. Mozer, C. Pal, and Y. Bengio (2018a). Sparse attentive backtracking: Temporal credit assignment through reminding, in *Advances in Neural Information Processing Systems 2018*. Cited on page 106.
- N. R. Ke, K. Żołna, A. Sordoni, Z. Lin, A. Trischler, Y. Bengio, J. Pineau, L. Charlin, and C. Pal (2018b). Focused Hierarchical RNNs for Conditional Sequence Processing, in *Proceedings of the 35th International Conference on Machine Learning 2018*. Cited on page 91.
- N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher (2019). Ctrl: A conditional transformer language model for controllable generation, *arXiv preprint arXiv:1909.05858*. Cited on page 151.
- J. Kim and R. J. Mooney (2010). Generative alignment and semantic parsing for learning from ambiguous supervision, in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters 2010*. Cited on page 117.
- Y. Kim, C. Denton, L. Hoang, and A. M. Rush (2017). Structured attention networks, *International Conference on Learning Representations(ICLR)*. Cited on pages 2, 14, and 123.

- Y. Kim and A. M. Rush (2016). Sequence-Level Knowledge Distillation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing 2016*. Cited on pages 42, 136, and 147.
- Y. Kim, S. Wiseman, and A. M. Rush (2018a). A tutorial on deep latent variable models of natural language, *arXiv preprint arXiv:1812.06834*. Cited on pages 11 and 123.
- Y. Kim, K. Zhang, A. M. Rush, Y. LeCun, *et al.* (2018b). Adversarially Regularized Autoencoders, *International Conference on Machine Learning (ICML)*. Cited on page 40.
- D. Kingma and J. Ba (2015). Adam: A method for stochastic optimization, *International Conference on Learning Representations (ICLR)*. Cited on pages 62, 76, 125, and 142.
- D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling (2016a). Improved variational inference with inverse autoregressive flow, in *Advances in Neural Information Processing Systems 2016*. Cited on pages 57, 62, 84, and 92.
- D. P. Kingma, T. Salimans, and M. Welling (2016b). Improving variational inference with inverse autoregressive flow, *Advances in Neural Information Processing Systems*. Cited on pages 23, 24, 25, and 52.
- D. P. Kingma and M. Welling (2014). Auto-encoding variational bayes, *International Conference on Learning Representations (ICLR)*. Cited on pages 4, 9, 16, 19, 20, 54, 55, 71, 73, 86, and 90.
- T. N. Kipf and M. Welling (2017). Semi-supervised classification with graph convolutional networks, *International Conference on Learning Representations (ICLR)*. Cited on page 3.
- S. Kiyono, S. Takase, J. Suzuki, N. Okazaki, K. Inui, and M. Nagata (2018). Unsupervised Token-wise Alignment to Improve Interpretation of Encoder-Decoder Models, in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP 2018*. Cited on pages 103 and 107.
- G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation, in *Proceedings of ACL 2017, System Demonstrations 2017*. Cited on page 142.
- P. Koehn (2004). Statistical significance tests for machine translation evaluation, in *Proceedings of the 2004 conference on empirical methods in natural language processing 2004*. Cited on page 78.
- P. Koehn and R. Knowles (2017). Six Challenges for Neural Machine Translation, in *Proceedings of the First Workshop on Neural Machine Translation 2017*. Cited on pages 106 and 107.

- P. Koehn and J. Schroeder (2007). Experiments in domain adaptation for statistical machine translation, in *Proceedings of the second workshop on statistical machine translation 2007*. Cited on page 140.
- R. Koncel-Kedziorski, H. Hajishirzi, and A. Farhadi (2014). Multi-resolution language grounding with weak supervision, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) 2014*. Cited on page 119.
- R. Kondadadi, B. Howald, and F. Schilder (2013). A statistical nlg framework for aggregated planning and realization, in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2013*. Cited on page 119.
- I. Konstas and M. Lapata (2013). A global model for concept-to-text generation, *Journal of Artificial Intelligence Research*, vol. 48, pp. 305–346. Cited on page 119.
- W. Kryściński, R. Paulus, C. Xiong, and R. Socher (2018). Improving Abstraction in Text Summarization, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 102, 108, and 110.
- S. Kuang, J. Li, A. Branco, W. Luo, and D. Xiong (2018). Attention Focusing for Neural Machine Translation by Bridging Source and Target Embeddings, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2018*. Cited on page 106.
- K. Kukich (1983). Design of a knowledge-based report generator, in *Proceedings of the 21st annual meeting on Association for Computational Linguistics 1983*. Cited on page 119.
- A. Lamb, V. Dumoulin, and A. Courville (2016a). Discriminative regularization for generative models, *arXiv preprint arXiv:1602.03220*. Cited on page 39.
- A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio (2016b). Professor forcing: A new algorithm for training recurrent networks, in *Advances In Neural Information Processing Systems 2016*. Cited on page 40.
- G. Lample, A. Conneau, L. Denoyer, and M. Ranzato (2018a). Unsupervised machine translation using monolingual corpora only, *International Conference on Learning Representations(ICLR)*. Cited on pages 137 and 140.
- G. Lample, M. Ott, A. Conneau, L. Denoyer, *et al.* (2018b). Phrase-Based & Neural Unsupervised Machine Translation, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 42, 43, 136, 140, and 141.
- A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther (2016). Autoencoding beyond pixels using a learned similarity metric, *International Conference on Machine Learning (ICML)*. Cited on page 38.

