Word meaning in context: A probabilistic model and its application to Question Answering

Georgiana Dinu

Dissertation
Presented to the Faculty
of Philosophy II
of Saarland University
in Candidacy for the Degree of
Doctor of Philosophy

Dissertation
zur Erlangung des Grades
des Doktors der Philosophie
der Philosophische Fakultät II
der Universität des Saarlandes

Saarbrücken
November 2011

Abstract

The need for assessing similarity in meaning is central to most language technology applications. Distributional methods are robust, unsupervised methods which achieve high performance on this task. These methods measure similarity of word types solely based on patterns of word occurrences in large corpora, following the intuition that similar words occur in similar contexts. As most Natural Language Processing (NLP) applications deal with disambiguated words, words occurring in context, rather than word types, the question of adapting distributional methods to compute sense-specific or context-sensitive similarities has gained increasing attention in recent work.

This thesis focuses on the development and applications of distributional methods for context-sensitive similarity. The contribution made is twofold: the main part of the thesis proposes and tests a new framework for computing similarity in context, while the second part investigates the application of distributional paraphrasing to the task of question answering.

New framework for context-sensitive distributional similarity

The capacity to identify words, or larger units of text, which convey similar meaning is of major importance to a large number of NLP applications which require at least a minimal level of text understanding.

Distributional methods are the most robust approach to the task of computing word similarity. These are based on the hypothesis, initially formulated by Harris (also known as the distributional hypothesis), stating that words occurring in similar contexts tend to have similar meaning. Following this intuition, these methods measure relatedness in meaning as indicated by patterns of word occurrence in large corpora. More precisely, words are represented as high-dimensional vectors, where the dimensions represent context features, such as co-occurring context words. The relatedness of two words can be assessed by comparing their associated vector representations.

The task of assessing meaning similarity is challenged by sense ambiguity, the capacity of words to convey a number of different meanings. In the case of homonymy etymologically distinct words accidentally share the same surface realization and the conveyed meanings are significantly divergent. On the other hand, polysemous words can exhibit a large number of *related* senses which are determined by the distinctions in the prototypical contexts in which they appear.

In a larger context, sense ambiguity is one of the major challenges that all language technology applications are faced with, from information retrieval to machine translation. The focus of recent work has been the question of adapting distributional methods to compute sense-specific or context-sensitive similarities. Distributional methods, in their traditional formulation, represent meanings of word types and are therefore incapable of differentiating between different usages, or different senses of a word. Furthermore, in a vector space model this is made difficult by the fact that the vector representation of a word mixes together all its different senses and usages over an entire corpus.

A number of methods for computing context-sensitive similarity within the distributional representation paradigm have been proposed in the literature.

Our research goal is to investigate means of computing context-sensitive distributional similarity which are aware of word senses. We propose to describe a word in terms of a distribution over a set of data-induced meaning components while the disambiguation process, leading to the meaning of words in context, is modeled as a shift in this distribution. Specifically, the framework developed represents words, occurring in isolation or in context, as probability distributions over a global set of corpus-induced meaning components, or meaning aspects. When given a word without a context, this representation reflects its a priori meaning as a distribution over this set of latent meaning components. When given a context, a shift in this distribution determines a *disambiguated* representation, in which meaning components which are validated by the context become more likely. Such representations can now be compared with a clear interpretation: words, occurring both isolated or in context, have similar meaning if they trigger the same latent components.

In turn, the meaning components themselves are induced form the corpus in an unsupervised fashion. The intuition behind this is that each occurrence of a word together with a context feature is explained by a latent meaning, or latent meaning component, and the sum of all occurrences of a word is a mixture over such latent classes. The goal is to induce the set of latent classes that best explain the corpus co-occurrence data. For this, we follow Hofmann's (Hofmann and Puzicha [1989]) framework for unsupervised learning from dyadic data and we use two variations of this method to induce latent classes.

Unlike previous work, our framework models the meaning of words in context in a probabilistic setting, in which the meaning representations, as well as the similarity computations, are obtained in a natural, intuitive fashion. Furthermore, the framework is completely modular, as it can be applied to any type of vector space model (VSM) and used with any suitable latent variable induction model/algorithm. In this thesis, the framework is instantiated on a word-level VSM, for the task of lexical substitution as well as on a paraphrase acquisition VSM; we obtain promising results on both of these tasks.

Distributional paraphrasing for question answering

The second part of this thesis approaches the application of distributional paraphrasing to the task of question answering. More precisely, the focus is twofold:

1. To investigate the integration of a paraphrasing component in an answer extraction module.
2. To test the impact of *context-sensitive* paraphrasing on the task of question answering.

Paraphrases are pairs of phrases which convey similar meaning, and can therefore substitute each other without changing the meaning of the sentences they occur in. The purpose of paraphrase induction is again that of addressing the challenge posed to most NLP applications by the fact that the same meaning can expressed through a variety of different surface realizations.

A robust way to approach the task of automatically acquiring paraphrases is, again, through the use of distributional methods. The mechanism behind this is similar to that of computing lexical similarity as (typically small) phrases are treated as atomic units; their occurrence patterns in large corpora are used for computing similarity scores between such phrases. In particular, throughout this thesis, we remain within the representation paradigm proposed by the DIRT (Discovery of Inference Rules from Text) algorithm in Lin and Pantel [2001a]. Phrases are paths in dependency graphs which are represented in a space of contextual features consisting of the two words to the left and right of the phrase paths. The vector representations obtained this way are used to measure phrase similarity. When given a particular phrase, its top most similar phrases, according to the similarity scores, are returned as paraphrases. An example of a typical paraphrase obtained with the DIRT algorithm is X *solve* Y $\approx$ X *find solution to* Y.

Our target application is Question answering (QA). This is the task of automatically extracting answers to questions from large collections of text. One of the main issues that QA systems face is caused by the fact that the same meaning is often expressed differently in questions and in answer-containing sentences. The use of paraphrases is one of the most natural ways to address this problem and methods such as Lin and Pantel [2001a] have been developed with applications such as QA in mind.

We investigate the use of paraphrasing in QA by building a basic question answering system centered around a paraphrasing component. We initially investigate what are appropriate ways of integrating a paraphrase component into such a system, which make the most use of the knowledge encoded in a collection. We propose a language-model based answer extraction module, which can be naturally enhanced with a paraphrasing component. Unlike previous work, we focus on robustness: rather than using the paraphrase rules to obtain exact matches between sentences and questions, we rather allow them to *reduce* the distance between question and answer-containing sentences.

Further on, we instantiate the context-sensitive similarity framework for the paraphrasing task and test the impact of context-sensitive paraphrasing to the question answering system. We observe that the use of paraphrasing brings overall improvements, which although significant, are severely limited by low coverage: while large improvements can be observed on the sentences in which paraphrases are used, the number of cases in which these are used is very small. In future work we plan to investigate in more detail the limitations we observe, in order to identify if these are true limitations of the paraphrasing method, or are caused by other factors.

Kurzzusammenfassung

Die Notwendigkeit der Beurteilung von Bedeutungs ähnlichkeit spielt für die meisten sprachtechnologische Anwendungen eine wesentliche Rolle. Distributionelle Verfahren sind solide, unbeaufsichtigte Verfahren, die für diese Aufgabe sehr effektiv sind. Diese Verfahren messen die Ähnlichkeit von Wortarten lediglich auf Basis von Mustern, nach denen die Wörter in grossen Korpora vorkommen, indem sie der Erkenntnis folgen, dass ähnliche Wörter in ähnlichen Kontexten auftreten. Da die meisten Anwendungen im Natural Language Processing (NLP) mit eindeutigen Wörtern arbeiten, also eher Wörtern, die im Kontext vorkommen, als Wortarten, hat die Frage, ob distributionelle Verfahren angepasst werden sollten, um bedeutungsspezifische oder kontextabhängige Ähnlichkeiten zu berechnen, in neueren Arbeiten zunehmend an Bedeutung gewonnen. Diese Dissertation konzentriert sich auf die Entwicklung und Anwendungen von distributionellen Verfahren für kontextabhängige Ähnlichkeit und liefert einen doppelten Beitrag: Den Hauptteil der Arbeit bildet die Präsentation und Erprobung eines neuen framework für die Berechnung von Ähnlichkeit im Kontext. Im zweiten Teil der Arbeit wird die Anwendung des distributional paraphrasing auf die Aufgabe der Fragenbeantwortung untersucht.

Neuer framework für kontextabhängige distributionelle Ähnlichkeit

Die Fähigkeit der Identifikation von Wörtern oder grösseren Texteinheiten, die eine ähnliche Bedeutung haben, spielt für viele NLP-Anwendungen, die ein Mindestmass an Textverständnis erfordern, eine grosse Bedeutung.

Distributionelle Verfahren sind die sicherste Methode zur Berechnung von Wortähnlichkeit. Diese Verfahren basieren auf der Hypothese, die ursprünglich von Harris aufgestellt wurde und auch als distributionelle Hypothese bekannt ist. Diese Hypothese besagt, dass Wörter, die in ähnlichem Kontext auftreten, dazu tendieren, eine ähnliche Bedeutung zu haben. Dieser Erkenntnis folgend, messen diese Verfahren Bedeutungsbezüge, die sich durch die Auftretensmuster von Wörtern in grossen Korpora äussern. Genauer gesagt werden Wörter als hochdimensionale Vektoren repräsentiert wenn die Dimensionen Kontextmerkmale wie z.B. gemeinsam auftretende Kontextwörter darstellen. Der Bezug von zwei Wörtern kann beurteilt werden, indem die ihnen zugehörigen Vektordarstellungen verglichen werden.

Die Beurteilung von Bedeutungsähnlichkeit wird durch Mehrdeutigkeit, also der Eigenschaft von Wörtern, eine Reihe verschiedener Bedeutungen zu transportieren, erschwert. Im Fall von Homonymie haben etymologisch verschiedene Wörter zufällig dieselbe äussere Form, wobei die transportierten Bedeutungen wesentlich voneinander abweichen. Andererseits können polyseme Wörter eine Vielzahl verwandter Bedeutungen aufweisen, die durch die Unterschiede in den prototypischen Kontexten, in denen sie vorkommen, bestimmt werden.

In einem grösseren Kontext ist Mehrdeutigkeit eine der grössten Herausforderungen, der sich alle sprachtechnologischen Anwendungen, von Information Retrieval bis hin zur maschinellen Übersetzung, gegenübersehen. Im Mittelpunkt aktueller Arbeiten steht die Frage, ob distributionelle Verfahren angepasst werden sollen, um bedeutungsspezifische oder kontextabhängige Ähnlichkeiten zu berechnen. So, wie sie ursprünglich formuliert wurden, stellen distributionelle Verfahren Wordarten dar und können daher nicht zwischen verschiedenen Verwendungen oder Bedeutungen eines Wortes unterscheiden. Darüber hinaus kommt in einem Vektorraummodell die Schwierigkeit hinzu, dass die Vektordarstellung eines Wortes all seine verschiedenen Bedeutungen und Verwendungen innerhalb eines ganzen Korpus vermischt.

In der Literatur wurde eine Reihe von Methoden für die Berechnung kontextabhängiger Ähnlichkeiten innerhalb des distributionellen Darstellungsparadigmas vorgeschlagen.

Das Ziel unserer Forschung ist es, Mittel zur Berechnung kontextabhängiger distributioneller Ähnlichkeiten zu untersuchen, die auf Wortbedeutungen sensibilisiert sind. Unser Vorschlag ist, ein Wort hinsichtlich der Verteilung in einer Gruppe Bedeutungskomponenten zu beschreiben, während der Verdeutlichungsprozess, der zur Bedeutung von Wörtern im Kontext führt, als eine Veränderung in der Verteilung dieser latenten Bedeutungsklassen modelliert wird. Der entwickelte framework stellt Wörter, die einzeln oder im Kontext vorkommen, als Wahrscheinlichkeitsverteilungen über eine globale Reihe von Korpus-induzierten Bedeutungskomponenten oder -aspekten. Bei einem Wort ohne Kontext spiegelt diese Darstellung seine a-priori-Bedeutung als eine Verteilung über eine Reihe latenter Bedeutungskomponenten wider. Ist Kontext vorhanden, bestimmt eine Veränderung in dieser Verteilung eine eindeutige Repräsentation, in der Bedeutungskomponenten, die durch den Kontext bestätigt werden, wahrscheinlicher werden. Solche Darstellungen können nun mit einer eindeutigen Interpretation verglichen werden: Wörter, die sowohl alleinstehend als auch im Kontext auftreten, haben eine ähnliche Bedeutung, wenn sie die gleichen latenten Komponenten haben.

Die Bedeutungskomponenten selbst werden ihrerseits wieder unbeaufsichtigt

vom Korpus abgeleitet. Die Erkenntnis, die sich dahinter verbirgt, ist, dass jedes Vorkommen eines Wortes mit einem Kontextmerkmal durch eine latente Bedeutung oder latente Bedeutungskomponente erklärt wird und die Gesamtheit all dieser Vorkommen eines Wortes eine Mischung über solchen latenten Klassen sind. Das Ziel ist es, die Gruppe latenter Klassen abzuleiten, die am besten die Daten der Korpuskookkurenz erklärt. Dafür lehnen wir uns an Hofmanns (Hofmann and Puzicha [1989]) framework für unüberwachtes Lernen aus dyadischen Daten und wir benutzen zwei Variationen dieser Methode, um latente Klassen abzuleiten.

Im Gegensatz zu früheren Arbeiten modelliert unser framework die Bedeutungen von Wörtern im Kontext in einem Wahrscheinlichkeitsrahmen, in dem die Darstellung von Bedeutungen und die Berechnung von Ähnlichkeiten auf natürliche, intuitive Weise erhalten werden. Weiterhin ist der Rahmen völlig modular, da er bei jeglicher Art von Vektorraummodell (VRM) angewendet und mit jedem geeigneten latenten Modell bzw. Algorithmus benutzt werden kann. In dieser Dissertation wurde der framework auf einem VRM auf Wortebene für die lexikalische Substitution sowie auf einem VRM der Paraphrasenerhalt konstruiert. In beiden Aufgabenbereichen erhalten wir vielversprechende Ergebnisse.

Distributional paraphrasing für Fragenbeantwortung

Der zweite Teil dieser Dissertation widmet sich der Anwendung des distributional paraphrasing auf die Fragenbeantwortung. Genauer gesagt stehen zwei Aspekte im Fokus:

1. Wir untersuchen die Einbindung einer Paraphrasierungskomponente in ein Antwortextraktionssystem.
2. Wir testen die kontextabhängige Paraphrasierungsmethode an der Fragenbeantwortung.

Paraphrasen sind Paare von Phrasen, die eine ähnliche Bedeutung transportieren und sich daher gegenseitig ersetzen können, ohne die Bedeutung der Sätze zu verändern, in denen sie vorkommen. Der Zweck von Paraphraseninduktion ist wiederum der, die Herausforderung anzugehen, die sich den meisten NLP-Anwendungen aufgrund der Tatsache stellt, dass die gleiche Bedeutung durch eine Vielzahl verschiedener äusserer Erscheinungsformen ausgedrückt werden kann.

Eine solide Methode, den automatischen Erhalt von Paraphrasen anzugehen, ist die Verwendung distributioneller Verfahren. Der Mechanismus, der dahinter steckt, ähnelt der Berechnung lexikalischer Ähnlichkeit, da (für gewöhnlich kleine) Phrase als atomische Einheiten behandelt werden und ihre Auftretensmuster in grossen Korpora für die Berechnung von Ähnlichkeitsscores zwischen

Phrasen benutzt werden. In dieser Dissertation bleiben wir durchweg innerhalb des Darstellungsparadigmas, das vom DIRT (Discovery of Inference Rules from Text) Algorithmus in Lin and Pantel [2001a] vorgeschlagen wurde. Phrasen sind Wege in Abhängigkeitsgraphen, die in einem Raum von Kontextmerkmalen dargestellt werden, die aus den beiden Wörtern links und rechts von den Satzwegen/-strecken bestehen. Anhand der so erhaltenen Vektordarstellungen werden Phrasenähnlichkeiten gemessen. Ist ein bestimmter Phrase vorhanden, so werden die häufigsten ähnlichen Phrase entsprechend der Ähnlichkeitsergebnisse als Paraphrasen ausgegeben. Ein Beispiel für eine typische Paraphrase, die anhand des DIRT Algorithmus erhalten wurde, ist: X solves Y $\approx$ X finds solution to Y.

Unsere Zielanwendung ist (Question answering - QA). Dies ist die automatische Extraktion von Antworten auf Fragen von grossen Textsammlungen. Eines der Hauptprobleme, dem sich QA gegenübersehen, wird dadurch aufgeworfen, dass die gleiche Bedeutung in Fragen und Antwortsätzen oft unterschiedlich ausgedrückt wird. Die Verwendung von Paraphrasen ist eine der natürlichsten Methoden, dieses Problem anzugehen, und Methoden wie z.B. Lin and Pantel [2001a] wurden mit Anwendungen wie QA im Hinterkopf entwickelt.

Wir untersuchen die Verwendung von Paraphrasierung bei QA, indem wir ein grundlegendes Fragenbeantwortungssystem entwickeln, dass um eine Paraphrasierungskomponente herum aufgebaut wurde. Zunächst untersuchen wir angemessene Verfahren zur Integration von Paraphrasenkomponenten in ein solches System, und zwar die, die den grössten Nutzen aus dem Wissen ziehen, das in solch einer Sammlung enkodiert ist. Wir schlagen ein sprachmodellbasiertes Antwortextraktionssystem vor, das mit einer Paraphrasierungskomponente auf natürliche Weise verbessert werden kann. Im Gegensatz zu vorherigen Arbeiten konzentrieren wir uns auf Stabilität: Anstatt die Paraphrasierungsregeln zu nutzen, um exakte Treffer von Sätzen und Antworten zu erhalten, erlauben wir ihnen, die Distanz zwischen Fragen und Antwortsätzen zu reduzieren.

Desweiteren konstruieren wir den kontextabhängigen Ähnlichkeitsrahmen für die Paraphrasierung und überprüfen die Auswirkungen kontextabhängiger Paraphrasierung auf das Fragenbeantwortungssystem. Unsere Beobachtungen zeigen, dass die Verwendung von Paraphrasierungen insgesamt Verbesserungen hervorbringt, die obwohl sie beachtlich sind aufgrund geringer Abdeckung stark eingeschränkt sind: Während bei den Sätzen, in denen Paraphrasen benutzt werden, starke Verbesserungen beobachtet werden können, ist die Anzahl der Fälle, in denen diese benutzt werden, sehr gering. In zukünftigen Arbeiten haben wir vor, die Einschränkungen, die wir beobachten können, näher zu untersuchen, um herauszufinden, ob es sich dabei wirklich um Einschränkungen der Paraphrasierungsmethode handelt oder ob sie in anderen Faktoren begründet sind.

Chapter 1

Introduction

The need for assessing similarity in meaning is central to most language technology applications. Distributional methods are robust, unsupervised methods which achieve high performance on this task. These methods measure similarity of word types solely based on patterns of word occurrences in large corpora, following the intuition that similar words occur in similar contexts. As most Natural Language Processing (NLP) applications deal with disambiguated words, words occurring in context, rather than word types, the question of adapting distributional methods to compute sense-specific or context-sensitive similarities has gained increasing attention in recent work.

This thesis focuses on the development and applications of distributional methods for context-sensitive similarity. The contribution made is twofold: the main part of the thesis proposes and tests a new framework for computing similarity in context, while in the second part we investigate the application of distributional paraphrasing to the task of question answering.

The remainder of this chapter describes the research context motivating our work and briefly overviews our proposals.

1.1 Context-sensitive distributional similarity

Distributional methods for similarity The capacity to identify words, or larger units of text, which convey similar meaning is of major importance to a large number of NLP applications which require at least a minimal level of text understanding.

Consider, for example, the task of information retrieval, one of the most important and widely-used end-user language technology applications. In information retrieval the goal is to return a set of documents which are most relevant to a query a user has entered. The main issue that information retrieval has to face is that of term mismatch: relevant documents may use different words than the ones present in the query. Query expansion is a popular technique which has been proposed to address this issue. A query expansion module takes a query as an input and generates terms which are related in meaning to the terms of the query. Consider, for example, the following query (from Riezler et al. [2007]):

(1) how to enhance competitiveness of indian industries

Methods for computing similarity in meaning can be employed to obtain similar words such as *increase* or *improve* for the query word *enhance*. These are added to the original set of query terms to form an expanded set of words, which are further used to retrieve documents. This technique has been shown to improve retrieval performance, as it generalizes over the particular lexical choices of a user's query.

Distributional methods are the most robust approach to the task of computing word similarity. These are based on the hypothesis initially formulated by Harris, also known as the distributional hypothesis, stating that words occurring in similar contexts tend to have similar meaning. Following this intuition, these methods measure relatedness in meaning as indicated by patterns of word occurrence in large corpora. More precisely, words are represented as high-dimensional vectors, where the dimensions represent context features, such as co-occurring context words. The relatedness of two words can be assessed by comparing their associated vector representations.

An example of meaning representations employed in a typical vector space model, the most widespread incarnation of distributional methods, is given in Table 1.1. In this representation words to be compared are vectors in a space in which contextual features are co-occurring words.

	relation	condition	quality	further	economic	country
improve	11489	9832	9008	2931	5191	5827
enhance	1392	117	756	2194	2023	1741

Table 1.1: Fragments of the distributional representations for words *improve* and *enhance*. Values represent co-occurrence counts (in a 5 words symmetric window) over the GigaWord corpus.

Context-sensitive distributional similarity The task of assessing meaning similarity is challenged by sense ambiguity, the capacity of words to convey a number of different meanings. In the case of homonymy, etymologically distinct words accidentally share the same surface realization and the conveyed meanings are significantly divergent. On the other hand, polysemous words can exhibit a large number of *related* senses, which are determined by the distinctions in the prototypical contexts in which they appear.

In a larger context, sense ambiguity is one of the major challenges that all language technology applications are faced with, from information retrieval to machine translation. In the context of the information retrieval task, consider for example the query (from Riezler et al. [2007]):

(2) how to induce labour.

Most likely in this case the user is interested in ways of inducing *birth* and is not concerned with *labour* as a *social class* or as in the *Labour Party.* Query expansion techniques have to account for this phenomenon, as expanding with related terms corresponding to other meanings, such as *work* or *employment*, may lead to harming retrieval performance.

The focus of recent work has been the question of adapting distributional methods to compute sense-specific or context-sensitive similarities.

Distributional methods, in their traditional formulation, represent meanings of word types and are therefore incapable of differentiating between different usages, or different senses of a word. Furthermore, in a vector space model this is made difficult by the fact that the vector representation of a word mixes together all its different senses and usages over and entire corpus.

As an example, consider the vector of *labor*, again represented in a space of co-occurring words.

	party	department	child	worker	birth	hospital
labor	24113	22314	6368	2966	82	180

Table 1.2: Fragment of the distributional representation for word *labor* (co-occurrence counts (in a 5 words symmetric window) over the GigaWord corpus)

Distinct usages of *labor* can be identified in this representation. Co-occurrence with words such as *party* and *department* indicate a political context while *birth* and *hospital* have most likely occurred when *labor* was used in the context of *giving birth.* As it can be observed, the data suggests that this sense occurs only to a small degree in the newspaper corpus used. Context-aware vector space models face the task of untangling these mixture vectors in order to obtain meaning-appropriate representations.

A number of methods for computing context-sensitive similarity within the distributional representation paradigm have been proposed in the literature. These distinguish themselves by the approach they take to solving this issue and can be categorized as type-based methods and token-based methods. Type-based methods are characterized by the attempt to shift the meaning of words in context through operations which combine the vector of a target word, the word to be represented, with a vector representation of the surrounding context. In the example above, the approach of a type-based method is to combine the vector of *induce* with that of *labor* in order to obtain a contextualized meaning of *labor*. The intuition behind these methods is that of approximating either a contextualized or a joint meaning through some vector combination. However, most of these methods lack a solid motivation as it often remains unclear if the vector operations proposed are the best ways to model the underlying intuitions of the proposed model.

Token-based methods take a conceptually different approach to this problem as they attempt to distinguish between different meanings of a word's occurrences in a large corpus. Unlike type-based methods, these do not attempt to recover specific meanings from *mixed* vector representations: they rather differentiate *individual word occurrences* that are indicative of distinct meanings. These are further on used to build context-specific, disambiguated, vector representations. In the example above, a token-based method builds a vector for *labor* in the context of *induce labor* only using a subset of all of *labor*'s occurrences: the subset containing instances which are judged to be similar to the current context, (for example *alternative methods of bringing on labor* or *learn about foods that induce labor*).

Our approach Our research goal is to investigate means of computing context-sensitive distributional similarity which are aware of word senses, in the line of token-based methods. However, unlike these methods we propose a unitary, complete probabilistic framework for computing meaning similarity in context.

We propose to describe a word in terms of a distribution over a set of data-induced meaning components while the disambiguation process, leading to the meaning of words in context, is modeled as a shift in the distribution of these latent meaning classes.

More precisely, we use co-occurrence input data to induce a global set of latent classes which intuitively correspond to different usages of words. We further represent an isolated word as the a priori probability of these classes. This distribution reflects its potential meanings, irrespective of a specific context. The latent classes are in turn distributions over context words, reflecting the typical context patterns associated with each class. Given a context feature, we use *posterior* likelihoods for each of these latent classes, this time to represent the context-aware meaning of a word. These representations are further on used to compute meaning similarity, based on the underlying hypothesis that words, isolated or in the presence of context features, exhibit similar meaning if they trigger the same meaning components.

To exemplify this, consider the word *labor* from Table 1.2. When induced from a newswire corpus, the most likely latent class may be a LABOUR PARTY class, defined by context words such as *party*. When given the phrase *induce labour* the distributional representation will reflect as most likely a BIRTH class, signaled by context words such as *hospital*. Such a context-aware representation can further be used to deduce that the word *birth* is an appropriate synonym for *labour* in this case.

A main advantage of our framework is the fact that it is completely modular, as we abstract away from a specific instantiation of the distributional hypothesis, as well as from the algorithm used to induce latent classes.

We test our framework for the task of assessing lexical similarity in context as well as for acquiring context-sensitive paraphrases.

1.2 Distributional paraphrasing for question answering

Despite the increased amount of attention that both distributional paraphrasing as well as context-sensitive distributional similarity methods have received in recent years, studies evaluating their impact in end-user NLP tasks are scarce.

The second part of this thesis approaches the application of distributional paraphrasing to the task of question answering. More precisely, the focus is twofold:

1. We investigate the integration of a paraphrasing component in a answer extraction module.

2. We test the *context-sensitive* paraphrasing method on the task of question answering.

Distributional paraphrasing Throughout this chapter we have discussed distributional methods for lexical similarity, however a long line of research has studied methods for automatic paraphrasing. In paraphrasing, the focus is not on assessing similarity of words, but of larger, more informative units of text.

Paraphrases are pairs of phrases which convey similar meaning, and can therefore substitute each other without changing the meaning of the sentences they occur in. The purpose of paraphrase induction is again that of addressing the challenge posed to most NLP applications by the fact that the same meaning can expressed through a variety of different surface realizations.

Consider for example, the task of Question Answering (QA). In QA a system is given a question and the goal is to extract the answer to the question from large collections of text. The following question-sentence pair is extracted from the data provided by the TREC02 QA (Voorhees [2002]) track:

(3) What does the acronym NATO stand for?

(4) NATO is an acronym, from the initials of the North Atlantic Treaty Organization.

In this example, a system can extract the answer by making use of a paraphrase such as *X acronym stands for Y* $\approx$ *X is an acronym, from the initials of Y.*

A robust way to approach the task of automatically acquiring paraphrases is, again, through the use of distributional methods. The mechanism behind this is similar to that of computing lexical similarity as (typically small) phrases are treated as atomic units and their occurrence patterns in large corpora are used for comparing them. When given a particular phrase, its top most similar phrases, according to the similarity computations, are returned as paraphrases.

The DIRT algorithm proposed by Lin and Pantel [2001a] in particular is one of the most popular methods for paraphrasing and has not been outperformed in accuracy by any other distributional method. The question of context-appropriateness has also been in the focus of recent work on distributional paraphrasing with a number of methods being developed to model context-appropriateness within the DIRT paraphrasing method. In particular the framework we propose in this thesis can also be instantiated on the underlying vector space model employed by DIRT in order to obtain context-sensitive paraphrasing. The second part of the thesis focuses on employing context-sensitive paraphrasing, as well as the original DIRT algorithm for the task of question answering.

Employing distributional paraphrasing for QA Despite the fact that the DIRT algorithm provides a method to easily acquire a relatively[1] accurate paraphrase resource, there has been rather little research on using this method to solve the variability problem in NLP in general and in QA in particular. Furthermore, despite the attention that the large number of its context-sensitive extensions have received, there is no account to this date on employing these for an end-user language technology application.

We focus on the use of distributional paraphrasing for the task of answer extraction in question answering. In particular we remain within the representation paradigm proposed by Lin and Pantel which has been proven to be very effective. Phrases are paths in dependency graphs which are represented in a space of contextual features consisting of the two words to the left and right of the phrase paths.

In more detail, we start by investigating the challenges behind the application of DIRT distributional paraphrasing for answer extraction. We approach this by building a basic question answering system centered around a paraphrasing component. We investigate what are appropriate ways of integrating a paraphrase component into such a system, which make the most use of the knowledge encoded in such a collection. Further on, we instantiate the context-sensitive similarity framework for the paraphrasing task and test the effects of context-sensitive paraphrasing to the question answering system.

[1]The accuracy of the method has been evaluated to be approximately 50% for the most confident paraphrases.

1.3 Overview of the thesis

This thesis consists of two parts: the main part (Chapters 2 to 7) proposes and tests a new framework for computing similarity in context, while in a second part (Chapters 8 and 9) we investigate the application of distributional paraphrasing to question answering. We further briefly summarize the individual chapters:

- Chapter 2 summarizes previous work on distributional methods for the computation of meaning similarity in context. This chapter focuses on *word*-level similarity, while the relevant work on context-sensitive paraphrasing is presented in more detail in Chapter 7.
- Chapter 3 introduces the context-sensitive similarity framework we develop, detailing the proposed vector representations as well as the computation of similarity based on these representations. These are defined in terms of latent classes which are learned from co-occurrence data; this chapter abstracts away from the particular method used to induce the latent classes.
- Chapter 4 presents two models which can be used to induce latent classes. These are initially described in their original formulation, followed by their particular instantiation in our framework.
- Chapters 5, 6 and 7 instantiate the framework proposed for the tasks of word similarity, lexical substitution and paraphrasing in context.
- Chapters 8 starts the second part of this thesis focused on distributional paraphrasing for question answering. In this chapter we overview related work on employing distributional paraphrasing both for QA and for the related task of Recognizing Textual Entailment.
- Chapter 9 defines a basic question answering system enhanced with a syntax-level paraphrasing component. In this context, we instantiate both the previously proposed DIRT algorithm as well as context-sensitive paraphrasing obtained within the framework developed in the first part of the thesis.
- Chapter 10 concludes by summarizing the main contributions of this thesis and discussing directions for future work.

Preliminary versions of the work presented here have been published in Thater et al. [2009] (Chapter 2), Dinu and Lapata [2010a] (Chapters 3 and 7), Dinu and Lapata [2010b] (Chapters 4, 5 and 6) Dinu and Wang [2009] (Chapter 8) and Chrupala et al. [2010] (Chapter 9).

Chapter 2

Distributional similarity. Background and related work

Methods for assessing word meaning similarity are needed by most language technology applications and they become crucial for the tasks requiring a deeper level of text understanding such as question answering.

With the availability of machine-readable dictionaries, a number of dictionary-based methods have been proposed in the literature. These make use of the information available in dictionaries, either in the form of definitions, example sentences or as the relation graph present in a sense inventory such as WordNet (Fellbaum [1998]). As opposed to dictionary-based methods, distributional methods build representations and compute similarity based on the co-occurrence patterns of words, as extracted from large amounts of typically un-processed text. These are widely used, unsupervised methods which have been shown to outperform dictionary-based methods not only in terms of robustness but also in the accuracy they achieve.

Vector space models are the most widely-spread distributional method and have been been introduced as early as with the work of Salton [1971]. In his original proposal, documents and queries are compared based on their representations as highly-dimensional vectors. Distributional methods for representing word meaning follow the intuition that words occurring in similar contexts tend to have similar meaning. In this case, words are represented as vectors in a space of indicative contextual features, such as other co-occurring words.

While traditional distributional methods are targeted at representing word types the question of modeling the meaning of words *occurring in context* is in the focus of recent work, which focuses on building linguistically and empirically motivated models for word meaning on context.

In this chapter we start by briefly introducing distributional methods and in particular vector space models, in their traditional formulation, in Section 2.1. We briefly discuss the main stages in building a vector space model as well as the known limitations of these methods. In Section 2.2 we move further to the issue of context-sensitivity and summarize the main approaches preceding our work.

2.1 Background

Common to all distributional methods is the choice of representing word occurrences as vectors in high-dimensional space, where the dimensions correspond to context features. Context features are usually chosen as linguistic clues indicative of the word's meaning. A typical choice of context features is the use of co-occurring words. Table 2.1 gives an example of a vector representation for the word *professor* as extracted from its occurrence in sentence 1). In this example, context features are words occurring within a window of size five around the target word *professor*.

(1) Once tenured, a professor can largely set his own responsibilities and decide to a large extent how to divide his time between teaching, writing, researching, and administration.

	once	tenured	a	can	largely	set	his	own
professor	1	1	1	1	1	1	1	1

Table 2.1: Non-zero dimensions of the *professor* vector extracted from sentence 1).

Vector space models are widely-used methods for computing word similarity with applications ranging from information retrieval to essay grading. Specific to vector space models, as a distributional method, are word *type* representations. These are obtained by adding the vectors of each individual occurrence of a word, over a large collection of text. Such a vector describes a word's patterns of use by summing up over all its occurrences. To exemplify, the word *professor* is represented in a vector space model extracted from GigaWord as in Table 2.2; we list only the top most frequently co-occurring context words as dimensions.

	university	law	science	political	associate	college
professor	42434	13615	7684	6257	6055	6011

Table 2.2: Dimensions with highest values of the *professor* vector extracted from GigaWord.

For the rest of this section we overview the stages involved in building a vector space model and we discuss the main limitations of this approach. A detailed account of vector space models together with their applications to NLP tasks can be found in Turney and Pantel [2010] .

Main stages in building vector space models There are four steps in building a vector space model: 1) extraction of the input vector representations, 2) weighting of the vector components, 3) dimensionality reduction and 4) similarity computations.

Input vector representations The most important step in building a vector space model is the choice of input vector representation (Turney and Pantel [2010]). This stage consists in building an input frequency matrix, which contains words represented over a set of meaning-indicative context features.

A large number of alternative representations exist, each of them usually targeted at a specific application. Information retrieval is the first, and one of the most prominent applications of vector space models (Salton [1971], Salton et al. [1975] Deerwester et al. [1990] Landauer and Dumais [1997] Finkelstein et al. [2002]). In information retrieval, vector spaces represent words in terms of the documents in which they occur, leading to word-document input matrices. These can be used to compute word similarity as well as document similarity with applications such as document retrieval and classification.

Weighting is the second stage in building vector space models. Weighting schemes are functions on the vector values which are applied to overcome bias that raw counts might introduce. To give an example, *say* or *tell* as context words are very frequent in newspaper corpora and they are not indicative clues of word meanings: in this domain they are too frequent to signal meaning similarity. Examples of widely-spread weighting schemes are point-wise mutual information or tf-idf (term frequency-inverse document frequency). A detailed account of these can be found in Turney and Pantel [2010] and Zhitomirsky-Geffet and Dagan [2009].

Dimensionality reduction methods aim at reducing the noise encountered in the input data in order to achieve higher accuracy in computing similarity. The goal is to reduce the dimensionality of the original space to obtain representations over a small set of *concept*-like dimensions. Latent semantic analysis (Deerwester et al. [1990] and Landauer and Dumais [1997]) is one of the most widely-used methods for dimensionality reduction. Not all vector space models perform this step, a popular alternative being to retain as context features only the most frequent or most indicative words (according to some weighting scheme).

Similarity computations are the final stage of building a vector space model. A number of vector similarity functions have been proposed in the literature. One of the most popular measures is cosine similarity which measures the angle between two vectors. Other measures commonly used are inverse divergence methods, Dice coefficient, Lin similarity (Lin [1998a]), etc. An overview of their properties can be found in Turney and Pantel [2010].