- D. Lawson, C.-C. Chiu, G. Tucker, C. Raffel, K. Swersky, and N. Jaitly (2018). Learning hard alignments with variational inference, in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2018*. Cited on pages 91 and 92.
- R. Lebrete, D. Grangier, and M. Auli (2016). Neural Text Generation from Structured Data with Application to the Biography Domain, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing 2016*. Cited on pages 93, 94, and 117.
- J. Lee, E. Mansimov, and K. Cho (2018). Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on page 3.
- S. Lee, S. P. S. Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra (2016). Stochastic multiple choice learning for training diverse deep ensembles, in *Advances in Neural Information Processing Systems 2016*. Cited on page 12.
- T. Lei, R. Barzilay, and T. Jaakkola (2016). Rationalizing Neural Predictions, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing 2016*. Cited on page 87.
- J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan (2016a). A Diversity-Promoting Objective Function for Neural Conversation Models, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2016*. Cited on pages 69, 70, 75, 77, 80, 135, 137, 139, 143, and 144.
- J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan (2016b). A persona-based neural conversation model, *arXiv preprint arXiv:1603.06155*. Cited on pages 135 and 137.
- J. Li and D. Jurafsky (2017). Neural net models for open-domain discourse coherence, *Conference on Empirical Methods in Natural Language Processing*. Cited on page 145.
- J. Li, W. Monroe, and D. Jurafsky (2016c). A Simple, Fast Diverse Decoding Algorithm for Neural Generation, *CoRR*, vol. abs/1611.08562. Cited on pages 70 and 137.
- J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao (2016d). Deep Reinforcement Learning for Dialogue Generation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing 2016*. Cited on pages 46, 70, 75, 77, 78, 80, and 137.
- J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky (2017a). Adversarial Learning for Neural Dialogue Generation, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on pages 46, 78, 79, 137, and 145.

- J. Li, W. X. Zhao, J.-R. Wen, and Y. Song (2019). Generating Long and Informative Reviews with Aspect-Aware Coarse-to-Fine Decoding, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics 2019*. Cited on page 15.
- P. Li, W. Lam, L. Bing, and Z. Wang (2017b). Deep Recurrent Generative Decoder for Abstractive Text Summarization, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on pages 110 and 158.
- W. Li, X. Xiao, Y. Lyu, and Y. Wang (2018). Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling, *Conference on Empirical Methods in Natural Language Processing*. Cited on pages 88 and 92.
- Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu (2017c). DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset, in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers) 2017*. Cited on pages 61 and 76.
- P. Liang, M. I. Jordan, and D. Klein (2009). Learning semantic correspondences with less supervision, in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1 2009*. Cited on pages 118, 119, 122, and 123.
- C.-Y. Lin (2004). Rouge: A package for automatic evaluation of summaries, *Text Summarization Branches Out*, pp. 74–81. Cited on pages 108 and 125.
- C.-Y. Lin and F. J. Och (2004). Orange: a method for evaluating automatic evaluation metrics for machine translation, in *Proceedings of the 20th international conference on Computational Linguistics 2004*. Cited on pages 78 and 157.
- J. Lin, X. SUN, S. Ma, and Q. Su (2018). Global Encoding for Abstractive Summarization, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) 2018*. Cited on page 92.
- J. Ling and A. Rush (2017). Coarse-to-Fine Attention Models for Document Summarization, in *Proceedings of the Workshop on New Frontiers in Summarization 2017*. Cited on pages 89, 92, 118, and 124.
- C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau (2016). How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing 2016*. Cited on pages 64 and 77.
- T. Liu, K. Wang, L. Sha, B. Chang, and Z. Sui (2018). Table-to-text generation by structure-aware seq2seq learning, in *Thirty-Second AAAI Conference on Artificial Intelligence 2018*. Cited on pages 93 and 97.

- J. Lu, A. Kannan, J. Yang, D. Parikh, and D. Batra (2017a). Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model, in *Advances in Neural Information Processing Systems 2017*. Cited on page 145.
- Y. Lu, P. Keung, S. Zhang, J. Sun, and V. Bhardwaj (2017b). A practical approach to dialogue response generation in closed domains, *CoRR*, vol. abs/1703.09439. Cited on page 79.
- Y. Luan, C. Brockett, B. Dolan, J. Gao, and M. Galley (2017). Multi-Task Learning for Speaker-Role Adaptation in Neural Conversation Models, in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers) 2017*. Cited on page 140.
- R. Luo, B. Price, S. Cohen, and G. Shakhnarovich (2018). Discriminability objective for training descriptive captions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018*. Cited on page 145.
- M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser (2016). Multi-task sequence to sequence learning, *International Conference on Learning Representations(ICLR)*. Cited on page 140.
- T. Luong, H. Pham, and C. D. Manning (2015). Effective Approaches to Attention-based Neural Machine Translation, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing 2015*. Cited on pages 2, 87, and 120.
- X. Ma, Y. Gao, Z. Hu, Y. Yu, Y. Deng, and E. Hovy (2017). Dropout with expectation-linear regularization, in *International Conference on Learning Representations 2017*. Cited on page 89.
- C. J. Maddison, A. Mnih, and Y. W. Teh (2017). The concrete distribution: A continuous relaxation of discrete random variables, *International Conference on Learning Representations(ICLR)*. Cited on pages 43 and 70.
- A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey (2016). Adversarial autoencoders, *International Conference on Learning Representations(ICLR)*. Cited on page 58.
- J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille (2015). Deep captioning with multimodal recurrent neural networks (m-rnn), *International Conference on Learning Representations(ICLR)*. Cited on page 94.
- X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley (2016). Least squares generative adversarial networks, *arXiv preprint ArXiv:1611.04076*. Cited on page 38.
- A. Martins and R. Astudillo (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification, in *International Conference on Machine Learning 2016*. Cited on page 3.

- C. May, A. Wang, S. Bordia, S. Bowman, and R. Rudinger (2019). On Measuring Social Biases in Sentence Encoders, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) 2019*. Cited on page 143.
- B. McCann, J. Bradbury, C. Xiong, and R. Socher (2017). Learned in translation: Contextualized word vectors, in *Advances in Neural Information Processing Systems 2017*. Cited on page 106.
- B. McCann, N. S. Keskar, C. Xiong, and R. Socher (2019). *The Natural Language Decathlon: Multitask Learning as Question Answering*. Cited on page 2.
- K. McKeown (1992). *Text generation*, Cambridge University Press. Cited on page 119.
- H. Mei, T. UChicago, M. Bansal, and M. R. Walter (2016). What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment, in *Proceedings of NAACL-HLT 2016*. Cited on pages 86, 92, 93, and 117.
- S. Merity, C. Xiong, J. Bradbury, and R. Socher (2017). Pointer sentinel mixture models, *International Conference on Learning Representations(ICLR)*. Cited on pages 11 and 101.
- L. Mescheder, S. Nowozin, and A. Geiger (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks, *International Conference on Machine Learning (ICML)*. Cited on page 38.
- L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein (2017). Unrolled generative adversarial networks, *International Conference on Learning Representations(ICLR)*. Cited on pages 32 and 35.
- Y. Miao, L. Yu, and P. Blunsom (2016). Neural variational inference for text processing, in *International Conference on Machine Learning 2016*. Cited on page 47.
- T. Mikolov, K. Chen, G. Corrado, and J. Dean (2013a). Efficient estimation of word representations in vector space, *International Conference on Learning Representation workshop*. Cited on page 76.
- T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013b). Distributed representations of words and phrases and their compositionality, in *Advances in neural information processing systems 2013*. Cited on page 105.
- A. Mnih and K. Gregor (2014). Neural Variational Inference and Learning in Belief Networks, in *International Conference on Machine Learning 2014*. Cited on pages 71, 86, and 90.
- A. Mnih and D. J. Rezende (2016). Variational Inference for Monte Carlo Objectives, in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 2016*. Cited on page 90.

- S. Mohamed and B. Lakshminarayanan (2017). Learning in implicit generative models, *International Conference on Learning Representation workshop*. Cited on page 28.
- T. K. Moon (1996). The expectation-maximization algorithm, *IEEE Signal processing magazine*, vol. 13(6), pp. 47–60. Cited on page 41.
- A. Moryossef, Y. Goldberg, and I. Dagan (2019). Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) 2019*. Cited on pages 92 and 119.
- R. Nallapati, F. Zhai, and B. Zhou (2017). SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents., in *Association for the Advancement of Artificial Intelligence 2017*. Cited on page 87.
- C. Napoles, M. Gormley, and B. Van Durme (2012). Annotated gigaword, in *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction 2012*. Cited on page 108.
- S. Narayan, S. B. Cohen, and M. Lapata (2018). Don’t Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 103, 107, 108, 110, and 158.
- R. M. Neal and G. E. Hinton (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants, in *Learning in graphical models 1998*, pp. 355–368, Springer. Cited on page 10.
- P. Nema, M. M. Khapra, A. Laha, and B. Ravindran (2017). Diversity driven attention model for query-based abstractive summarization, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2017*. Cited on page 85.
- D. T. Nguyen and T. Tran (2018). Structurebased Generation System for E2E NLG Challenge, *arxiv*. Cited on page 127.
- T. Nguyen and D. Chiang (2018). Improving Lexical Choice in Neural Machine Translation, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) 2018*. Cited on page 106.
- T. Niu and M. Bansal (2018). Polite dialogue generation without parallel data, *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 373–389. Cited on pages 137 and 140.