Instances of vector space models A large number of vector space models for word meaning similarity have been proposed as alternatives to the most basic models, which define words by co-occurrence with documents or with other words.

Syntax-based methods propose the use of dependency-based context-features in order to obtain more informative word meaning representations (Grefenstette [1994], Lin [1998b]). These have been shown to outperform their simpler predecessors on tasks such as detecting synonymy or inducing first-sense heuristics (Pado and Lapata [2007]).

A number of vector space models using different input representations have been recently evaluated on a word similarity task in Agirre et al. [2009]. They conclude that tighter definitions of context as well as syntactic features capture tighter semantic relations such as synonymy while the bag-of-words approaches find topically related words. The study also highlights once again the high performance of vector spaces which are not only more robust in nature but are also shown to be capable of outperforming dictionary-based methods.

In general, vector space models go beyond the representation of words or of documents and can be used as a general paradigm in a variety of tasks. An example of this is the use of vector space models for the task of categorizing relations between pairs of words such as in Turney and Littman [2005] and Turney [2006]. The vector space in Table 2.3 is an example of this and was proposed by Baroni and Lenci [2009] for the tasks of recognizing SAT analogies and semantic relation classification.

	in	at	with	use
teacher school	11894.4	7020.1	28.9	0.0
teacher handbook	2.5	0.0	3.2	10.1
soldier gun	2.8	10.3	105.9	41.0

Table 2.3: Fragment of the *Concept* $\times$ *Concept by Link* space proposed by Baroni and Lenci [2009] using point-wise mutual information weighting.

In this example, the words *(teacher, handbook)* stand in a similar relation to the words *(soldier, gun)* and this is signaled by their occurrence with words such as *with* and *use.*

A vector space model which is of particular relevance to our work is introduced by the DIRT (Discovery of Inference Rules from Text) algorithm in Lin and Pantel [2001b] and Lin and Pantel [2001a]. We further briefly overview the underlying vector space model, which is presented in greater detail in Chapter 7.

The DIRT algorithm introduces a method for extracting paraphrases such as

X solve Y $\approx$ *X find solution to Y*. The method has been developed for NLP applications that deal with the variability problem, i.e. the fact that in natural language, the same meaning can be expressed in various ways.

The exact representation of a phrase (also called pattern), is that of a binary relation with noun arguments, extracted as a path in a dependency graph.

The core of the DIRT method is a vector space model. In this vector space model, each pattern is represented in two spaces: by its co-occurrence with left hand side (X) and with right hand side (Y) noun fillers in a large corpus.

Table 2.4 exemplifies the DIRT input representation as extracted from the XIE fragment of GigaWord. In the X-filler space phrases *X solve Y* and *X settle Y* share common nouns such as *country* or *china* while in the Y-filler space the nouns *issue* and *problem* occur most frequently.

X	country	china
X $\xleftarrow{subj}$ *solve* $\xrightarrow{obj}$ Y	556	108
X $\xleftarrow{subj}$ *settle* $\xrightarrow{obj}$ Y	211	379

Y	issue	problem
X $\xleftarrow{subj}$ *solve* $\xrightarrow{obj}$ Y	988	481
X $\xleftarrow{subj}$ *settle* $\xrightarrow{obj}$ Y	605	377

Table 2.4: Fragment of the DIRT vector space. Paths in dependency graphs are represented in the space of co-occurring left and right filler nouns.

The patterns are compared in the X-filler space, and correspondingly in the Y-filler space by using the Lin similarity measure introduced in Lin [1998a]. This is preceded by the use of point-wise mutual information as a weighting scheme. The final similarity score between two patterns is obtained by multiplying the X and Y similarity scores. Further on, this similarity is used for building a paraphrase collection by following two steps: 1) extract a large collection of patterns from a corpus and 2) paraphrase each of these patterns by returning its top most similar patterns, according to the similarity score.

Limitations of distributional methods Each choice for an input distributional representation can be thought of instantiating a different variant of the distributional hypothesis. As the framework we develop throughout this thesis abstracts away from any particular choice of input representation, we will not be concerned with the validity of the distributional hypothesis. In general, we assume that the linguistic units to be compared are similar if they occur with similar context features, as defined by the choice of input representation. However, the main limitations of vector space models stem precisely from the limitations of the distributional hypothesis.

A classic example is the case of synonymy and antonymy: some antonymous words are mistaken for synonyms because they also tend to occur in similar contexts. Differences between them are difficult to capture, irrespectively of the choice made in defining contextual features. In general, vector space models encounter difficulties in distinguishing between synonymy and other semantic relations such as hyper/hyponymy, *instance of* relations or simple topical relatedness. Various vector space models have been proposed for targeting some of these specific relations, leading to observations such as the fact that small context window correlates with detecting tighter semantic relations such as synonymy. A special case is the case of asymmetric relations, such as hyper/hyponymy or uni-directional entailment which have received particular attention (Bhagat et al. [2007], Erk [2009]). In general, despite the advances that have been made, directionality is not easy to determine and one of the reasons for this is that it does not correlate very well with the subset relation. To exemplify this, consider the word *cat* which is a hyponym of *life form*: it is unrealistic to expect the contextual features of *cat* occurrences to be a subset of those of *life form* occurrences.

A major limitation of vector space models, which has gained increasing attention, is caused by sense ambiguity: a word is represented as a type vector which mixes together the different senses it exhibits. However, most applications need to assess the meaning similarity of words occurring in context, which gives rise to the question of meaning-aware, context-sensitive vector representations. The same concern has been the focus of work extending the original DIRT algorithm for distributional paraphrasing: as initially pointed out by the authors, paraphrases accompanied by a description of the context in which they apply should intuitively form a more accurate, reliable knowledge resource.

2.2 Context-sensitive distributional similarity

Polysemy is an issue to most NLP applications, which typically devise implicit methods to perform meaning ambiguity resolution. It has been observed, however, that in many of these applications the question of meaning disambiguation can be in fact reduced to that of computing context-sensitive, meaning-aware similarity. For this reason, in recent years, the assessment of context-sensitive similarity (in both mono-lingual and cross-lingual settings) has been introduced as a stand-alone task (McCarthy and Navigli [2007]).

Traditionally Word Sense Disambiguation (WSD) has been proposed as stand-alone task to help the development of meaning resolution modules to be integrated in end-user NLP applications. It is commonly agreed on that WSD in its traditional formulation has failed to prove itself useful in its targeted applications (Resnik [2006]).

As opposed to WSD, context-sensitive similarity assessment as a stand-alone task circumvents the use of sense inventories as well as the controversial underlying principle of meaning resolution as a strict word-sense assignment. The use of distributional methods for context-sensitive word similarity is a way to approach this task in an unsupervised manner, without the use of a dictionary or of labeled data.

For the remainder of this section we summarize the main contributions to the problem of context-sensitive meaning similarity. Context-sensitive similarity methods distinguish themselves by the approach to solving this issue. Type-based models, which we overview in Section 2.2.2, use vector spaces as input and try to shift the meaning of words in context through operations which combine the vector of the target word with the vector of context. On the other hand, token-based methods make use of individual occurrence vectors, richer information that is lost in some vector space representations. These methods try to rather distinguish between subsets of a word's occurrences which are indicative of the specific meanings. These are presented in Section 2.2.3.

Finally, a related question in the focus of recent work is that of compositional methods for distributional meaning similarity. These aim at building vector representations of sentences as functions of the representations of the composing words, much like in the tradition of formal semantics. These will be discussed in Section 2.2.4.

2.2.1 Type-based methods for context-sensitive similarity

Type-based methods for context-sensitive distributional similarity build contextualized vector representations for words and employ these for performing context-aware similarity computations. These methods implement two main stages in obtaining contextualized meaning representations: 1) initially a vector space model is built and 2) given a target word and a context word, their vectors are combined in order to obtain a context-sensitive vector representation.

The challenge faced by type-based methods is determined by the vector representation of a word as a mixture of different usages in a large corpus.

Consider, for example, the fragment of the vector space representation for the word *heavy* and two synonyms corresponding to two distinct meanings: *frequent* and *fat*, given in Table 2.5.

	use	drug	pound	american	bombing	attack
frequent	738	181	14	225	188	929
heavy	3199	778	655	453	778	1707
fat	43	23	367	140	0	4

Table 2.5: Fragment of vector representations for words *frequent*, *heavy* and *fat*. Values are co-occurrence counts within a symmetric context window of size 5 over the GigaWord corpus.

As it can be observed, the similarity between *heavy* and *frequent* is reflected only in a fragment of these vectors, by co-occurrence with words such as *use* or *drug*. The words *pound* or *american* as context features signal a different meaning for which *fat* is an appropriate synonym. Type-based approaches develop methods to shift these mixed vector representations to meaning-specific vectors using the given context. The methods proposed in the literature vary with respect to the input vector space model as well as with the composition method. We overview these for the remainder of this section.

Mitchell and Lapata [2008], Mitchell and Lapata [2009], Mitchell and Lapata [2010] In Mitchell and Lapata [2008] the authors propose a general formulation for vector composition which fits most of the type-based methods.

The composition p of two constituents, is obtained as a function of the two vectors of the constituents u and v, R, the relation they stand in and K, additional, background knowledge that may influence the meaning composition:

$$p = f(u, v, R, K)$$

This was proposed as a framework for vectorial composition, where meaning representations of larger units of text are obtained from the vectors of its components. However, the representations derived can also be used as contextualized representations of words.

The authors build representations for short sentences consisting of intransitive verbs and their subjects such as 2) and 3):

(2) The fire glowed.

(3) The face glowed.

The contextualized representations are evaluated based on their performance on predicting human similarity judgments. More precisely, the system has to return a high similarity score between sentence (2) and the word *burn*, as *burn* is a synonym of *glow* in this context. In sentence (3), however, *burn* is not a meaning-preserving substitute.

Throughout this study, the authors use a simple bag-of-words vector space and test a number of vector addition and multiplication-based models: simple component-wise addition, weighted addition and addition including distributional neighbors as proposed by Kintsch [2001], component-wise multiplication and a combination model of both addition and multiplication. All the methods tested are naive instantiations of the general framework proposed, as they ignore the syntactic relation and background knowledge components.

The experimental results show that models performing point-wise multiplication of component vectors outperform all additive methods and perform similar to the combination method. Multiplication requires no parameter tuning and therefore no optimization, which makes if preferable to the combination method. The intuition behind the component-wise composition methods is that multiplication approximates the intersection of the meaning of two vectors, whereas addition their union. This is detailed in Mitchell and Lapata [2009]. Given a word vector u, its components are weighted such that $u_i = \frac{P(c_i|u)}{P(c_i)}$, where c_i is a context feature, the i'th dimension of the vector space. Following simple calculations they obtain that the components of the multiplied vectors give the probability of each context feature given both u and v :

$$u_i v_i = \frac{p(c_i|u)}{p(c_i)} \frac{p(c_i|v)}{p(c_i)} = \frac{p(c_i|u,v)}{p(c_i)}$$

In Mitchell and Lapata [2009], the authors further show that these simple models yield improvements in language modeling. The results are supported by Mitchell and Lapata [2010], where they test three types of word combinations: noun-noun, verb-object and adjective-noun. Again, the multiplicative model is the best parameter-free method for all these combinations and it is only slightly outperformed by more complex composition functions such as weighted average and dilation.

Erk and Padó [2008], Erk and Padó [2009] In Erk and Padó [2008] the authors focus on using syntax for addressing the same issue of context-sensitive distributional similarity.

Their model is guided by syntax in two ways: 1) by using a syntactic vector space model as input representation and 2) by employing selectional preferences to contextualize occurrences of target words.

To exemplify their method, consider the meaning of a verb in the presence of its object as in the phrase *catch a ball.* The meaning of the verb is modeled using the verb's vector together with the vector capturing the inverse selectional preferences of the object; the latter is computed as the centroid of the verbs that occur with this object. More precisely they compose the vector of *catch* with vectors of other verbs that take *ball* as object (such as *throw* or *toss*). This way the hope is that the vector of *catch* will be shifted towards its correct meaning and further way from wrong meanings as in *catch a disease.* The intuition behind this is that the meaning of a verb is a combination between its overall meaning (its type vector) and the expectations that its object enforces on the verb.

The exact vector composition they propose involves raising the selectional preference vector to a power $n = 30$, followed by point-wise multiplication with the target word vector. Their method is tested on the Semeval Lexical Substitution task. To build this data set, annotators are presented with a set of target words occurring in sentential context and they are asked to provide appropriate substitutes for these words in each of these contexts. The authors test the contextualized representations on the task of ranking the set of all possible substitutes for a target word: for each data sentence, the appropriate substitutes have to be ranked higher than substitutes corresponding to different meanings. However, the subsequent study in Erk and Padó [2009] shows that good parameters for this computational model are difficult to estimate and might vary to a great extent from one data set to another.

Thater et al. [2009], Thater et al. [2010] Thater et al. [2010] and Thater et al. [2009] address the same issue, again in a syntactic space, but distinguish themselves from previous work by proposing the use of distinct vector representations for predicates and arguments. In their models the focus is on building complex, informative input vector space representations, which are then followed by very intuitive and straightforward composition operations.

More precisely, they choose to represent verbs in a second order syntactic vector space where the dimensions are triples containing two syntactic relations and a verb lemma. For example, *catch* is represented using basis elements such as: *(obj,obj,toss)*. The values are given by second-order co-occurrence counts of *catch* and *toss* as guided by syntax, in this case co-occurrence counts with arguments that are objects of both verbs. Nouns are represented in a first order space: for example *ball* is described by the co-occurrence frequency with the context feature *(obj, toss)*, i.e. the number of times it is an object of *toss*.

In Thater et al. [2009] the actual vector composition is defined in a very natural manner. The meaning of a verb is obtained by restricting its vector to the features active in the argument noun. More precisely, dimensions with value larger than zero in the argument noun are kept intact while all others are set to zero. Unlike in Erk and Padó [2008], the composition is not syntax-aware, as the syntactic information is encoded only in the vector space representation. Thater et al. [2010] implements slight changes to this model, leading to large performance gains: the composition is component-wise vector multiplication, similarly to Mitchell and Lapata [2008], and it is guided, this time, by the syntactic relation between the two words to be composed.

The evaluation setting is that of Erk and Padó [2008] and the experimental results obtained in both Thater et al. [2009] and Thater et al. [2010] significantly outperform previous methods.

2.2.2 Token-based methods for context-sensitive similarity

Token-based methods take a different approach to the task of context-sensitive similarity. Unlike type-based methods, which attempt to contextualize the already mixed, type vector representations (also called prototypes), token-based methods make use of the evidence present in individual occurrence vectors (also called examplar vectors). Individual occurrence vectors are vectors extracted from a *single* occurrence of a word; these representations are lost in vector space models, which sum up over all occurrences of a word.

More precisely, given a word occurring in context, token-based methods contextualize meaning (i.e. build context-sensitive representations, or compute context-sensitive similarities) as guided by similar occurrences of the word in the input corpus.

Consider, for example, contextualizing the meaning of the word *heavy* in the context of *heavy user* as in sentence 4). In token-based methods this is only guided by occurrences of *heavy* in contexts similar to the current one. An example of a similar context is the one in sentence 5), as indicated by overlapping words such as *use*:

(4) Some heavy users develop a psychological dependence on cannabis.

(5) Heavy marijuana use doesn't damage brain.

While following the same general intuition, token-based methods can vary to a great extent in their exact instantiations. For the rest of this section we overview two recent token-based methods for computing word similarity in context, as well as the closely-related methods of attaching contextual preference classes to DIRT paraphrases.

Reisinger and Mooney [2010] In Reisinger and Mooney [2010] the authors propose to represent words, isolated, or occurring in context as sets of sense-specific vectors (multi-prototype representation).

Specifically, a word's occurrence vectors are clustered to produce groups of similar context vectors. Following this, an average prototype vector is computed separately for each cluster, as the centroid of the cluster's elements. Such a prototype vector is to be seen as a description of the contextual features associated to a cluster. In the example above, the two occurrences of *heavy* would be clustered together, along with other similar instances. The centroid of these *heavy* vectors, i.e. the prototype vector, will ideally reflect the meaning of *frequent.*

As words, both isolated and in context, are represented as sets of vectors, the authors introduce new ways of measuring similarity. A first method (AvgSim) computes similarity as an average over all pairings of prototype vectors (i.e. cluster centroids). A second method, MaxSim, returns the similarity of the most similar two clusters. The intuition behind this is that while AvgSim computes the overall similarity (in the direction of type-based methods), MaxSim performs mutual disambiguation as it only selects the most similar clusters and returns their similarity. In the case of words in context, the distance between prototype vectors is weighted by the probability that the context belongs to the prototype's cluster.

On a task of predicting human assessments of word similarity (words occurring in isolation) the authors show that the multi-prototype method outperforms classic token- and type-based models. The context-sensitive method is evaluated on the task of near-synonym generation for words occurring in sentential context. On this task, their approach outperforms a context-ignoring method for words occurring in infrequent senses, while no difference is observed for words occurring in their majority sense.

Erk and Padó [2010] A similar idea is implemented in Erk and Padó [2010] where the authors propose an exemplar-based model for capturing word meaning in context. More precisely, in contrast to the multi-prototype approach, no clustering takes place, as it is assumed that there are as many senses as there are instances. An isolated word is represented by its individual token (exemplar) vectors. Given a sentential context, the meaning of the target word is represented by a *subset* of the individual occurrence vectors of the word. More precisely, the relevant, similar occurrence vectors are obtained through an activation process which is defined in terms of similarity to the current sentential context. For example the vector of *heavy* in (5) would be activated for the target word *heavy* in (4).

The authors experiment with different forms of activation as well as with different type of contextualization: of the target word, of the candidate substitute or of both. Their best results outperform the method proposed in Erk and Padó [2008] although no syntactic component is used for this. However, this paper only presents preliminary work, as the results are obtained by tuning the model's parameters (such as the activation threshold or type of activation) on the test data.

Context-sensitive extensions of DIRT The issue of context-appropriateness of distributional similarity computations has also been highlighted in the context of the DIRT method for paraphrasing proposed by Lin and Pantel [2001b]. Although attempting to solve the same problem as the context-sensitive vector space models, the research on context-appropriate DIRT paraphrases has been carried out mostly independently of this.