- J. Novikova, O. Dušek, A. C. Curry, and V. Rieser (2017a). Why We Need New Evaluation Metrics for NLG, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on page 125.
- J. Novikova, O. Dušek, and V. Rieser (2017b). The E2E Dataset: New Challenges For End-to-End Generation, in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue 2017*. Cited on page 125.
- S. Nowozin, B. Cseke, and R. Tomioka (2016). f-GAN: Training generative neural samplers using variational divergence minimization, in *Advances in Neural Information Processing Systems 2016*. Cited on pages 30 and 84.
- F. J. Och and H. Ney (2000). A comparison of alignment models for statistical machine translation, in *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics 2000*. Cited on page 114.
- F. J. Och, C. Tillmann, and H. Ney (1999). Improved alignment models for statistical machine translation, in *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora 1999*. Cited on pages 1, 13, and 122.
- A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu (2016). Pixel recurrent neural networks, *arXiv preprint arXiv:1601.06759*. Cited on page 48.
- P. Over, H. Dang, and D. Harman (2007). DUC in context, *Information Processing & Management*, vol. 43(6), pp. 1506–1520. Cited on pages 95 and 112.
- T. Oya, Y. Mehdad, G. Carenini, and R. Ng (2014). A template-based abstractive meeting summarization: Leveraging summary and source text relationships, in *Proceedings of the 8th International Natural Language Generation Conference (INLG) 2014*. Cited on pages 117 and 119.
- J. Paisley, D. Blei, and M. Jordan (2012). Variational Bayesian inference with stochastic search, *International Conference on Machine Learning (ICML)*. Cited on page 19.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu (2002). BLEU: a method for automatic evaluation of machine translation, in *Proceedings of the 40th annual meeting on association for computational linguistics 2002*. Cited on pages 78, 125, and 144.
- T. Parshakova, J.-M. Andreoli, and M. Dymetman (2019). Distributional Reinforcement Learning for Energy-Based Sequential Models, *arXiv preprint arXiv:1912.08517*. Cited on page 3.
- A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017). Automatic differentiation in PyTorch, in *Advances in Neural Information Processing Systems workshop 2017*. Cited on page 76.

- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.* (2019). PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 2019*. Cited on pages 4 and 125.
- R. Paulus, C. Xiong, and R. Socher (2018). A deep reinforced model for abstractive summarization, *International Conference on Learning Representations (ICLR)*. Cited on pages 70 and 108.
- J. Pennington, R. Socher, and C. D. Manning (2014). GloVe: Global Vectors for Word Representation, in *Empirical Methods in Natural Language Processing (EMNLP) 2014*. Cited on pages 105 and 125.
- M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). Deep Contextualized Word Representations, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) 2018*. Cited on page 106.
- O. Press, A. Bar, B. Bogin, J. Berant, and L. Wolf (2017). Language Generation with Recurrent Generative Adversarial Networks without Pre-training, *arXiv preprint arXiv:1706.01399*. Cited on page 44.
- O. Press and L. Wolf (2017). Using the Output Embedding to Improve Language Models, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers 2017*. Cited on pages 3 and 104.
- Y. Pu, L. Chen, S. Dai, W. Wang, C. Li, and L. Carin (2017). Symmetric Variational Autoencoder and Connections to Adversarial Learning, *Advances in Neural Information Processing Systems*. Cited on page 39.
- R. Puduppully, L. Dong, and M. Lapata (2019). Data-to-text generation with content selection and planning, in *Proceedings of the AAAI Conference on Artificial Intelligence 2019*. Cited on pages 15 and 119.
- Y. Puzikov and I. Gurevych (2018). E2e nlg challenge: Neural models vs. templates, in *Proceedings of the 11th International Conference on Natural Language Generation 2018*. Cited on page 127.
- G. Qin, J.-G. Yao, X. Wang, J. Wang, and C.-Y. Lin (2018). Learning Latent Semantic Annotations for Grounding Natural Language to Structured Data, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 85 and 119.
- R. E. Quandt (1972). A new approach to estimating switching regressions, *Journal of the American statistical association*, vol. 67(338), pp. 306–310. Cited on page 11.
- L. R. Rabiner (1989). A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE*, vol. 77(2), pp. 257–286. Cited on page 122.

- A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever (2019). Language models are unsupervised multitask learners, *arXiv*. Cited on page 2.
- C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu (2019). Exploring the limits of transfer learning with a unified text-to-text transformer, *arXiv preprint arXiv:1910.10683*. Cited on page 2.
- T. Raiko, M. Berglund, G. Alain, and L. Dinh (2015). Techniques for learning binary stochastic feedforward neural networks, *International Conference on Learning Representations(ICLR)*. Cited on page 90.
- L. Reed, S. Oraby, and M. Walker (2018). Can Neural Generators for Dialogue Learn Sentence Planning and Discourse Structuring?, in *Proceedings of the 11th International Conference on Natural Language Generation 2018*. Cited on page 123.
- E. Reiter and A. Belz (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems, *Computational Linguistics*, vol. 35(4), pp. 529–558. Cited on page 125.
- E. Reiter and R. Dale (1997). Building applied natural language generation systems, *Natural Language Engineering*, vol. 3(1), pp. 57–87. Cited on pages 117 and 119.
- E. Reiter and R. Dale (2000). *Building natural language generation systems*, Cambridge university press. Cited on pages 1 and 85.
- E. Reiter, C. Mellish, and J. Levine (1995). Automatic generation of technical documentation, *Applied Artificial Intelligence an International Journal*, vol. 9(3), pp. 259–287. Cited on page 1.
- D. J. Rezende and S. Mohamed (2015). Variational inference with normalizing flows, *arXiv preprint arXiv:1505.05770*. Cited on pages 23 and 57.
- D. J. Rezende, S. Mohamed, and D. Wierstra (2014). Stochastic backpropagation and approximate inference in deep generative models, *arXiv preprint arXiv:1401.4082*. Cited on pages 4, 19, 54, and 55.
- S. Riezler and J. T. Maxwell (2005). On some pitfalls in automatic evaluation and significance testing for MT, in *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization 2005*. Cited on pages 78 and 157.
- A. Ritter, C. Cherry, and W. B. Dolan (2011). Data-driven response generation in social media, in *Proceedings of the conference on empirical methods in natural language processing 2011*. Cited on page 76.
- M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed (2017). Variational Approaches for Auto-Encoding Generative Adversarial Networks, *arXiv preprint arXiv:1706.04987*. Cited on page 38.

- K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann (2017). Stabilizing Training of Generative Adversarial Networks through Regularization, *arXiv preprint arXiv:1705.09367*. Cited on page 34.
- V. Rus and M. Lintean (2012). A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics, in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP 2012*. Cited on page 77.
- A. M. Rush, S. Chopra, and J. Weston (2015). A Neural Attention Model for Abstractive Sentence Summarization, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing 2015*. Cited on pages 2, 93, 94, 103, and 107.
- R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani (2003). Optimization with EM and expectation-conjugate-gradient, in *Proceedings of the 20th International Conference on Machine Learning (ICML-03) 2003*. Cited on page 10.
- T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). Improved techniques for training gans, in *Advances in Neural Information Processing Systems 2016*. Cited on pages 33, 34, and 43.
- T. Salimans, D. Kingma, and M. Welling (2015). Markov chain monte carlo and variational inference: Bridging the gap, in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15) 2015*. Cited on pages 22 and 57.
- N. Schluter (2017). The limits of automatic summarisation according to rouge, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers 2017*. Cited on page 112.
- F. Schmidt, S. Mandt, and T. Hofmann (2019). Autoregressive Text Generation Beyond Feedback Loops, *arXiv preprint arXiv:1908.11658*. Cited on page 3.
- A. See, P. J. Liu, and C. D. Manning (2017). Get To The Point: Summarization with Pointer-Generator Networks, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2017*. Cited on pages 11, 101, 104, 107, 108, 109, 111, 121, and 158.
- S. Semeniuta, A. Severyn, and E. Barth (2017). A Hybrid Convolutional Variational Autoencoder for Text Generation, *arXiv preprint arXiv:1702.02390*. Cited on pages 51 and 62.
- R. Sennrich, B. Haddow, and A. Birch (2016a). Improving Neural Machine Translation Models with Monolingual Data, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2016*. Cited on pages 42, 136, 137, and 140.
- R. Sennrich, B. Haddow, and A. Birch (2016b). Neural Machine Translation of Rare Words with Subword Units, in *Proceedings of the 54th Annual Meeting of the*

- Association for Computational Linguistics (Volume 1: Long Papers) 2016*. Cited on pages 93 and 103.
- I. V. Serban, A. G. O. II, J. Pineau, and A. Courville (2017a). Piecewise Latent Variables for Neural Variational Text Processing, *Conference on Empirical Methods in Natural Language Processing*. Cited on page 57.
- I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, *et al.* (2017b). A deep reinforcement learning chatbot, *arXiv preprint arXiv:1709.02349*. Cited on page 137.
- I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau (2016). Building end-to-end dialogue systems using generative hierarchical neural network models, in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence 2016*. Cited on pages 53, 69, and 71.
- I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio (2017c). A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues, in *Thirty-First AAAI Conference on Artificial Intelligence 2017*. Cited on pages 54, 55, 70, 75, 76, 77, 137, and 143.
- I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio (2017d). A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues., in *Association for the Advancement of Artificial Intelligence 2017*. Cited on pages 4 and 47.
- L. Shang, Z. Lu, and H. Li (2015). Neural responding machine for short-text conversation, *arXiv preprint arXiv:1503.02364*. Cited on pages 136 and 142.
- S. Shankar, S. Garg, and S. Sarawagi (2018). Surprisingly easy hard-attention for sequence to sequence learning, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 12, 106, and 107.
- S. Shankar and S. Sarawagi (2019). Posterior Attention Models for Sequence to Sequence Learning, in *International Conference on Learning Representations 2019*. Cited on pages 12, 14, 102, 107, 119, and 122.
- D. Shen, Y. Zhang, R. Henao, Q. Su, and L. Carin (2017a). Deconvolutional Latent-Variable Model for Text Sequence Matching, *arXiv preprint arXiv:1709.07109*. Cited on pages 43 and 52.
- T. Shen, T. Lei, R. Barzilay, and T. Jaakkola (2017b). Style transfer from non-parallel text by cross-alignment, in *Advances in neural information processing systems 2017*. Cited on page 1.
- T. Shen, M. Ott, M. Auli, and M. Ranzato (2019a). Mixture Models for Diverse Machine Translation: Tricks of the Trade, in *International Conference on Machine Learning 2019*. Cited on page 12.