In this context, it has been observed that the accuracy of the paraphrases extracted is highly dependent on the context in which they are used. The problem to be solved has been formulated as that of determining if a particular paraphrase rules is appropriate or not in a given context. For example, consider the rule X *is charged by* Y $\approx$ Y *announced the arrest of* X. This is correct in a context such as 6), however not in 7):

(6) The prosecutors announced the arrest of Terry Nichols.

(7) My account was charged by the gas station.

The approaches to this problem are in the line of token-based methods for word similarity in context. Pantel et al. [2007], Basili et al. [2007], Szpektor et al. [2008] and Connor and Roth [2007] focus on making DIRT rules context-sensitive, all following the original proposal of Lin and Pantel [2001b], which we summarize here. These methods are further described in more detail in Chapter 7.

The methods proposed in this context attempt to solve the problem by attaching semantic classes to the X and Y slots of an inference rule. The semantic classes are indicators of the correct context, and an instantiation of a rule is judged as correct if the X and Y fillers belong to the attached semantic classes.

We will further detail the method described in Pantel et al. [2007] as subsequent work closely follows their methodology.

The initial step in this method is to acquire a paraphrase database, using the DIRT algorithm. The second step involves learning semantic classes for the X and Y slots by grouping together semantically similar nouns. For this, Pantel et al. [2007] build a set of semantic classes using WordNet in one case, and the CBC clustering algorithm, in the other. Following this, given a specific inference rule, only the semantic classes that can be associated to the rule are selected. These are chosen based on the filler nouns common to both patterns. For the example above, given the inference rule *X is charged by Y, Y announced the arrest of X*, semantic classes attached to the rule may be: *X: Person, Y: Law Enforcement Agent.* Further on, given an actual instance such as *X: Terry Nichols, Y: prosecutors*, the degree to which X and Y belong to the attached semantic classes is used as an indicator of the rule's correctness in this context.

A number of confidence scores are estimated during each of these stages, such as the similarity of the original DIRT rule, the confidence of the attached semantic classes or the degree to which a noun (a rule instantiation) belongs to such a class. A final score is computed as an aggregation of these scores. The methods differ mostly with respect to the initial step of building semantic classes as well as w.r.t. the computation of confidence scores and their aggregation.

All these methods show improvement over DIRT by evaluating on occurrences of rules in context which are annotated as correct/incorrect by human participants.

Although presented in rather distinct manners all the token-based methods use the same methodology for approaching this issue, which can be summarized as in Table 2.6. In this table *disambiguate* denotes a generic function which returns a common disambiguated representation based on the overlap of distributional features. This together with a measure for distributional similarity, *sim*, are used in different ways to compute context-sensitive similarity by all these methods, as depicted in Table 2.6.

Pantel et al. [2007] Connor and Roth [2007] Basili et al. [2007] Szpektor et al. [2008]
•$sim(u, v) \wedge sim(c, disambiguate(u, v))$
Erk and Padó [2010]
•$sim(u, disambiguate(v, c))$
•$sim(disambiguate(u, c), v)$
•$sim(disambiguate(u, c), disambiguate(v, c))$
Reisinger and Mooney [2010]
•$sim(disambiguate(u, c), disambiguate(v, c))$
•$\sum_{c_i} sim(c, c_i) sim(disambiguate(u, c_i), disambiguate(v, c_i))$
•u, v : words (phrases) to be compared
•c : context in which u, v occur
•$sim(u, v)$: distributional similarity score
•$disambiguate(u, v)$: common disambiguated representation based on overlap of distributional features

Table 2.6: Summary of token-based methods for distributional similarity in context

2.2.3 Compositional distributional representations

Compositional frameworks using distributional representations aim at obtaining vector representations for larger units of text, or even entire sentences, from individual word vectors.

Early methods for distributional representations of sentences are shallow approaches based on simple vector operations. Variants of addition-based composition, in which the vectors of larger units of text are the sum of the individual vectors, are common to many of these and in information retrieval this is still one of the most widely-used methods for representing queries or documents (Widdows [2008]). Similarly, in other applications, the same idea is used for word sense discrimination (Schuetze [1998]) or for representing a sentence as the addition of word vectors in a LSA space (Landauer and Dumais [1997]). Kintsch [2001] proposes a variant of this in which the representations are strengthened with the use of distributionally similar neighbors, which are also added to the target word vectors.

These naive methods are unable to capture even basic differences in meaning such as the fact that *the man bit the dog* is not the same as *the dog bit the man.* Recent work such as Clark and Pulman [2007], Clark et al. [2008] and Grefenstette et al. [2011] focus on comprehensive theoretical frameworks that allow for these distinctions to be made. The explicit goal of these methods is to bridge the gap between traditional formal semantics which is concerned with functional composition of words while paying little attention to the lexicon, and vector space models as empirical models for lexical meaning. In itself a very ambitious task, a solution to this problem would be invaluable to the large number of NLP applications that require the assessment of similarity between fragments of text.

From a theoretical perspective, a lot of progress has been made in these studies. However the question of obtaining this theoretical model from data, which is the key aspect of using distributional representations, remains unanswered. In general the interpretation of distributional methods for modeling lexical meanings is straightforward: the vector associated to a word can be interpreted as a summary of its occurrence patterns. However, despite the recent advances made, there is still no consensus on what is the appropriate interpretation of representing a sentence as a vector, or how can this representation be translated into accurate similarity computations. As noted by Widdows [2008], important questions remain unanswered such as how to represent the complex information available in a traditional parse tree as a point in space.

Our goal is more modest and these question need not be addressed in the context of our work, as we only model *lexical* meaning as determined by context.

2.2.4 Summary

Throughout this section we have summarized type-based and token-based methods for word meaning similarity in context. These distinguish themselves by the input data they use as well as with respect to the underlying principles guiding them. While token-based methods use, in line with vector space models, frequency matrices as input, which sum up over all of a word's occurrences, token-based methods store and make use of word vectors obtained from individual occurrences. Type-based methods, combine the mixed representations in order to obtain disambiguated, context-specific ones, while token-based methods build disambiguated representations by identifying and selecting only the corpus occurrences which are similar to the current context.

The framework we propose can be categorized as a type-based method as it uses a typical vector space frequency matrix as input data. However, unlike most of the previous methods, instead of trying to "guess" and empirically verify ways of composing word vectors, we derive a motivated, probabilistic model for contextualized meaning representations.

Conceptually, our work comes close to token-based methods which also assume an underlying set of meanings, an underlying structure in the occurrence patterns. However, unlike these methods, we build on the assumption that a latent, hidden layer of meaning components can be induced solely from the input frequency matrix; this task can be formulated within a complete probabilistic framework which induces a set of latent classes which best approximate the input frequency data. This is an unitary approach which avoids dividing the problem of context-sensitive similarity into separate components, such as in previous work, which are difficult to tune both individually as well as a combined system.

Chapter 3

Distributional similarity via latent senses

In vector space models, the vector representation of a word “mixes” together its different senses and can be thought of as a summary of the word’s usages over an entire corpus. For this reason, special methods need to be devised in order to obtain disambiguated vector representations which distinguish between different usages, or meanings of a word.

This chapter describes the general framework we propose for building context-sensitive distributional representations and computing meaning similarity. More precisely, we use a co-occurrence input frequency matrix to induce a set of latent classes. We represent a word type as a probability distribution over the induced classes. The intuition behind this is that the distribution reflects the potential meanings of a word, irrespective of context. The latent classes are in turn distributions over context words, reflecting the typical context patterns associated with each class.

Given a context feature, the context-aware meaning of a word is represented this time by the *posterior* probabilities of the latent classes, conditioned on the context. These representations are further on used to compute meaning similarity, based on the underlying hypothesis that words, isolated or in the presence of context features, exhibit similar meaning if they trigger the same meaning components.

Our approach follows Hofmann’s proposal for unsupervised learning from dyadic data introduced in Hofmann and Puzicha [1989]. We start this chapter by summarizing Hoffman’s framework in Section 3.1. In Section 3.2 we detail on the main stages of the method we propose: 1) extraction of the input data 2) building the contextualized representations and 3) performing similarity computations. Finally in Section 3.3 we highlight the relation to previous work.

3.1 Discovering latent meaning components as structure detection

In Hofmann and Puzicha [1989], Hofmann et al. [1998] and in subsequent studies such as Hofmann [1999], the authors propose a framework for learning from dyadic data. Dyadic data consists of a multiset of observation pairs, or *dyads*, (x, y) from a domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are abstract sets of objects; each dyad (x, y) reflects one co-occurrence of x and y.

Hoffman points out that dyadic data is present in a large number of real-world applications. These range from NLP applications to computer vision or preference analysis and consumption behavior. In NLP, dyads can be occurrences of a word in a document or occurrences of words with contextual features. In consumption behavior analysis for example, these may reflect the preferences of individuals ($\mathcal{X}$) for various objects ($\mathcal{Y}$).

Hofmann and Puzicha [1989] propose a unifying framework for statistical, unsupervised learning from such data, which is based on a latent class model. A latent class model assumes the data is explained by a hidden structure and specifies a joint probability over the observation data and the latent structure. As Hoffman points out, this approach can be used to solve two problems: the one of *statistical modeling* and that of *structure detection.* In *statistical modeling* the goal is to learn a joint distribution over the two variables $\mathcal{X}$ and $\mathcal{Y}$. This is not trivial due the data sparseness problem: in real world data, most of the pairs in $\mathcal{X} \times \mathcal{Y}$ are never observed. The joint probability can thus be estimated more reliably by marginalization over the latent variables. *Structure detection* is obtained by estimating the posterior probabilities of the latent structure, given observations.

Hofmann and collaborators propose a number of flat latent class models as well as a hierarchical one. The most simple of these models is known as the *aspect-based* model. In the aspect model, one assumes a latent variable over a finite countable set of *aspects* $\mathcal{A} = \{a_1, ..., a_K\}$ and each dyadic observation (x_i, y_j) is paired with a latent variable realization a_k. The observation data is therefore defined as a discrete mixture over latent variables: $P(x, y) = \sum_a P(x, y, a)$.

The latent structure can thus be seen as partitioning the set of observations into K classes. There is a main difference to hard clustering models as different occurrences of the *same* dyad are distinct objects and may therefore be associated to different latent classes. The variables x and y are assumed to be conditionally independent given a latent class a, therefore the probability of observing a dyad can be factorized as $P(x, y) = \sum_a P(x)P(a|x)P(y|a)$. Given this model, Hoffman proposes maximal likelihood estimation, i.e. the estimation of the latent structure (the parameters $P(x)$, $P(a|x)$ and $P(y|a)$) which maximize the likelihood of the observation data.

We formulate the initial step of building isolated and contextualized meaning representations as a problem of structure detection. When analyzing data containing occurrences of words and context features, it is natural to assume we can induce a structure of classes associated to meaning aspects or meaning components, as derived from distinctions in usage patterns. Following the structure detection step, we propose to represent the meanings of words as distributions over the induced latent classes.

Throughout this thesis we use the aspect-based model together with an extension of this proposed in Blei et al. [2003] in order to induce latent classes. The remainder of this chapter describes the three main stages of the framework we propose. The concrete specification of the latent variable models together with the algorithms used for inducing them are the focus of Chapter 4.

3.2 Vector space framework for meaning similarity in context

As discussed in Chapter 2 the steps of building a vector space model can be summarized as: 1) extract an input frequency matrix from a large corpus, 2) apply weighting scheme to decrease the effect of very frequent un-informative context features, 3) perform dimensionality reduction to obtain a noise-reduced representation and 4) perform vector similarity computations.

The framework we propose is composed of three main stages. Similarly to classic vector space models we start by extracting an input matrix. The second stage is that of building vector representations. We propose a target linguistic unit (occurring with a context feature or in isolation) to be represented as a probability distribution over a total set of latent classes; the extraction of latent classes is formulated as the structure detection problem in the context of dyadic data. This step comprises steps 2) and 3) of traditional vector space models. The representation we obtain is a proper distribution over a set of classes corresponding to latent, corpus-specific, concepts; therefore it is dimensionality reduced and makes weighting schemes unnecessary. The third and final stage is that of similarity computations.

3.2.1 Input frequency matrix

In vector space models, the input (frequency) matrix is a general structure composed of vector representations for a set of linguistic units to be compared (for example words for assessing word similarity or phrases for the task of paraphrasing). Throughout this chapter we use the terminology of *target items* which are represented in a space of *context features* to denote the rows and columns of the input frequency matrix.

The input matrix is extracted from a corpus and contains co-occurrence counts

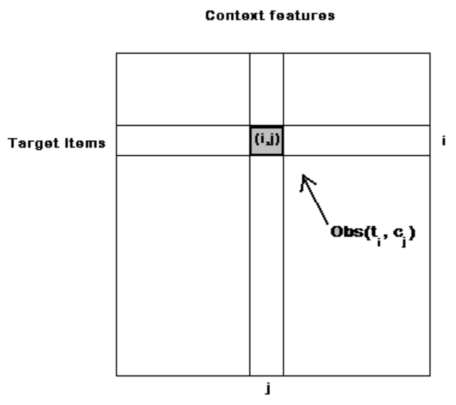


Figure 3.1: Input frequency matrix

of target items occurring with context features. Specific input representations in vector space models can be seen as instances of this general structure. Examples of targets can be simple word types, if the goal is to acquire lexical similarity, or fragments of text with an internal structure such as paths in dependency parse graphs, for the task of acquiring paraphrases. Examples of context features are document labels, typically employed for information retrieval tasks or co-occurring words in simple bag-of-words spaces for word similarity. Models aiming at capturing tighter, specific, semantic relations can employ more complex features based on syntactic structure. We assume this input structure to satisfy the distributional hypothesis. A specific hypothesis is not to our interest in this chapter as we only assume an abstract property of the input matrix, stating that *target items* occurring with similar *context features* have similar meanings.

Input frequency matrices are typical examples of dyadic data. More formally, we can consider a set of target items $\mathcal{T} = \{t_1, ..., t_I\}$, a set of context features $\mathcal{C} = \{c_1, ..., c_J\}$ and a multiset $\mathcal{S}$ consisting of observations pairs $(t_i, c_j) \in \mathcal{T} \times \mathcal{C}$. The observation data $\mathcal{S}$ can be alternatively seen as a $I \times J$-dimensional matrix which sums up the occurrences of each distinct observation pair in $\mathcal{S}$.

For the rest of this chapter, we will use notation t_i with $i : 1..I$ to stand for targets and c_j with $j : 1..J$ for context features. We denote an input matrix by V, a $I \times J$ matrix containing the total set of targets, indexed by the set of contextual features.

3.2.2 Representation over latent senses

As detailed in the previous chapter, despite the wide-spread use of vector space models, it has been noted that representing words as a sum of their contextual features over a large corpus has a major drawback as it ignores sense ambiguity. Such representations can only reflect the *mixture of usages* of a target word, while many applications require the similarity assessment of words occurring in context, of disambiguated words.

We approach this problem in a natural manner: since it is caused by the fact that different senses of a word are mixed together, we attempt to solve it by recovering the hidden structure in the corpus data. We formulate the problem of inducing the latent, meaning-level structure, as the problem of structure detection in dyadic data. More precisely we follow the aspect model summarized in Section 3.1 and we assume the existence of latent classes $Z = \{z_1, ..., z_K\}$, such that every occurrence of an observation pair (t_i, c_j) is associated to a latent class z_k. Under the aspect mode,l one can estimate the latent structure which maximizes the probability of the observed input data. More precisely, we estimate the distribution of the latent class given a target word: $P(z_k|t_i)$ and the distribution of the context feature variable given a latent class: $P(c_j|z_k)$.

Inducing the latent structure allows us to define new vector representations for the meaning of words. We propose to represent these in terms of the likelihood of each of the latent classes, leading to the following K-dimensional representations:

Basic representation for target t_i

$$\mathbf{v(t_i) = (P(z_1|t_i), ..., P(z_K|t_i))} \tag{3.1}$$

The assumption behind such a representation is that a target word can be described by a set of core classes and the frequency with which these are attested in a corpus. Note that the representation in (3.1) is not fixed but parametrized with respect to the input corpus (i.e. it only reflects word usage as attested in that corpus). Representation over classes $z_1 \dots z_K$ can also be seen as a means of reducing the dimensionality of the original co-occurrence matrix.

One main advantage of this formulation is that we can naturally define meaning representations for words occurring in context. Given an observation (t_i, c_j), a target word occurring with a context feature, we can compute the probabilities of the latent classes $P(z_k|t_i, c_j)$, leading to the following representation:

In-context representation for target t_i with context feature c_j

$$\mathbf{v(t_i, c_j) = (P(z_1|t_i, c_j), ..., P(z_K|t_i, c_j))} \tag{3.2}$$

where component $P(z_k|t_i, c_j)$ is the probability of class z_k given target word t_i and context feature c_j.

Here, target t_i is again represented as a distribution over latent classes, but is now modulated by a specific context c_j, which reflects actual word usage. As the empirical evaluation in Chapter 5 will show, this distribution is more "focused" compared to the one in (3.1): the context helps disambiguate the meaning of the target word, and as a result fewer classes will share most of the probability mass.

3.2.3 Computing similarity

In vector space models, vector similarity measures capture the intuition that words are similar in meaning if they have similar occurrence patterns. The vector representations we propose are proper distributions, unlike in most of the vector space models. For this reason, some of the typical similarity measures employed in vector spaces are particularly suited to our framework, and this section will briefly highlight these.

A number of similarity measures are used in vector spaces, out of which functions for measuring divergence between distributions are a natural choice for computing meaning similarity in the framework we have proposed.

Kullback-Leibler divergence, or relative entropy, is a popular divergence measure. Intuitively, $D(p||q)$ stands for the inefficiency of assuming that the distribution is q when the actual distribution is p.

$$\mathrm{D}_{\mathrm{KL}}(p|q) = \sum_i p_i log(\frac{p_i}{q_i})$$

Since this is an asymmetric measure, we use the Jensen-Shannon divergence which is the sum of the KL divergences of the two distributions from their average distribution.

$$\mathrm{D}_{\mathrm{JS}}(p,q) = \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(p|m) + \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(q|m) \tag{3.3}$$

where $m = \frac{1}{2}(p+q)$. The JS divergence has values in the $[0, \inf)$ interval, with identical distributions having 0 distance. We use the inverse of this distance as a similarity score: $\mathrm{sim}_{\mathrm{JS}}(v,w) = \frac{1}{D_{JS}(v,w)}$.