- T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang (2018a). Reinforced Self-Attention Network: a Hybrid of Hard and Soft Attention for Sequence Modeling, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18 2018*. Cited on page 91.
- X. Shen, E. Chang, H. Su, C. Niu, and D. Klakow (2020). Neural Data-to-Text Generation via Jointly Learning the Segmentation and Correspondence, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics 2020*. Cited on page 117.
- X. Shen, Y. Oualil, C. Greenberg, M. Singh, and D. Klakow (2017c). Estimation of Gap Between Current Language Models and Human Performance, *Proc. Interspeech 2017*, pp. 553–557. Cited on pages 2, 117, and 135.
- X. Shen, H. Su, W. Li, and D. Klakow (2018b). Nexus network: Connecting the preceding and the following in dialogue generation, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 69, 117, and 137.
- X. Shen, H. Su, Y. Li, W. Li, S. Niu, Y. Zhao, A. Aizawa, and G. Long (2017d). A Conditional Variational Framework for Dialog Generation, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) 2017*. Cited on pages 53, 70, and 137.
- X. Shen, H. Su, S. Niu, and V. Demberg (2018c). Improving Variational Encoder-Decoders in Dialogue Generation, *Association for the Advancement of Artificial Intelligence*, pp. 5456–5463. Cited on pages 53, 73, 75, 78, 92, and 137.
- X. Shen, J. Suzuki, K. Inui, H. Su, D. Klakow, and S. Sekine (2019b). Select and Attend: Towards Controllable Content Selection in Text Generation, *arXiv preprint arXiv:1909.04453*. Cited on pages 4, 5, 16, 85, 117, and 135.
- X. Shen, Y. Zhao, H. Su, and D. Klakow (2019c). Improving Latent Alignment in Text Summarization by Generalizing the Pointer Generator, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) 2019*. Cited on pages 101, 119, and 137.
- R. Shetty, M. Rohrbach, L. A. Hendricks, M. Fritz, and B. Schiele (2017). Speaking the same language: Matching machine to human captions by adversarial training, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) 2017*. Cited on page 70.
- K. Sohn, H. Lee, and X. Yan (2015). Learning structured output representation using deep conditional generative models, in *Advances in Neural Information Processing Systems 2015*. Cited on pages 47, 54, 55, 75, and 97.

- C. K. Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár (2016). Amortised map inference for image super-resolution, *arXiv preprint arXiv:1610.04490*. Cited on page 33.
- A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan (2015). A Neural Network Approach to Context-Sensitive Generation of Conversational Responses, in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2015*. Cited on page 78.
- A. Sriram, H. Jun, S. Satheesh, and A. Coates (2018). Cold Fusion: Training Seq2Seq Models Together with Language Models, *Proc. Interspeech 2018*, pp. 387–391. Cited on page 3.
- A. Srivastava and C. Sutton (2017). Autoencoding Variational Inference For Topic Models, *International Conference on Learning Representations(ICLR)*. Cited on page 47.
- A. Srivastava, L. Valkov, C. Russell, M. Gutmann, and C. Sutton (2017). VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning, *Advances in Neural Information Processing Systems*. Cited on page 38.
- F.-G. Su, A. R. Hsu, Y.-L. Tuan, and H.-Y. Lee (2019a). Personalized Dialogue Response Generation Learned from Monologues, *Proc. Interspeech 2019*, pp. 4160–4164. Cited on page 137.
- H. Su, X. Shen, P. Hu, W. Li, and Y. Chen (2018). Dialogue generation with GAN, in *Thirty-Second AAAI Conference on Artificial Intelligence 2018*. Cited on pages 135 and 137.
- H. Su, X. Shen, R. Zhang, F. Sun, P. Hu, C. Niu, and J. Zhou (2019b). Improving Multi-turn Dialogue Modelling with Utterance ReWriter, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics 2019*. Cited on pages 117 and 135.
- H. Su, X. Shen, S. Zhao, Z. Xiao, P. Hu, R. Zhong, C. Niu, and J. Zhou (2020). Diversifying Dialogue Generation with Non-Conversational Text, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics 2020*. Cited on page 135.
- S. Subramanian, G. Lample, E. M. Smith, L. Denoyer, M. Ranzato, and Y.-L. Boureau (2019). Multiple-Attribute Text Style Transfer, *International Conference on Learning Representations(ICLR)*. Cited on pages 42 and 137.
- S. Subramanian, S. Rajeswar, F. Dutil, C. Pal, and A. Courville (2017). Adversarial Generation of Natural Language, in *Proceedings of the 2nd Workshop on Representation Learning for NLP 2017*. Cited on page 44.

- I. Sutskever, O. Vinyals, and Q. V. Le (2014). Sequence to sequence learning with neural networks, in *Advances in neural information processing systems 2014*. Cited on pages 1, 48, 55, 69, 85, and 107.
- C. Sutton, A. McCallum, *et al.* (2012). An introduction to conditional random fields, *Foundations and Trends® in Machine Learning*, vol. 4(4), pp. 267–373. Cited on page 10.
- H. Thompson (1977). Strategy and tactics: A model for language production, in *Papers from the... Regional Meeting. Chicago Ling. Soc. Chicago, Ill 1977*. Cited on page 1.
- J. Tomczak and M. Welling (2018). VAE with a VampPrior, in *International Conference on Artificial Intelligence and Statistics 2018*. Cited on page 84.
- T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard (2016). Complex embeddings for simple link prediction, in *International Conference on Machine Learning 2016*. Cited on page 106.
- G. Tucker, D. Lawson, S. Gu, and C. J. Maddison (2019). Doubly Reparameterized Gradient Estimators for Monte Carlo Objectives, in *International Conference on Learning Representations 2019*. Cited on page 91.
- G. Tucker, A. Mnih, C. J. Maddison, J. Lawson, and J. Sohl-Dickstein (2017). Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models, in *Advances in Neural Information Processing Systems 2017*. Cited on pages 71 and 90.
- D. Ulyanov, A. Vedaldi, and V. Lempitsky (2017). Adversarial Generator-Encoder Networks, *arXiv preprint arXiv:1704.02304*. Cited on page 39.
- A. van den Oord, O. Vinyals, *et al.* (2017). Neural discrete representation learning, in *Advances in Neural Information Processing Systems 2017*. Cited on pages 71 and 92.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is all you need, in *Advances in Neural Information Processing Systems 2017*. Cited on pages 3 and 103.
- R. Vedantam, C. Lawrence Zitnick, and D. Parikh (2015). Cider: Consensus-based image description evaluation, in *Proceedings of the IEEE conference on computer vision and pattern recognition 2015*. Cited on page 125.
- A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. J. Crandall, and D. Batra (2018). Diverse Beam Search for Improved Description of Complex Scenes., *Association for the Advancement of Artificial Intelligence*, pp. 7371–7379. Cited on pages 80 and 143.
- C. Villani (2008). *Optimal transport: old and new*, vol. 338, Springer Science & Business Media. Cited on page 37.

- O. Vinyals, M. Fortunato, and N. Jaitly (2015). Pointer networks, in *Advances in Neural Information Processing Systems 2015*. Cited on pages 104 and 107.
- O. Vinyals and Q. V. Le (2015). A Neural Conversational Model, *CoRR*, vol. abs/1506.05869. Cited on pages 1, 53, 77, 135, and 137.
- E. Vylomova, L. Rimell, T. Cohn, and T. Baldwin (2016). Take and Took, Gaggles and Gooses, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2016*. Cited on page 105.
- C. Wang, Y. Wang, P.-S. Huang, A. Mohamed, D. Zhou, and L. Deng (2017a). Sequence modeling via segmentations, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70 2017*. Cited on pages 119 and 123.
- D. Wang, N. Jojic, C. Brockett, and E. Nyberg (2017b). Steering Output Style and Topic in Neural Response Generation, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on pages 70, 75, 137, and 140.
- L. Wang, A. Schwing, and S. Lazebnik (2017c). Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space, in *Advances in Neural Information Processing Systems 2017*. Cited on pages 88 and 94.
- W. Wang, Z. Hu, Z. Yang, H. Shi, F. Xu, and E. Xing (2019). Toward Unsupervised Text Content Manipulation, *arXiv preprint arXiv:1901.09501*. Cited on page 91.
- W. Wang, D. Zhu, T. Alkhoul, Z. Gan, and H. Ney (2018a). Neural hidden markov model for machine translation, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) 2018*. Cited on page 119.
- X. Wang, H. Pham, P. Yin, and G. Neubig (2018b). A Tree-based Decoder for Neural Machine Translation, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 15 and 153.
- L. Weaver and N. Tao (2001). The Optimal Reward Baseline for Gradient-based Reinforcement Learning, in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence 2001*. Cited on page 90.
- T.-H. Wen, M. Gasic, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young (2015). Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing 2015*. Cited on pages 2 and 91.
- T.-H. Wen, Y. Miao, P. Blunsom, and S. Young (2017). Latent Intention Dialogue Models, in *Proceedings of the 34th International Conference on Machine Learning 2017*. Cited on page 92.

- R. J. Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine learning*, vol. 8(3-4), pp. 229–256. Cited on pages 46, 70, 89, and 90.
- S. Wiseman, S. Shieber, and A. Rush (2017). Challenges in Data-to-Document Generation, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on pages 104, 120, and 121.
- S. Wiseman, S. M. Shieber, and A. M. Rush (2018). Learning Neural Templates for Text Generation, *Conference on Empirical Methods in Natural Language Processing*. Cited on pages 14, 85, 117, 119, 122, 123, 127, 130, and 132.
- C.-S. Wu, R. Socher, and C. Xiong (2019). Global-to-local Memory Pointer Networks for Task-Oriented Dialogue, in *International Conference on Learning Representations 2019*. Cited on page 12.
- S. Wu, P. Shapiro, and R. Cotterell (2018). Hard Non-Monotonic Attention for Character-Level Transduction, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on pages 13, 14, 118, 119, and 153.
- Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.* (2016). Google’s neural machine translation system: Bridging the gap between human and machine translation, *arXiv preprint arXiv:1609.08144*. Cited on pages 103 and 108.
- Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li (2017). Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2017*. Cited on pages 136, 139, and 142.
- Z. Xie (2017). Neural text generation: A practical guide, *arXiv preprint arXiv:1711.09534*. Cited on pages 3 and 153.
- Z. Xie, S. I. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, and A. Y. Ng (2017). Data Noising as Smoothing in Neural Network Language Models, *International Conference on Learning Representations(ICLR)*. Cited on pages 50, 57, and 63.
- K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio (2015). Show, attend and tell: Neural image caption generation with visual attention, in *International conference on machine learning 2015*. Cited on pages 1, 2, 87, and 106.
- X. Yan, J. Yang, K. Sohn, and H. Lee (2016). Attribute2image: Conditional image generation from visual attributes, in *European Conference on Computer Vision 2016*. Cited on page 47.