Scalar product, or dot product, is another measure for vector similarity. In our framework the scalar product stands for the probability that two words (isolated or in-context) have the same meaning, as a sum over all possible meanings:

$$sp(v,w) = \langle v,w \rangle = \sum_i v_i w_i$$

$$sp(v(t_1), v(t_2)) = \Sigma_k P(z_k|t_1)P(z_k|t_2) \tag{3.4}$$

$$sp(v(t_1,c_1), v(t_2,c_2)) = \Sigma_k P(z_k|t_1,c_1)P(z_k|t_2,c_2) \tag{3.5}$$

This function has an additional property when used to compare distributional meaning representations. The scalar product values range in the $(0,1]$ interval; however, the similarity between two identical vectors is 1 if and only if the vectors are unit vectors, i.e. $P(z_i) = 1$ for some i and $P(z_j) = 0, \forall j \neq i$. Such a vector can be interpreted as the meaning representation of a fully disambiguated word, a word for which there is no uncertainty about its meaning. The scalar product can therefore be said to reflect not only distribution similarity but also how *focused* or *un-ambiguous* the distributions are, as very ambiguous words are unlikely to be scored very similar. This is in contrast with the inverse Jensen-Shannon divergence in which maximally similar words are words that have identical distributions over classes, irrespective of their ambiguity level.

This leads to the observation that the framework we provide gives rise to a natural concept of "ambiguity" of a word, as measured by the entropy of its distributional representation. For an isolated word, this results in:

$$Amb(t_i) = -\Sigma_k P(z_k|t_i) log P(z_k|t_i)$$

Similarly, the entropy of a posterior distribution can be used as an indicator of the ambiguity level of a word given a context feature. This measure of ambiguity reflects how diverse the corpus usages of a word are, rather than the word's ambiguity in the sense of number of senses in a dictionary. One advantage is that such a measure is corpus and domain-dependent; however, as this value depends on the number of latent classes that we assume, it is not interpretable in itself but can only be used as a way to compare words in terms of their ambiguity level.

Similarity between words given a set of context features

We have considered so far the case of a target word occurring with *one* context feature. However, in most applications, we are given words occurring in larger context such as entire sentences. In this section we consider the case of a target word t occurring with a *set* of n features $\vec{c} = (c_1, ..., c_n)$ extracted from its context.

In the aspect model, each observation of a pair (t, c) is associated with a latent class z_k. This has made it straightforward to induce probabilities of latent classes conditioned on a (target,context) pair, $P(z_k|t, c)$, which give the components of our contextualized representations.

In the case of t occurring with a number of n context features we have n observation pairs $(t, c_1), (t, c_2), ..., (t, c_n)$. One can represent this tuple in terms of the latent classes associated to each of these pairs, leading to the computation of $P(\vec{z}|t, \vec{c})$, for each $\vec{z} = (z_1, z_2, ..., z_n) \in Z^n$. The interpretation of $P((z_1, z_2, ..., z_n)|t, (c_1, c_2, ..., c_n))$ is that the feature c_1 is associated to latent class z_1, c_2 with class z_2, etc. This is however a K^n-dimensional representation and computations on such representations are no longer feasible in practice.

For this reason we propose two alternative methods to compute similarity when given a set of context features.

The first proposal is to represent a word in context as a K-dimensional vector where each component stands for the probability that *all* context features have been generated with the same meaning z_k: $P((z_k, z_k, ..., z_k)|t, (c_1, c_2, ..., c_n))$

with $k : 1, ..., K$. The disadvantage to such a proposal is that the representation of a word in context is no longer a proper distribution, as it only selects K components out of the complete K^n-dimensional representation.

We propose a second method to integrate multiple context features into similarity computations. This does not specifically build a vector representation for a word and a set of context features as multiple context features are integrated only at the final stage of computing similarity.

This method can be applied to typical lexical substitution data. In the lexical substitution scenario, we are interested in comparing two target words, or phrases, t_1 and t_2 where t_1 occurs in a sentential context and t_2 is a candidate substitute for t_1 in the *identical*, given context. This is usually formulated as a ranking task, where the goal is to rank a list of candidate substitutes; it has been previously observed (Erk and Padó [2008], Thater et al. [2010]) that more discriminative power is obtained when only one of the words, t_1 or t_2, is contextualized. For this reason, we propose to compute the similarity between t_1 and t_2 given context $\vec{c} : \{c_1, ..., c_n\}$ as:

$$sim(t_1, t_2|\vec{c}) = sim(v(t_1, c_1), v(t_2)) * ... * sim(v(t_1, c_n), v(t_2))$$

where $\vec{c} : \{c_1, ..., c_n\}$ are the features extracted from the sentential context. Here $v(t_1, c_1)$ is the single-feature representation given in 3.2 and $v(t_2)$ is an uncontextualized vector representation as in 3.1. This way the context features are considered individually and each feature yields a similarity score. The similarity scores obtained from the entire context are multiplied to obtain a final score.

3.3 Relation to previous work

In the context of distributional methods for similarity in context, the method we have proposed can be categorized as type-based as it only uses a frequency co-occurrence matrix as input. However, unlike any of the previous type-based methods, it defined a probabilistic setting, in which the notions of word meaning and word meaning in context are in line with linguistic intuitions on these concepts. Conceptually our framework comes closer to the token-based method of Reisinger and Mooney [2010] and the methods for judging the context-appropriateness of DIRT paraphrases initiated by Pantel et al. [2007]. These also assume a latent structure which is typically discovered through some clustering step. Unlike this line of work, our approach is much more unitary: the aspect model framework allows us to formulate the problem as that of inducing the latent structure which maximizes the likelihood of the data. Following this, we build reduced vector representations which allow us to work within the vector space model paradigm. This is a natural, general approach which can be used to address the context-sensitive similarity problem in any of its instantiations.

In particular, we would like to point to the similarities between our framework and the theoretical proposal of Hanks [2000], which we briefly summarize here.

Following a long line of controversies on the appropriate nature of word senses and of the process of meaning disambiguation, Hanks [2000] proposes to describe word meanings in terms of *meaning components*. Meaning components reflect different aspects of the potential senses of a word and are to be triggered in a graded, probabilistic, fashion when given a word in context. Each of these components are in turn corpus-derived, determined by the word's usages. They are not necessarily mutually exclusive or mutually compatible, and an occurrence of a word triggers one or more of the potential meaning aspects. In Hanks [2010], the author further proposes these corpus-derived meanings to be expressed in terms of typical collocations and other prototypical context features. He argues that only combinations of these components can allow for the flexibility encountered in natural language.

To exemplify his proposal, in Table 3.1 we list the potential meaning components evoked by *bank* as suggested by Hanks [2000][1].

bank
• IS AN INSTITUTION
• IS A LARGE BUILDING
• FOR STORAGE
• FOR SAFEKEEPING
• OF FINANCE/MONEY
• CARRIES OUT TRANSACTIONS
• CONSISTS OF STAFF AND PEOPLE

Table 3.1: Meaning components invoked by usages of *bank* Hanks [2000]

For example a *blood bank* activates the components FOR STORAGE and FOR SAFEKEEPING. A larger context may also offer clues for other components such as IS AN INSTITUTION, IS A LARGE BUILDING or CONSISTS OF PEOPLE. On the other hand a *financial bank* will trigger OF FINANCE/MONEY component, CARRIES OUT TRANSACTIONS and others, as indicated by the entire context. To conclude, Hanks proposes a word's meaning to be represented as "a unique combination of the components that make up its meaning potential, activated by contextual triggers". This activation should be of probabilistic nature, as indicated by occurrence patterns observed in large corpora.

The framework we have developed is very much in line with the theoretical proposals of Hanks, as we represent the meaning of each word as a distribution over corpus-derived latent classes, while a latent class is defined as a distribution over context features. In the proposal of Hanks [2000], disambiguation can be seen as the process of activating meaning components, as triggered by context features. In our concrete framework, disambiguation is a shift in the distribution of meaning components, indicative of *a posteriori* meaning. The

[1]Here, the author leaves out the homonym *bank* as in *river bank*.

intuition behind is that when given an isolated word, we represent it in terms of the most likely meaning components that we associate it with, given the input corpus. Given an actual context clue, these beliefs change, as one or more meaning aspects become more likely.

In the following chapter, we continue the presentation of our framework with the description of two latent variable models which can be used to induce latent classes.

Chapter 4

Models for Latent Variable Factorization

This chapter describes the latent variable models employed for building contextualized meaning representations.

What is common to all latent variable models is the assumption that some data can be explained by a latent, hidden structure, and the goal is to uncover this structure using only this raw observation data. This is a very natural assumption, as such a hidden structure is conceptually motivated in many cases. For example in consumption behavior, underlying preferences can be thought of as explaining the observation data which may consist only of consumers' shopping records. In NLP, the hidden structure corresponds to an abstract representation level which we want to induce without the use of annotated data. A number of hidden models have a long tradition in NLP, such as Hidden Markov Models for POS tagging or unsupervised PCFG (Probabilistic Context Free Grammar) induction for parsing, to name just a few.

The methods vary with respect to the particular assumptions on the underlying structure. In this thesis, we use latent variable models developed in the context of topic modeling, which are particularly suited for modeling word co-occurrence data and the underlying senses of words.

In topic modeling the goal is to find the topics which span a corpus of documents. A document is associated to a distribution over topics, and in turn each topic is defined as a distribution over words. The following example (from Blei et al. [2003]) shows the typical output of a topic model. A set of topics, in this case three topics, are associated with distributions over words (top 5 words listed here), and each word in a document can be assigned to such a topic.

The most straightforward application of topic modeling is the computation of document similarity. Two documents can be compared based on their distri-

Arts	Budgets	Education
NEW	MILLION	EDUCATION
FILM	TAX	STUDENTS
SHOW	PROGRAM	SCHOOLS
MUSIC	BUDGET	EDUCATION
MOVIE	BILLION	TEACHERS

The Metropolitan Opera Co. and New York Philharmonic will receive $400000 each. The Juilliard School where music and the performing arts are taught will get $250000.

Table 4.1: Latent topics in a collection of news text

butions over topics, representation which abstracts away from the actual words contained by the documents and leads to more accurate similarity assessments. Topic models have also been used in a variety of applications such as image analysis (Li et al. [2005] Russell et al. [2006] Blei and Jordan [2003]), population genetics (Pritchard et al. [2000]) or for unsupervised part of speech tagging (Toutanova and Johnson [2008]). In Sections 4.1 and 4.2 we detail on Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA). PLSA is the terminology used for the aspect model briefly introduced in Section 3.1 while LDA is an extension of this.

In Section 4.1.2 we describe Non-negative Matrix Factorization (NMF). Matrix factorization methods can also be thought of as uncovering a hidden structure in the observation data. In this case, the observation data is a matrix, representing the co-occurrence of two abstract sets of variables. Latent Semantic Analysis (LSA) is such a factorization method which uses Singular Value Decomposition (SVD) to identify the directions of maximum variation and project the data on these directions. Intuitively, the goal of LSA is to identify the main concepts underlying the data and to represent the words in terms of these concepts.

Unlike other matrix decomposition methods, NMF imposes the constraint of non-negativity. This translates into assuming that the original data is an *additive* mixture over a set of latent components: a number of latent components combine to explain the observation data, however only in an additive fashion. This is a very important property as in most data there is no support for the idea of a concept occurring in a negative quantity[1].

NMF has been originally introduced in the context of computer vision. In image analysis applications, the features, or latent components discovered this way can be visualized; Lee and Seung [1999] show that when used to analyze a collection of images of human faces, these components correlate very well with natural parts of human faces, such as mouth or eyes. This way, the original, noisy vector representation of a face image is transformed into a reduced representation over these components. This natural decomposition into parts of faces is not

[1]A particular case of this problem has been shown to be equivalent to PLSA with maximum likelihood estimation (Ding et al. [2008]). However, the research on these two has been conducted mostly independently and a number of algorithms have been proposed in the literature, specifically for solving the NMF problem. In our experiments we use one of these algorithms motivating thus our presentation of NMF in this chapter.

obtained through any of the other dimensionality reduction methods tested in this study, such as Principal Component Analysis or Vector Quantization.

NMF has been further used for computer vision (Li et al. [2001]), microarray data analysis (Kim and Park [2007]), biomedical text mining (Chagoyen et al. [2006]) or gene expression data analysis (Brunet et al. [2004], Gao and Church [2005]).

4.1 Probabilistic Latent Semantic Analysis

This section describes the aspect model introduced by Hofmann and Puzicha [1989] and Hofmann [1999] in 4.1.1. In section 4.1.2 we introduce the Non-negative Matrix Factorization problem, a special case of which has been shown to be equivalent to the PLSA method; section 4.1.3 details on the relation between the two. This section is concluded with an NMF example.

4.1.1 Overview

This section details on the aspect model briefly introduced in Section 3.1. In the context of topic modeling, the terminology used for the aspect model is that of Probabilistic Latent Semantic Analysis (PLSA). In the PLSA model, the observation data $\mathcal{S}$ is formed of occurrences of words in documents, i.e. pairs in the domain $\mathcal{D} \times \mathcal{W}$, where $\mathcal{D} = \{d_1, ..., d_I\}$ is a set of document labels and $\mathcal{W} = \{w_1, ..., w_J\}$ is a set of words. Alternatively, the observation data can be thought of as a collection of documents, where each document d comprises of all its word occurrences. We will denote a document by the words occurring in it: $d = \{w_{d1}, ..., w_{dN}\}$. We use $dn \in \{d1, ..., dN\}$ as a word position index in document d[2].

PLSA associates to each observation pair a latent class from a finite discrete variable $\mathcal{Z} = \{z_1, ..., z_K\}$. The observation data is assumed to be generated according to the following random sampling proccess:

1. Select a document with probability $P(d)$
2. For each word position dn in document d:
   (a) Generate a latent topic z with probability $P(z_{dn}|d)$
   (b) Generate a word w with probability $P(w_{dn}|z_{dn})$

The aspect model makes two independence assumptions which are necessary in order to reduce the complexity of the model: 1) the observation pairs (d, w), a

[2]To simplify the presentation, subscripts are omitted when their presence does not carry particular relevance.

word occurring in a document, are generated independently, and 2) words and documents are conditionally independent given a latent topic. This results in $P(w|z,d) = P(w|z)$, which allows the sampling of a word w to be conditioned only on the latent class z.

Under the first independence assumption, the probability of the observed data is:

$$P(\mathcal{S}) = \prod_{d} \prod_{dn=d1}^{dN} P(d, w_{dn}) = \prod_{d} \prod_{dn=d1}^{dN} \sum_{k=1}^{K} P(d, w_{dn}, z_k)$$

where d ranges over the documents in the collection, and dn over each position in these documents.

The goal of structure detection is to induce the latent structure which maximizes $P(\mathcal{S})$, the likelihood of the observation data. Two parameterizations are possible under the second independence assumption, as the probability of a distinct word-document dyad (d, w) can be alternatively factorized as in 4.1 or 4.2:

$$P(d,w) = \sum_{z} P(d,w,z) = P(d) \sum_{z} P(z|d)P(w|z) \tag{4.1}$$

$$P(d,w) = \sum_{z} P(d,w,z) = \sum_{z} P(z)P(d|z)P(w|z) \tag{4.2}$$

4.1 corresponds to the generative process earlier defined, and it is known under the name of *asymmetric* parametrization. Equivalently the *symmetric* parametrization in 4.2 is also possible. In the asymmetric parametrization the latent structure to be detected consists of the probability of the latent topics given each document, $P(z|d)$, and $P(w|z)$, the probability of each word given a topic. Depending on the particular application at interest, the symmetric parametrization may be preferable; this parametrization defines both documents and words as probability distributions conditioned on latent topics.

The parameters ($P(d)$, $P(z|d)$ and $P(w|z)$ for the asymmetric parametrization or $P(z)$, $P(d|z)$ and $P(w|z)$ for the symmetric one) can be determined through maximum likelihood estimation, i.e. by maximizing the log-likelihood of the observed data $logP(\mathcal{S})$. For solving this problem, Hofmann [1999] proposes a few variants of the EM algorithm.

4.1.2 Non-negative matrix factorization

Non-negative matrix factorization (NMF) is a matrix approximation method which has become popular with the work of Lee and Seung in Lee and Seung [1999] and Lee and Seung [2000]. NMF has also been proposed as a method to obtain representations over a reduced space of features which reflect the hidden structure in the input data. NMF decomposes a non-negative matrix as a mixture over a set of basis elements which combine in an *additive* fashion. The intuition that these elements will correspond to *meaningful* latent components, as opposed to other decompositions which do not enforce non-negativity, is supported by the experimental results obtained by Lee and Seung.

Formally, given a $(I \times J)$ non-negative matrix V, the goal is to find non-negative matrices W and H of sizes $I \times K$ and $K \times J$ such that WH best approximates V under a given cost function.

$$V \approx WH \tag{4.3}$$

A non-negative matrix is a matrix whose elements are larger or greater than 0. Lee and Seung [2000] give algorithms for approximations under two cost functions: Euclidean distance and a second cost function inspired from the Kullback-Leibler (KL) divergence. This is the cost function of interest for our work; it is defined as follows:

$$J_{NMF-KL}(V||WH) = \sum_{i,j} V_{ij} log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}$$

The problem is to find factors W and H which minimize $J_{NMF-KL}(V||WH)$ subject to the constraints $W, H \geq 0$.

4.1.3 Equivalence between NMF and PLSA

As previously discussed, the PLSA problem is that of finding the factorized distributions that maximize the likelihood of the observation data, under one of the following equivalent parametrizations:

$$P(d,w) = \sum_z P(d,z)P(w|z) = \sum_z P(d)P(z|d)P(w|z)(asymmetric) \tag{4.4}$$

$$P(d,w) = \sum_z P(d,z)P(w|z) = \sum_z P(z)P(d|z)P(w|z)(symmetric) \tag{4.5}$$

This problem has been shown to be equivalent to the NMF with the J_{NMF-KL} cost function problem in Gaussier and Goutte [2005] and Ding et al. [2008]. In particular, we are interested in a solution to the PLSA problem (more precisely in the asymmetric parametrization), and for the rest of this section we show how to obtain this from a NMF factorization.

Let V be a document-term input frequency matrix (therefore non-negative) and $T = \sum_{i,j} V_{ij}$ the sum of its elements. We can normalize V such that the sum of its elements is 1 ($\sum_{i,j} V_{ij} = 1$) by $V_{ij} := \frac{V_{ij}}{T}$. V can now be interpreted as the (empirical) distribution $\hat{P}(d,w)$.

We define the following operations on the two factors W and H. Let B be a $(K \times K)$ diagonal matrix with $B_{kk} = \sum_j H_{kj}$. It can be easily shown that the rows of matrix $B^{-1}H$ sum to 1 ($\sum_j (B^{-1}H)_{kj} = 1$). The sum of all elements of WB is also 1 ($\sum_{i,k}(WB)_{ik} = 1$) because $\sum_{i,j} WH_{ij} = 1$.

Matrices WB and $B^{-1}H$ are also a solution to the same NMF problem because $WH = (WB)(B^{-1}H)$. Furthermore, the factors (WB) and $(B^{-1}H)$ can now be interpreted as: $P(d,z)(WB)$ and $P(w|z)(B^{-1}H)$. The equivalence result in Ding et al. [2008] yields that $P(d,z)$ and $P(w|z)$ distributions obtained this way give a solution to the maximum likelihood PLSA problem. This factorization is depicted in Figure 4.1.

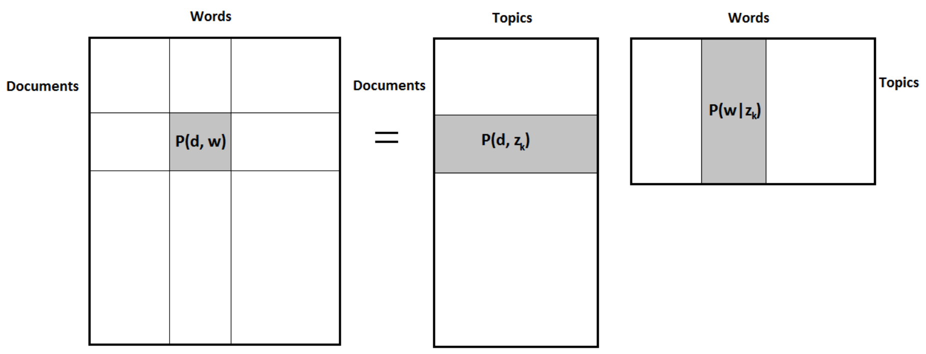


Figure 4.1: PLSA as matrix factorization: $P(d,w) = \sum_k P(d,z_k)P(w|z_k)$

Similarly, the symmetric and asymmetric parametrizations of Eq. 4.4 and 4.5 can be obtained from a NMF solution.

Symmetric parametrization Let B be a $(K \times K)$ diagonal matrix with $B_{kk} = \sum_j H_{kj}$ and A a $(K \times K)$ diagonal matrix such that $A_{kk} = \sum_i W_{ki}$. We obtain:

$$WH = (WA)(A^{-1}B)(B^{-1}H)$$

$$\begin{array}{llll} WA & (I \times K) & P(d|z) & \sum_i P(d_i|z) = 1 \\ A^{-1}B & (K \times K) & P(z) & \sum_k P(z_k) = 1 \\ B^{-1}H & (K \times J) & P(w|z) & \sum_j P(w_j|z) = 1 \end{array}$$

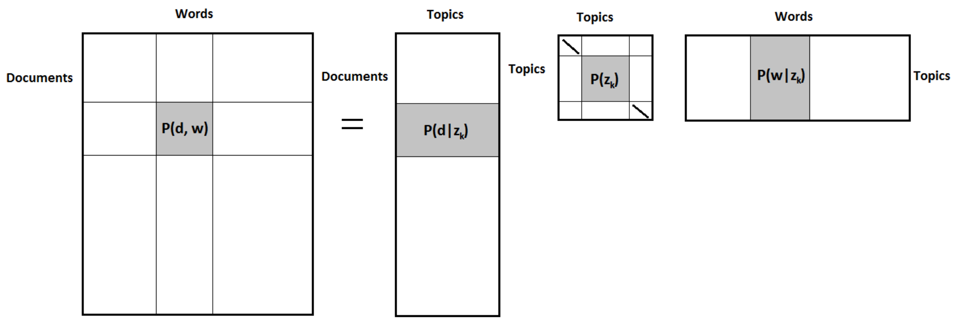


Figure 4.2: Symmetric PLSA parametrization as matrix factorization: $P(d,w) = \sum_k P(d|z_k)P(z_k)P(w|z_k)$

Asymmetric parametrization Let B be a diagonal matrix with $B_{kk} = \sum_j H_{kj}$ and A a diagonal matrix with $A_{ii} = \sum_k (WB)_{ik}$.

$$WH = (A)(A^{-1}WB)(B^{-1}H)$$

$$\begin{array}{llll} A & (I \times I) & P(d) & \sum_i P(d_i) = 1 \\ A^{-1}WB & (I \times K) & P(z|d) & \sum_k P(z_k|d) = 1 \\ B^{-1}H & (K \times J) & P(w|z) & \sum_j P(w_j|z) = 1 \end{array}$$

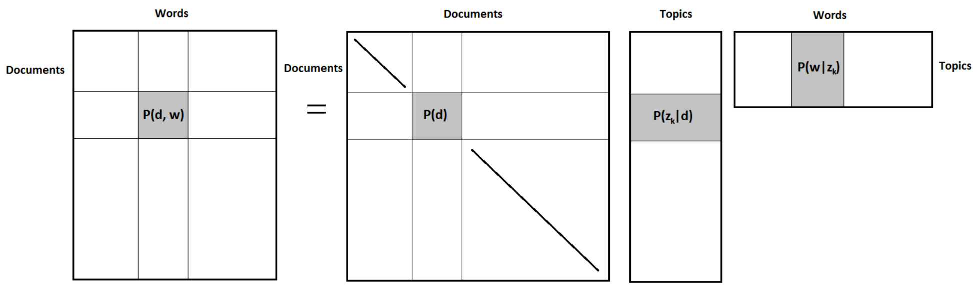


Figure 4.3: Asymmetric PLSA parametrization as matrix factorization: $P(d,w) = \sum_k P(d)P(z_k|d)P(w|z_k)$

Example In this section we exemplify the use of NMF for modeling word occurrences and their underlying senses on a toy input frequency matrix of dimension 5×4, given in Table 4.2.

The co-occurrence matrix reflects two meanings: one shared by target words *spread, transmit* and signaled by co-occurring words *virus* and *disease*; a second meaning is shared by *reveal, show* in the context of *test* and *study.* The word *shed* is ambiguous between the two meanings, co-occurring with context features indicative of both of them.

	virus	disease	test	study
spread	4	8	0	0
transmit	6	8	0	0
show	0	0	3	4
reveal	0	0	2	3
shed	3	1	2	4

Table 4.2: Input frequency matrix

The input to the NMF problem is the empirical distribution of the observed data $P(t,c)$, which is obtained as the normalized input matrix given in Table 4.3.

We set $K = 2$; the distributions estimated after 1000 iterations of the multiplicative rules algorithm of Lee and Seung [2000] are given in Tables 4.4, 4.5 and 4.6.

	virus	disease	test	study
spread	0.08	0.16	0	0
transmit	0.12	0.16	0	0
show	0	0	0.06	0.08
reveal	0	0	0.04	0.06
shed	0.06	0.02	0.04	0.08

Table 4.3: Normalized input matrix V: $V_{ij} = P(t_i, c_j)$

As expected, words *spread* and *transmit* are fully associated to a class $Z1$ ($p(Z1|spread) = 1.0$ and $p(Z1|transmit) = 1.0$) while *show* and *reveal* are fully represented by class $Z2$. Without the presence of context both meanings are possible for word *shed* with $P(Z1|shed) = 0.4$ and $P(Z2|shed) = .6$.

As indicated by the $P(c|z)$ matrix, class $Z1$ is described by context words *virus* and *disease* while $Z2$ is defined by context words *test* and *study*.

	spread	trans.	show	reveal	shed
spread	0.25	0	0	0	0
trans.	0	0.29	0	0	0
show	0	0	0.14	0	0
reveal	0	0	0	0.10	0
shed	0	0	0	0	0.20

Table 4.4: $P(t)$

	Z1	Z2
spread	1.00	0.00
transmit	1.00	0.00
show	0.00	1.00
reveal	0.00	1.00
shed	0.40	0.60

Table 4.5: $P(z|t)$

	virus	disease	test	study
Z1	0.43	0.57	0.00	0.00
Z2	0.00	0.00	0.39	0.61

Table 4.6: $P(c|z)$

Alternatively, this decomposition is equivalent to obtaining the original observation data as a sum over two classes, or "aspects": $P(t, c) \approx \sum_z P(t, c, z) \approx P(t, c, Z1) + P(t, c, Z2)$. These are given in Tables 4.7 and 4.8.

The distributions in Tables 4.5 and 4.6 can further be used to compute representations for the ambiguous word *shed* which are given in Table 4.9.

Without context the word *shed* is represented by classes $Z1$ and $Z2$ in a $0.4 : 0.6$ ratio. However, given a context word such as *virus* we obtain a fully disambiguated representation with $P(Z1|shed, virus) = 1.0$ and $P(Z2|shed, virus) = 0.0$. This way, *shed* is maximally similar to *transmit* and *spread* when encountered in the context of *virus* and to *show* and *reveal* when encountered with

	virus	disease	test	study
spread	0.10	0.14	0	0
transmit	0.12	0.16	0	0
show	0	0	0	0
reveal	0	0	0	0
shed	0.03	0.04	0	0

Table 4.7: $P(t, c, Z1)$

	virus	disease	test	study
spread	0	0	0	0
transmit.	0	0	0	0
show	0	0	0.05	0.09
reveal	0	0	0.04	0.06
shed	0	0	0.05	0.07

Table 4.8: $P(t, c, Z2)$

(target)	$P(Z1\|t)$	$P(Z2\|t)$
shed	0.4	0.6
(target, context)	$P(Z1\|t,c)$	$P(Z2\|t,c)$
(shed, virus)	1	0
(shed, study)	0	1

Table 4.9: Meaning representations

context *study*.

4.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) was introduced in Blei et al. [2003] as an extension of the PLSA model, and was originally proposed in the same context of modeling collections of documents and their underlying topics.

The main difference to the PLSA aspect model is that LDA makes assumptions about the mixtures of topics in a document, as these are assumed to be drawn from a Dirichlet distribution. The LDA variant we use was introduced by Griffiths and Steyvers [2004]. This variant uses Dirichlet distributions both for generating topic mixtures as well as for generating the mixture of words associated to a topic. LDA has been proposed as a way to overcome the overfitting issues that PLSA has been criticized for (Blei et al. [2003]).

More precisely, the generative model for each document d in the corpus is the following:

1. Generate a distribution over topics $\theta^d : Dirichlet(\alpha)$
2. For each word position dn in document d:
   (a) Generate a latent topic z_{dn} with probability $P(z_{dn}|\theta^d)$
   (b) Generate a distribution over words $\phi^{z_{dn}} : Dirichlet(\beta)$
   (c) Generate a word w_{dn} with probability $P(w_{dn}|\phi^{z_{dn}})$

In more detail, for each document, we first draw the mixing proportion over topics θ_d from a K-dimensional Dirichlet random variable with parameters α. The Dirichlet distribution (see Appendix A for a detailed definition) is a K-dimensional distribution with parameters $\alpha = (\alpha_1, ...\alpha_K)$ with $\alpha_1, ..., \alpha_K > 0$. Its domain is the $(K-1)$ simplex, i.e. it is defined for all $x_1, ...x_{K-1} > 0$,with $x_1 + ... + x_{K-1} \leq 1$ and $x_K = 1 - x_1 - ... - x_{K-1}$ and it has the following probability density function:

$$f(x_1, ..., x_K; \alpha_1, ..., \alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{4.6}$$

where the beta function $B(\alpha)$ is a normalizing constant. Since a Dirichlet random variable is defined such that $\sum_{i=1}^{K} X_K = 1$ it can be seen as a distribution over distributions. In the LDA case, $\theta^d : Dir(\alpha)$ models the distribution over topic mixtures, indicating how probable each topic mixture is, given document d.

Next, for each of the word positions in document d, a topic z is first drawn from $P(z|\theta_d)$. Given each topic z, we sample ϕ^z a J-dimensional Dirichlet variable with parameter β. Similarly, this will give mixture proportions, this time over words w, given a topic z. Finally a word w is sampled from $P(w|\phi^z)$.

The joint probability of a document collection $\mathcal{S}$ and of the hidden variables is:

$$P(\mathcal{S}, \boldsymbol{\theta}, \boldsymbol{\phi}|\alpha, \beta) = \prod_{d} P(\theta^d|\alpha) \prod_{dn=d1}^{dN} P(z_{dn}|\theta^d) P(\phi^{z_{dn}}|\beta) P(w_{dn}|\phi^{z_{dn}}), \tag{4.7}$$

The central computational problem in LDA is to obtain the posterior distribution $P(\theta, \phi, \mathbf{z}|\mathbf{w}, \alpha, \beta)$ of the hidden variables $\mathbf{z} = (z_{d1}, z_{d2}, \ldots, z_{dN})$ given a document $\mathbf{d} = (w_{d1}, w_{d2}, \ldots, w_{dN})$. Although this distribution is intractable in general, a variety of approximate inference algorithms have been proposed in the literature, such as a variational Bayes approximation in Blei et al. [2003] or Gibbs sampling, a Monte Carlo Markov Chain method, in Griffiths and Steyvers [2004].

Effect of hyperparameter choice The parameters of the Dirichlet distributions α and β, also called hyperparameters, are predefined, corpus-level parameters.

More precisely, the α parameters of a Dirichlet distribution control 1) the mean of the marginals X_i i.e. the expected probability of each component and 2) the variance of the marginals.

In the LDA case, a symmetric α, i.e. $\alpha_1 = \alpha_2 ... = \alpha_K$ for the θ Dirichlet random variable, means that, over the entire corpus, all topics are equally likely. Given a symmetric α, a large α value determines low variance, meaning that most of the mixtures will be close to the mean, a mixture in which all topics are equally likely. On the other hand a small α value will turn skewed, sparse mixtures more likely; in a sparse mixture only a small number of topics share most of the probability mass while the rest of the topics have very low probability.

The effect of the parameter β of the ϕ Dirichlet random variable is similar. A small value will indicate that only a small number of words are associated to a latent topic and a symmetric β prior indicates that over the entire corpus, all words are equally likely.

In topic modeling it is common to use symmetric α and β priors. These are usually chosen to have small values to indicate the intuition that the topic distribution of a document and the word distribution of a topic should be sparse distributions.

4.3 Latent variable factorization for inducing latent meaning classes

The observation data that is the focus of this thesis consists of occurrences of target words with context features $(t_i, c_j) \in \mathcal{T} \times \mathcal{C}$. In the topic modeling terminology, a document would correspond to a target and the words in this document are its co-occurring context features.

We use the PLSA/NMF and LDA models to induce a latent structure over a set of K classes $\mathcal{Z} = \{z_1, ..., z_K\}$. Although, as discussed, these methods differ in the assumptions they make on the generative model of the data, they are both motivated models for structure detection in dyadic data.

More precisely, the distributions we induce from the input frequency matrix are the posterior of the latent class variable given a target word: $P(z_k|t_i)$ and the distribution over context features given a latent class $P(c_j|z_k)$.

The distributions over latent classes for each target word $P(z_k|t_i)$ give the isolated word representations as described in Chapter 3. For the contextualized representations we proposed the use of $P(z_k|t_i, c_j)$, which is the probability of a latent class z_k given the observation of a word and a context feature. This is obtained as:

$$P(z_k|t_i, c_j) = \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)}$$

Again, the independence assumption made here is that context features and target words are conditionally independent given a latent class, leading to: $P(c_j|z_k, t_i) = P(c_j|z_k)$. Although not true in general, the assumption is relatively weak. We do not assume that words and context features occur independently of each other, but only that they are generated independently *given an assigned meaning.*

For the PLSA estimation we use the multiplicative rules algorithm proposed to solve the NMF problem[3]. The distributions at interest correspond to the *asymmetric* parametrization previously described. As discussed, NMF with KL-divergence cost function is an equivalent problem and the previous section has shown how to obtain the factorized distributions at interest from a NMF solution. More precisely, the input matrix is the normalized frequency matrix V. Given W and H, a solution to the NMF problem, we follow the operations earlier described for obtaining the asymmetric parametrization and we obtain meaning representations using:

$$P(z_k|t_i) = (A^{-1}WB)_{ik}; \tag{4.8}$$

$$P(z_k|t_i, c_j) = \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)} = \frac{(A^{-1}WB)_{ik}(B^{-1}H)_{kj}}{\sum_k (A^{-1}WB)_{ik}(B^{-1}H)_{kj}} \tag{4.9}$$

For the estimation of the LDA model we use an implementation of the collapsed Gibbs sampling algorithm as described in Heinrich [2008]. This is a Monte Carlo Markov Chain (MCMC) method. MCMC are methods for simulating complex distributions by generating a set of samples which is proven to converge to the desired distribution. In our case, the method produces a chain of samples of topic assignments $\mathbf{z} : z_1, ..., z_n$ for each context feature occurring with a word $\mathbf{c} : c_1, ..., c_n$, which converges to the posterior probability of topics, given features: $P(\mathbf{z}|\mathbf{c})$. After a sufficient number of iterations, the chain converges to the posterior and θ and ϕ can be estimated from samples of this chain. These are further used for building meaning representations:

$$P(z_k|t_i) = \theta_i^k \tag{4.10}$$

$$P(z_k|t_i, c_j) = \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)} = \frac{\theta_i^k \phi_j^k}{\sum_k \theta_i^k \phi_j^k} \tag{4.11}$$

[3]Despite the findings reported in Ding et al. [2008], we have in fact failed to identify any differences between the NMF multiplicative rules algorithm and the PLSA EM algorithm proposed by Hofmann.

In particular, the LDA model requires the specification of the hyperparameters. For this, we use Dirichlet parameters that have been shown to perform well in topic modeling, as the intuition behind this choice is similar. A small α value indicates that only several meanings components are characteristic of a word and small β that a meaning component is signaled by a relatively small number of prototypical contextual features.