- Z. Yang, P. Blunsom, C. Dyer, and W. Ling (2017a). Reference-Aware Language Models, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing 2017*. Cited on pages 12 and 101.
- Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen (2018). Breaking the Softmax Bottleneck: A High-Rank RNN Language Model, in *International Conference on Learning Representations 2018*. Cited on page 12.
- Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick (2017b). Improved Variational Autoencoders for Text Modeling using Dilated Convolutions, *International Conference on Machine Learning (ICML)*. Cited on pages 43, 52, 57, 58, and 92.
- Z. Yang, w. wu, J. Yang, C. Xu, and z. li (2019). Low-Resource Response Generation with Template Prior, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) 2019*. Cited on page 124.
- L. Yao, N. Peng, W. Ralph, K. Knight, D. Zhao, and R. Yan (2019). Plan-And-Write: Towards Better Automatic Storytelling, *Association for the Advancement of Artificial Intelligence*. Cited on page 92.
- L. Yu, J. Buys, and P. Blunsom (2016). Online Segment to Segment Neural Transduction, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing 2016*. Cited on pages 14, 107, 119, and 122.
- L. Yu, W. Zhang, J. Wang, and Y. Yu (2017). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient., in *Association for the Advancement of Artificial Intelligence 2017*. Cited on pages 3 and 46.
- N. Yu, J. Zhang, M. Huang, and X. Zhu (2018). An Operation Network for Abstractive Sentence Compression, in *Proceedings of the 27th International Conference on Computational Linguistics 2018*. Cited on pages 87 and 92.
- B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang (2016a). Variational neural machine translation, *arXiv preprint arXiv:1605.07869*. Cited on page 47.
- F. Zhang, J.-g. Yao, and R. Yan (2018a). On the Abtractiveness of Neural Document Summarization, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing 2018*. Cited on page 110.
- S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston (2018b). Personalizing Dialogue Agents: I have a dog, do you have pets too?, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2018*. Cited on page 137.
- Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, and B. Dolan (2018c). Generating informative and diverse conversational responses via adversarial information maximization, in *Advances in Neural Information Processing Systems 2018*. Cited on pages 137, 139, and 144.

- Y. Zhang, Z. Gan, and L. Carin (2016b). Generating text via adversarial training, in *Advances in Neural Information Processing Systems workshop on Adversarial Training 2016*. Cited on page 44.
- Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin (2017). Adversarial Feature Matching for Text Generation, *International Conference on Machine Learning (ICML)*. Cited on page 44.
- J. Zhao, M. Mathieu, and Y. LeCun (2017a). Energy-based generative adversarial network, *International Conference on Learning Representations (ICLR)*. Cited on page 38.
- S. Zhao, J. Song, and S. Ermon (2017b). Towards Deeper Understanding of Variational Autoencoding Models, *arXiv preprint arXiv:1702.08658*. Cited on page 22.
- S. Zhao, J. Song, and S. Ermon (2018a). The Information Autoencoding Family: A Lagrangian Perspective on Latent Variable Generative Models, *UAI*. Cited on pages 87 and 91.
- T. Zhao, R. Zhao, and M. Eskenazi (2017c). Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2017*. Cited on pages 51, 57, 62, 70, 75, 77, 78, 135, and 143.
- Y. Zhao, H. Senuma, X. Shen, and A. Aizawa (2017d). Gated neural network for sentence compression using linguistic knowledge, in *International Conference on Applications of Natural Language to Information Systems 2017*. Cited on page 2.
- Y. Zhao, X. Shen, W. Bi, and A. Aizawa (2019). Unsupervised Rewriter for Multi-Sentence Compression, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics 2019*. Cited on page 137.
- Y. Zhao, X. Shen, H. Senuma, and A. Aizawa (2018b). A comprehensive study: Sentence compression with linguistic knowledge-enhanced gated neural network, *Data & Knowledge Engineering*, vol. 117, pp. 307–318. Cited on pages 2, 119, and 137.
- C. Zhou and G. Neubig (2017). Multi-space Variational Encoder-Decoders for Semi-supervised Labeled Sequence Transduction, *Annual Meeting of the Association for Computational Linguistics*. Cited on page 43.
- H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu (2018). Emotional chatting machine: Emotional conversation generation with internal and external memory, in *Thirty-Second AAAI Conference on Artificial Intelligence 2018*. Cited on page 137.
- Q. Zhou, N. Yang, F. Wei, and M. Zhou (2017). Selective Encoding for Abstractive Sentence Summarization, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) 2017*. Cited on pages 86, 92, 93, 96, 97, 108, 110, and 158.

- W. Zhou, T. Ge, K. Xu, F. Wei, and M. Zhou (2020). Self-Adversarial Learning with Comparative Discrimination for Text Generation, in *International Conference on Learning Representations 2020*. Cited on page 3.
- Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu (2018). Texus: a benchmarking platform for text generation models, in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval 2018*. Cited on page 95.