4.4 Previous applications to modeling semantics

In this section we summarize the previous work on using topic models and NMF for modeling semantics. As shown throughout this chapter, these models can provide a very straightforward and motivated basis for computing context-aware meaning similarity within the vector space paradigm. Despite the clear advantages that these probabilistic models offer, they have not been yet used for approaching this issue.

Despite their general formulation, their use for modeling semantics has been mostly restricted to the analysis of the document-word data specific to the information retrieval scenario. The models we have described in this section can be used as dimensionality reduction for vector space models, however the most widely-used method is still Latent Semantic Analysis introduced by Landauer and Dumais [1997]. For the rest of this chapter we will briefly introduce Latent Semantic Analysis in order to point out the major differences between this method and topics models. Following this, we overview some of the previous work on using topic models and NMF as dimensionality reduction methods in modeling semantics.

Latent Semantic Analysis (LSA) (Landauer and Dumais [1997] Deerwester et al. [1990]) is one of the earliest and most popular dimensionality reduction methods. The goal is to map frequency matrices to lower dimensional spaces, in which the dimensions hopefully reflect meaningful concepts, and thus rendering similarity measures on these vectors more accurate.

Given a co-occurrence matrix X of size $m \times n$, LSA computes the singular value decomposition (SVD): $U\Sigma V$. Matrices $U : m \times m$ and $V : m \times n$ are orthogonal (the left and right singular vectors) and $\sum : m \times m$ is a diagonal matrix of singular values. By deleting $m - k$ rows from the SVD factorization, we obtain $X \approx \hat{U}\hat{\Sigma}\hat{V}$, with $\hat{U} : m \times k$, $\hat{\Sigma} : k \times k$, $\hat{V} : k \times n$. $\hat{U}\hat{\Sigma}\hat{V}$ is the best rank k approximation of X under Frobenius norm.

Matrix $\hat{U}$ maps terms to a reduced space and similarity between terms i and j is computed as the similarity between the two row vectors in $\hat{U}$. Two documents can be compared using their corresponding column vectors in $\hat{V}$. Over the years, it has been shown that this method can closely match human similarity judgments and that it can be used in various applications such as information retrieval, document classification, essay grading etc[4].

One of the main criticisms to the reduced-representations obtained in LSA stems from the difficulty to interpret them. Unlike in the case of topic models, these do no have probabilistic interpretations. Although empirically proven useful, as similarity on these representations becomes more accurate, it is not possible to see the underlying concepts that are discovered. Further on, it has also been argued that, from a theoretic perspective, in the way it is currently employed, LSA is not a sound method to discover such underlying latent structure (Bast and Majumdar [2005]).

For the same reasons, the LSA method is not a candidate as a method for latent structure detection in our framework. Just to exemplify this, consider the following SVD factorization obtained on the input data in our previous toy example.

spread	-0.64	0.12
transmit	-0.73	0.10
show	-0.03	-0.64
reveal	-0.02	-0.46
shed	-0.21	-0.58

Table 4.10: $\hat{U}$

13.62	0.00
0.00	7.61

Table 4.11: $\hat{\sum}$

-0.55	-0.08	0.80	0.19
-0.82	0.16	-0.52	-0.12

Table 4.12: $\hat{V}$

Table 4.13: SVD decomposition of the input matrix in 4.2; k = 2

As it can be observed, the 2-dimensional representation given by matrix $\hat{U}$ reflects a similar structure to the one PLSA models find. However, without an underlying probabilistic model it is impossible to obtain the contextualized meaning representations we propose in our framework.

Topic models have been previously used for semantic tasks, however mostly on document-word co-occurrence data. Work such as Cai et al. [2007] or Boyd-Graber et al. [2007] uses the document-level topics extracted with LDA as indicators of meanings for word sense disambiguation. These are integrated as features in a supervised learning scenario.

More related to our work is that of Brody and Lapata [2009] who uses LDA-based models which induce latent variables from task-specific data rather than from simple documents. Brody and Lapata [2009] apply such a model for word sense induction on a set of 35 target nouns. They assume senses as latent variables and context features as observations; unlike our model they induce local senses specific to every target word by estimating separate models with the final goal of explicitly inducing word senses. Although motivated for the WSD task, such word-specific representations are not straightforward to use for computing similarity.

[4]There are number of interpretations on why this method is empirically successful. An account of these can be found in Turney and Pantel [2010].

NMF has been used as a dimensionality reduction method in Van de Cruys [2008] and Novakovitch et al. [2009].

In Van de Cruys [2008], the authors propose an extension of this to handle three-way data for word sense discrimination. After mapping words to reduced representations, these are further on clustered based on these representations. The authors do not investigate what exactly is the effect of this dimensionality reduction compared to other methods. They also do not motivate the choice for a second clustering step on these representations, as an alternative to the soft clustering that the NMF algorithm implicitly performs.

The study in Novakovitch et al. [2009] performs a set of experiments to show the superiority of NMF as a dimensionality reduction over LSA. This is a small controlled study in which the authors compare NMF with LSA on inducing "shades of meaning". More precisely a small number of nouns are the target words and input co-occurrence matrices are built for each of these nouns. Each of these matrices makes use of corpus sentences which contain the target word. The input representation chosen in these experiments is not that of standard vector space models, as each word is represented by a matrix, rather than a vector. Following this, a NMF algorithm and SVD are run in order to obtain a K-dimensional reduced representations for each of the matrices associated to a word. Values for K are chosen to correlate with the number of meanings that the words have (as judged by humans or as appearing in WordNet). The authors show that the meanings obtained by NMF method correlate better with human judgments than the ones obtained through LSA.

Chapter 5

Word Similarity

We experimentally evaluate the framework proposed on three tasks, in chapters 5, 6 and 7.

Word similarity (current chapter) In the word similarity task, the goal is to accurately asses the similarity in meaning of two words. The gold standard which word similarity methods are typically tested against consists of human assessments of word similarity. For example (WordSim-353 dataset), words such as *midday-noon* are judged as highly similar while *professor-cucumber* are considered to be unrelated words by humans. The ability to recognize words that are meaning-related is crucial to all NLP applications that require at least some level of language understanding, turning thus word similarity into one of the most fundamental semantic tasks. The current chapter addresses this task within the context of our framework, by employing the basic representations proposed in Chapter 3 for comparing words occurring in isolation. In particular the focus of this chapter is to determine the best parameters for the probabilistic methods we have proposed as well as for a set of baseline methods we compare against.

Lexical substitution (Chapter 6) The lexical substitution task addresses the question of word meaning similarity for words occurring *in context*. Lexical substitution has been originally proposed as a robust alternative to word sense disambiguation: the goal is to find substitutes of words occurring in sentential context, rather than to assign to a word a single sense chosen from a particular sense inventory. Although in general the lexical substitution task is more challenging than the (isolated) word similarity one, it addresses the important problem of assessing similarity of words *in context*, as they naturally occur. Computation of meaning similarity in context for the task of lexical substitution is the main evaluation of the framework we have proposed and is carried out in Chapter 6.

Contextual appropriateness of distributional paraphrases (Chapter 7) The role of contextual information in assessing similarity in meaning has also been highlighted in the development of distributional paraphrasing methods[1]. The key observation made in the context of this work is, again, the fact that certain phrases convey the same meaning only when occurring in particular contexts; this has lead to the conclusion that more accurate paraphrase resources can only be obtained by analyzing context, and possibly enhancing paraphrases with such contextual information. Chapter 7 evaluates our framework on assessing context-sensitive *phrase*-level similarities based on the paraphrasing vector space model introduced by Lin and Pantel [2001b].

5.1 Experimental setup

In the word similarity task, the goal is to assign scores to pairs of words reflecting their degree of similarity in meaning. A model for word similarity can be evaluated in an application scenario, such as information retrieval or question answering, or against human similarity judgments. Throughout this chapter, we use the test collection of Finkelstein et al. [2002] (WordSim-353), containing human-assigned similarity scores.

Data The WordSim-353 data set contains 353 pairs of words and their similarity scores as annotated by human subjects. The annotation is not specific about the type of similarity or relatedness in question, as the only instructions provided to the annotators ask for "a numerical similarity score between 0 and 10 (0 = words are totally unrelated, 10 = words are very closely related)"

A subset of the data is annotated by 13 participants while the remainder is annotated by 16 subjects. We use the entire data set together with mean assigned similarity scores. Table 5.1 gives an example of this data.

Both highly similar words, such as *midday-noon*, are assigned high scores as well as very related, however not-synonymous words, such as *Maradona-football.* Despite the fact that guidelines require antonyms to be judged as similar if they "belong to the same domain or represent features of the same concept", antonymous pairs such as *smart-stupid* are judged, on average, as being mildly related. Despite the rather underspecified guidelines, the data set exhibits substantial agreement, as we have obtained a mean rank correlation between different annotators of over 0.8 Spearman Rho on this data set.

[1]We use the term "distributional paraphrasing" to stand for methods which derive paraphrases solely based on occurrence statistics, such as the DIRT algorithm (Lin and Pantel [2001b]), which is based on an underlying syntactic vector space model.

Word pair		Avg. relatedness
midday	noon	9.29
magician	wizard	9.02
Maradona	football	8.62
tennis	racket	7.56
smart	stupid	5.81
professor	cucumber	0.31

Table 5.1: Sample of the WordSim-353 data set

Evaluation metrics We test a number of methods on assigning similarity scores to each pair of words in the data set. In order to compare these scores with the scores assigned by human participants, both the gold and the system scores are transformed into rankings. The gold annotation results in a tied rank, as a number of pairs are assigned the same score. We use Spearman Rho adjusted for tied rankings as a metric for the correlation between the rankings returned by different methods and the gold ranking.

Given samples of two random variables X and Y corresponding to two rankings, this is defined as:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

where μ_X and μ_Y are estimated with the empirical means $\bar{X}$ and $\bar{Y}$ and σ_X is the standard deviation:

$$\rho_{X,Y} = \frac{\Sigma_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\Sigma_i (X_i - \bar{X})^2}\sqrt{\Sigma_i (Y_i - \bar{Y})^2}}$$

The values of this correlation coefficient are in the $[-1, 1]$ interval, with a value of 1 indicating perfectly correlated variables, and -1 inversely correlated variables. A value of 0 indicates no correlation[2].

Models In this chapter we compare the LDA- and NMF-based vector representations against traditional vector space models (VSMs) baselines. Throughout this chapter we the term *probabilistic methods* for these, in order to distinguish them from the baselines.

As discussed in Chapter 2.1, the main stages of building a VSM can be summarized as: 1) Input matrix extraction, 2) Weighting of vector elements 3) Dimensionality reduction and 4) Vector similarity computation. In order to maintain a setting in which all methods are fully comparable, we test all of them under the same input representation, a simple bag-of-words frequency matrix.

[2]This coefficient is in fact the cosine similarity of the two *centered* ranking vectors. A random variable is centered when the values of the sample have been shifted such that their mean is 0.

Table 5.2 summarizes the different settings we test, both for baselines as well as for the probabilistic methods. For the rest of this section we briefly describe these settings.

	Baselines	Probabilistic methods
1) Input matrix	BoW input matrix	
2) Weighting	tf, tf-idf, pmi, if	-
3) Dim. reduction	LSA	LDA, NMF
K:	$[200 - 1000]$	$\{600, 800, 1000, 1200, 1400\}$
4) Sim. measures	cos, sp, Lin, JS	

Table 5.2: Settings for baselines and probabilistic methods for the word similarity tasks.

1) Input matrix In order to facilitate comparison, both the baselines and the proposed methods tested in this chapter use the same bag-of-words input matrix. This represents target words in terms of co-occurrence with context words, where context is defined as a symmetric window of size five around the target word. In building the matrix the corpus used is the English GigaWord, a collection of newswire text, amounting to a total of 1756M tokens.

For computational efficiency reasons, we prune the input matrix to size 30000×3000: as targets we use the most frequent 30000 words, as found in the corpus; context words are the most frequent 3000 words. A stop word list is used in order to remove pronouns, articles, modal verbs and other function words from the set of context words. In order for the LDA algorithm to scale to the large amount of data, the co-occurrence counts were scaled down by a factor of 70. This is necessary for the LDA method because the collapsed Gibbs sampling algorithm scales linearly also with the total number of tokens in the input data: it runs in $\mathcal{O}(NK)$ time, where K is the number of topics and N the total number of tokens in the input data. The total number of tokens does not influence the running times of the other algorithms and for this reason scaling is performed only for the LDA method.

Table 5.3 shows a fragment of the input matrix. We list the three most frequently co-occurring context words for target words *professor* and *cucumber*.

	university	law	science	pepper	bean	add
professor	42434	13615	7684	19	3	273
cucumber	1	0	0	166	88	79

Table 5.3: Fragment of the input matrix. Target words *professor* and *cucumber*.

2) Weighting schemes Weighting schemes are used in vector space models as a means to enhance the effect of rarely occurring events. The intuition behind is that words occurring very often tend to be less informative as context features and therefore less indicative of a word's meanings. We implement a number of widely used weighting schemes, which have been shown in the past to outperform the use or raw frequency counts.

Term frequency (tf) weighting involves normalizing the rows of the input matrix such that all elements sum up to 1. A cell v_{ij} represents the percentage of t_i co-occurring with context feature c_j out of the total occurrences of t_i. This makes the vector associated to a word less sensitive to its total frequency.

$$tf(v_{ij}) = \frac{v_{ij}}{\Sigma_j v_{ij}}$$

Term frequency-inverse document frequency (tf-idf) is defined as the product between term frequency and inverse document frequency.

$$tf - idf(v_{ij}) = tf(v_{ij}) \times idf(j)$$

where

$$idf(j) = log \frac{J}{|\{j|v_{ij} > 0\}|}$$

where J is the number of columns in the matrix. The inverse document frequency $idf(j)$ turns a context feature j more relevant if its occurrence frequency in the entire matrix is small, indicating a relevant event.

Point-wise mutual information (pmi) is an information-theoretic measure. It quantifies the correlation between two co-occurring events in terms of their independence as random variables.

$$pmi(v_{ij}) = log \frac{P(t_i, c_j)}{P(t_i)P(c_j)}$$

If the two variables (i.e. occurrence of t_i and of c_j) are statistically independent we obtain a pmi value of 0, indicative of no correlation. Correlated variables result in positive pmi values, while negative correlation is indicated by negative pmi. We estimate the probabilities as:

$$P(t_i) = \frac{\Sigma_j v_{ij}}{\Sigma_{ij} v_{ij}} \qquad P(c_j) = \frac{\Sigma_i v_{ij}}{\Sigma_{ij} v_{ij}} \qquad P(t_i, c_j) = \frac{v_{ij}}{\Sigma_{ij} v_{ij}}$$

Finally, we also test the weighting scheme used in the composition models of Mitchell and Lapata [2009]. We denote this weighting scheme as if (inverse frequency); this is computed as the probability of context c_j given t_i, divided by the overall probability of c_j:

$$if(v_{ij}) = \frac{P(c_j|t_i)}{P(c_j)}$$

Simple calculations show that this weighting scheme is identical to point-wise mutual information without the logarithmic scaling:

$$if(v_{ij}) = \frac{P(c_j|t_i)}{P(c_j)} = \frac{P(c_j|t_i)P(t_i)}{P(t_i)P(c_j)} = \frac{P(t_i, c_j)}{P(t_i)P(c_j)}$$

3) Dimensionality reduction As baseline dimensionality reduction we test the widely-used Latent Semantic Analysis method.

LSA computes the singular value decomposition of a $m \times n$ input matrix X: $X = U\Sigma V$. Matrices $U : m \times m$ and $V : m \times n$ are orthogonal (the left and right singular vectors) and $\Sigma : m \times m$ is a diagonal matrix of singular values. By deleting $m - K$ rows from the SVD factorization we obtain $X \approx \hat{U}\hat{\Sigma}\hat{V}$, with $\hat{U} : m \times K$, $\hat{\Sigma} : K \times K$, $\hat{V} : k \times n$. $\hat{U}\hat{\Sigma}\hat{V}$ is the best rank K approximation of X under Frobenius (least squares) norm. We use the target word representation obtained in matrix $\hat{U}$, which maps words to a reduced space of dimension K[3].

Since reduction to any value of K can be obtain from a single SVD decomposition, we test a wider range of values for K, a total of eight values, with K ranging from 200 to 900. The results decrease considerably for K values outside this interval. It is common to apply weighting schemes before computing the SVD decomposition (see for example Husbands et al. [2005] or Aswani Kumar [2009] for discussions on the importance of prior weighting). We perform SVD decomposition on the original matrix of raw counts, as well as after applying tf-idf, pmi and if weighting.

The probabilistic methods we propose use the following vector representations for a target word t_i:

$$v(t_i) = (P(z_1|t_i), ..., P(z_K|t_i))$$

where the components $P(z_k|t_i)$ are estimated using two methods: 1) NMF with multiplicative update rules algorithm and 2) LDA with Gibbs sampling algorithm.

Both these models take K as parameter. Since we are modeling an entire set of 30K words we have experimented with relatively large number of classes. We test five K values: $\{600, 800, 1000, 1200, 1400\}$[4].

The NMF objective function is not defined on 0 values. For this reason, we precede the NMF update rules algorithm by adding to all entries in the matrix a small positive value[5].

Additionally to the K parameter, the LDA model takes as input the α and β hyperparameters. For the α parameter we use $50/K$ which has been reported as a good value for modeling collections of documents and their underlying topics (Griffiths and Steyvers [2004]). Similarly to the topic modeling scenario, we expect the number of senses triggered by a word to be very small in comparison to the total number of senses. Similarly, only a relatively small number of

[3]There are a number of views on why SVD decomposition works as a dimensionality reduction method; an overview of these can be found in Turney and Pantel [2010].

[4]If we see the classes as determining a hard partition on the set of target words, a number of 1000 classes would amount to assigning 30 words to each such cluster.

[5]Arbitrarily set to 1e-05 in these experiments.

context words should be associated to a particular meaning and therefore we also use a small β value of $\beta = 0.01$ (also used before by Porteous et al. [2008]).

4) Similarity measures A large number of vector similarity measures have been proposed in the literature.

Cosine measures the angle between vectors and the intuition behind is that similar words will be vectors pointing in the same direction, although different overall frequency of words might determine very different vector lengths.

$$\cos(v, w) = \frac{< v, w >}{||v|| \, ||w||} = \frac{\Sigma_i v_i w_i}{\sqrt{\Sigma_i v_i^2} \sqrt{\Sigma_i w_i^2}}$$

Lin similarity was introduced in Lin [1998a] and is defined as follows:

$$\text{lin}(v, w) = \frac{\sum_{i \in I(v) \cap I(w)} (w_i + v_i)}{\sum_{i \in I(v)} v_i + \sum_{i \in I(w)} w_i}$$

The i^{th} value of a vector is the pointwise mutual information score and $I(\cdot)$ gives the indices of positive values in a vector.

Divergence measures quantify the distance between two distributions and the inverse of such a distance can be used to measure vector similarity. Such measures can only be applied to proper distributional representations; in our experiments these are the ones obtained with the probabilistic methods as well as those using the tf weighting scheme.

We use the inverse of the Jensen-Shannon divergence, which is defined as follows:

$$\text{D}_{\text{JS}}(p, q) = \frac{1}{2}\text{D}_{\text{KL}}(p|m) + \frac{1}{2}\text{D}_{\text{KL}}(q|m)$$

where $m = \frac{1}{2}(p + q)$ and D_{KL} is the standard Kullback-Leibler divergence:

$$\text{D}_{\text{KL}}(p|q) = \sum_i p_i log(\frac{p_i}{q_i})$$

We use the inverse of the divergence to measure similarity:

$$\text{JS}(v, w) = \frac{1}{D_{JS}(v, w)}$$

Scalar (dot) product is the simplest function we implement for returning similarity scores:

$$sp(w, w) =< v, w >= \Sigma_i v_i w_i$$

5.2 Model selection

Throughout this section we investigate the effect of different parameter settings for the probabilistic models, namely: the effect of dimensionality K, of the similarity measure used and of the number of iterations for which the two algorithms are run. Parameter tuning is performed also for the baselines, however a detailed account of the results obtained is not provided here, as this is not the focus of the evaluation. The best results and corresponding settings for both baseline and probabilistic methods are reported in Section 5.3.

Dimensionality K and similarity measure

In this section we investigate the effect of the dimensionality K and of the similarity measure on the performance of the models.

Together with the individual models obtained with each of the five K values, we also experiment with a mixture model. A mixture model combines the individual predictions of different models in order to obtain a single aggregate prediction. It is generally known that mixing the predictions of models with different parameter settings may lead to an increase in predictive power (see for example Reisinger and Mooney [2010] for the same observation in a similar experimental setting).

We test a mixture obtained from the similarity scores returned when using each of the five K values. This is a straightforward mixture setting, in which all the single scores are given equal weight by averaging over them:

$$sim_{MIX}(v,w) = avg(\sum_{K} sim_K(v,w))$$

where v an w are two words and $sim_K(v,w)$ is the similarity between them as returned by a model with K latent classes. The equal-weights mixture turns both the tuning of mixture weights, and of parameter K, unnecessary.

We test three similarity measures: scalar product (sp), cosine similarity (cos) and inverse Jensen-Shannon divergence (JS). The use of Lin similarity is not appropriate in this case, as it is defined on vectors of pmi values.

For the results reported in this section we run the LDA Gibbs sampling algorithm for 1000 iterations and we average over three chain samples (at 800, 900 and 1000 iterations) to obtain estimates of the $P(z_k|t_i)$ distributions[6]. We run the NMF multiplicative rules algorithm for 100 iterations. In order to minimize the effect of the initial random initialization of this algorithm, we start two independent runs for each K value and average over their predictions.

The results for the individual K values as well as for the mixture methods, for different similarity measures, are plotted in Figures 5.1, 5.2, 5.3 and 5.4. The

[6]Averaging over different states of the Monte Carlo Markov Chain chain is standard procedure in topic modeling. We observe no topic drift.

LDA method yields the best results when using scalar product as similarity measure (Figure 5.1). This is followed by the use of JS, which outperforms cosine (Figures 5.3 and 5.4).

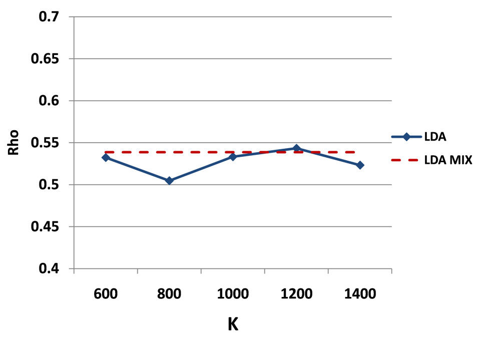


Figure 5.1: LDA-Scalar product

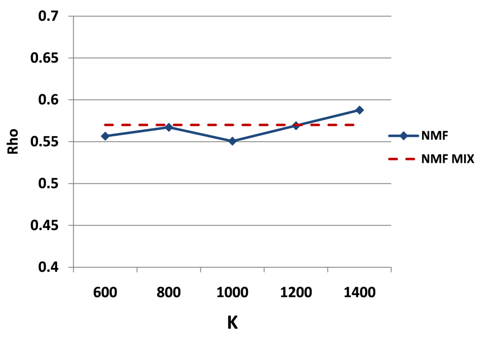


Figure 5.2: NMF-Scalar product

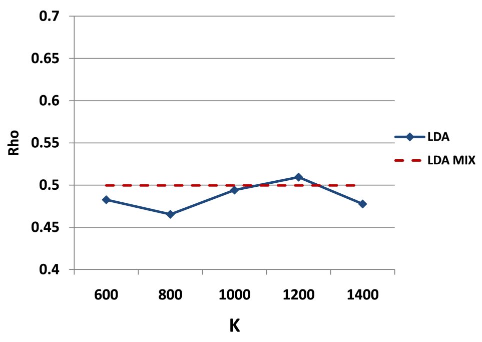


Figure 5.3: LDA-Cosine

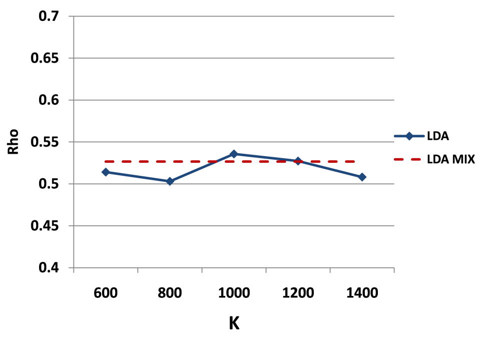


Figure 5.4: LDA-Inverse JS

Irrespective of the similarity measure used, the LDA models perform better with larger number of topics and their performance peaks at K = 1200 for cosine and scalar product and K = 1000 for JS similarity. Mixture models are, however, significantly better than the average performance for all similarity measures and they are only slightly outperformed by the best single settings in all cases.

Figure 5.2 plots the results obtained with the NMF method using scalar product similarity. Scalar product is again the best way of measuring similarity, outperforming JS, which in turn outperforms cosine. As it can be observed NMF mixture outperforms the LDA mixture by over 3% in Spearman Rho correlation. Unlike LDA, the best results are obtained when using the largest K setting tested. This is similar to the findings of Hofmann [1999], where the performance of the closely-related PLSA model increases with the dimensionality K, without reaching an upper bound (K is however bounded by computational efficiency considerations). The mixture setting is consistently performing well for the NMF method as well.

To summarize, both methods prefer more complex models, with large K values.

Mixing over all K settings, which turns parameter tuning unnecessary, is a good strategy which significantly improves over the average performance. In terms of similarity measures, we observe that scalar product and inverse JS divergence perform best, again for both methods. This result corresponds to the intuition that, in the probabilistic framework we have proposed, scalar product and divergence methods are particularly motivated similarity measures, as detailed in Section 3.2.3.

Number of iterations

A second aspect we investigate is the correlation between the number of iterations of the two algorithms and the performance of the models obtained. Throughout this section we discuss the convergence of the two algorithms and the use of early stopping.

Convergence The Gibbs sampling LDA algorithm generates a chain of samples which is proven to converge to the desired posterior distribution. In reality, it is difficult to determine the convergence of the chain, and several methods are available to measure the variation within a chain (Gilks [1999]). If there is sufficient evidence for convergence one can either use a chain state (a sample) which maximizes the likelihood of the data, or average over a number of samples at a lag of, usually 50 or 100 iterations. In practice, it is common to use the model obtained after the chain runs for a large number of iterations (usually more than 1000, also called *burn-in* period) without the check for convergence.

The convergence of the NMF multiplicative rules algorithm can be monitored through the KL-divergence error function. One our data, we observe that a few hundred iterations are sufficient to result in a minimal error reduction rate.

Early stopping In the topic modeling literature, the use of early stopping has been proposed as a technique for preventing overfitting. The intuition behind this is that instead of running an algorithm until it has converged, a model obtained at an earlier iteration might perform better. At an earlier iteration the error function has not reached a local minimum, therefore the model learned does not perfectly fit the observed data. This in turn may help prevent overfitting and generalize better to unseen data. In the topic models literature early stopping has been employed for the different variations of the EM algorithm for PLSA in Hofmann [2004] or Hofmann [1999]. In this work held-out data is used to determine a good stopping iteration.

In order to investigate whether a similar effect can be observed here, we perform evaluations of the LDA and NMF methods using earlier iterations.

In the previous section we have run the LDA Gibbs algorithm for 1000 iterations and averaged over 3 iterations at lag 100. In this section we evaluate the the algorithm when using the output from 100 to 1000 iterations[7].

The performance of the LDA model, for each similarity measure, is plotted in Figure 5.5. For simplicity, the LDA model plotted is the mixture over all K values, however the same trends in the graphs are also observable for the individual models.

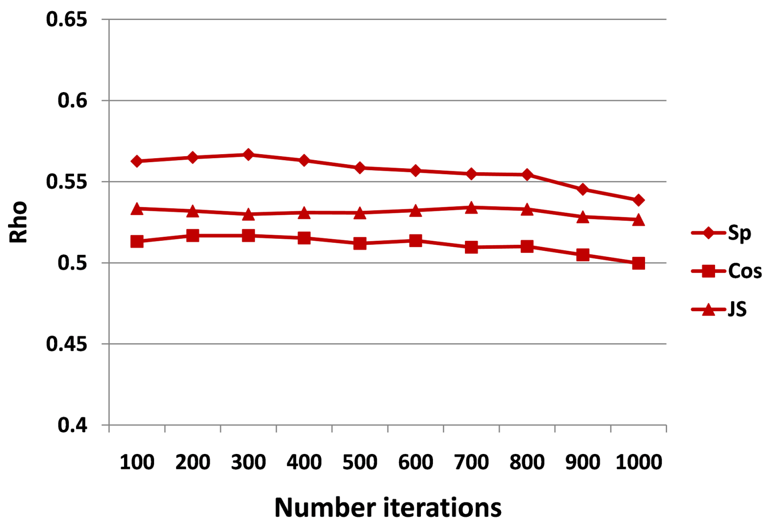


Figure 5.5: LDA MIX across different number of iterations

In the case of scalar product, the best performing similarity measure, we observe a significant drop in performance with the increase in the number of iterations, of approximately 3% from the peak, at iteration 300, to iteration 1000. When using cosine, also peaking at 300 iterations, we can observe a similar trend with a total performance drop of 2%. The JS measure makes a distinctive note as it reaches its best results after 700 iterations.

For the NMF formulation, we continue to run the algorithm for 300 iterations and investigate the results at every 50 iterations step. We observe however that the performance peaks at as early as iteration 100. For this reason we test results every 10 iterations until iteration 100 is reached. We observe that the performance peaks at around 50-100 iterations, with slight variations depending on the similarity measure used. The best setting, scalar product, peaks at iteration 70. After 300 iterations we observe a significant total drop of 5% in Rho correlation, irrespective of the similarity measure used. Iterations 30 through 100 are plotted in Figure 5.6.

Overall we observe that both methods benefit from the use of early stopping, for all K values tested[8]. This may be caused by the models not being fully

[7]When running the algorithm for, e.g. 300 iterations, we again average over three samples at lag 100 (iterations 100, 200 and 300); at iteration 100 we only use the current state rather than averaging over multiple states.

[8]In particular, in both algorithms, specific to earlier iterations are distributional represen-

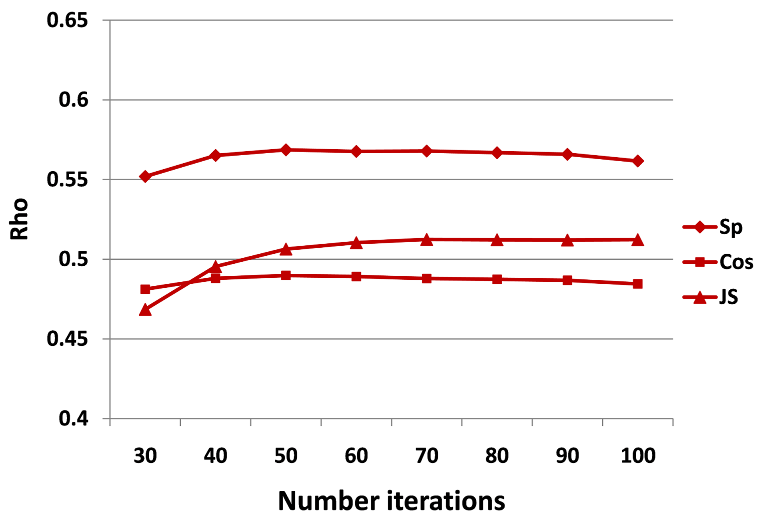


Figure 5.6: NMF MIX across different number of iterations

adequate for the task we test on, or by an overfitting effect. In future work we plan experiment with monitoring log-likelihood on held-out data in order to investigate whether the observed behavior is caused by overfitting; overfitting is not unlikely given the high complexity, i.e. the large dimensionality K, of the models.

5.3 Results

In Table 5.4 we list the Spearman Rho correlation scores of the best performing settings for each type of model.

The simple vector space model (SVS) performs best with *pmi* weighting and Lin similarity. *If* weighting together with cosine similarity is the next best setting. These two configurations outperform by a large margin any other combinations of similarity measures and weighting schemes.

As previously discussed, LSA dimensionality reduction is performed on raw counts as well as with prior weighting functions. We observe that the weighting scheme has a dramatic effect on the performance of the LSA method, with *if* outperforming the other methods by a margin of over 10%. The best results using LSA are obtained with *if* weighting, K=600 and using scalar product as similarity. This setting outperforms the SVS models.

For the LDA and NMF models, we report results obtained with early stopping, i.e. 300 iterations for LDA and 70 iterations for NMF. The similarity measure

tations which are less sparse, "less focused". With larger number of iterations, these become more sparse, with only a few topics sharing most of the distribution mass.

used is scalar product. The LDA and NMF methods perform similarly to LSA. When using early stopping, both LDA and NMF reach their best performance with the largest K value of $K = 1400$ classes. Despite the fact that the LDA model is at disadvantage, as it uses input data which is two orders of magnitude smaller, it performs comparably to the other methods.

Model	Weighting	K	Sim. measure	Pearson
SVS	pmi	-	lin	53.15
SVS	if	-	cos	55.96
LSA	if	600	sp	58.20
NMF	-	1400	sp	57.89
LDA	-	1400	sp	58.22
LSA_{MIX}	if	Mix	sp	58.00
NMF_{MIX}	-	Mix	sp	56.79
LDA_{MIX}	-	Mix	sp	56.49

Table 5.4: Model results on out of context word similarity.

The second part of the table shows the results for the mixture models. For comparison, we also present an LSA mixture model. The mixtures perform slightly worse than their best individual models, however, they are more robust as they do not require parameter tuning.

5.4 Discussion

Throughout this section we exemplify the distributional representations induced in our framework[9] and following this we examine some of the errors made on the word similarity data.

Tables 5.5 and 5.6 show the representations for the pair of words ranked as most similar, according to all similarity measures: (*psychology, psychiatry*). Words are described as distributions over classes, the top most probable ones being given in Table 5.5. A class is described by a distribution over context words. These are assigned names in 5.5 and further described by their most likely context words in Table 5.6.

As it can be observed, the classes obtained for words *psychiatry* and *psychology* capture prototypical contextual features, rather than distinct meanings. Both words are described by two university-related classes and a third class indicating the context of science, medicine. The two words only differ in their fourth most frequent class, and these differences intuitively match the meaning difference between the two words: for *psychiatry* this class points to a medical domain *illness, hospital* while for *psychology* we find a class containing verbs such as *like, do, take, include* which may refer to the study of psychology. The pair is

[9]as outputted by a single LDA model, K=1400 after 300 iterations

	Probability	Class
psychiatry	0.26	UNIVERSITY 1
	0.19	UNIVERSITY 2
	0.06	SCIENCE-MEDICAL
	0.05	MEDICINE
psychology	0.29	UNIVERSITY 1
	0.16	UNIVERSITY 2
	0.06	SCIENCE-MEDICAL
	0.05	VERB

Table 5.5: Distribution over latent classes for target words *psychology* and *psychiatry*

Class	Word Distribution			
UNIVERSITY 1	0.12 *professor*	0.05 *school*	0.04 *harvard*	0.04 *university*
UNIVERSITY 2	0.47 *university*	0.06 *professor*	0.04 *school*	0.03 *study*
SCIENCE-MEDICAL	0.11 *science*	0.04 *journal*	0.04 *academy*	0.03 *medicine*
MEDICINE	0.28 *medical*	0.08 *say*	0.06 *doctor*	0.05 *hospital*
VERB	0.06 *like*	0.06 *do*	0.05 *take*	0.05 *include*

Table 5.6: Context word distributions associated to latent classes

scored very similar also by human judges receiving a mean similarity score of 8.08.

As a second example, we consider the words *gin*, *vodka* and *brandy* present in the WordSim data. Their representations are given in Table 5.7.

	Prob.	Class	Word distribution
gin	0.12	ALCOHOL-DRINK	*wine, drink, bottle, beer*
	0.02	GAME	*game, play, team, match*
vodka	0.12	ALCOHOL-DRINK	*wine, drink, bottle, beer*
	0.12	COMMERCIAL	*sell, product, company, market*
	0.04	RUSSIA	*russia, russian, boris, moscow*
brandy	0.24	ALCOHOL-DRINK	*wine, drink, bottle, beer*
	0.09	RECIPE	*1, fresh, cup, 2, add, green*
	0.09	CELEBRITY	*star, play, actor, movie, film*

Table 5.7: Distribution over latent classes for target words *gin*, *vodka* and *brandy*.

We observe that for the word *gin*, actual sense distinctions are captured: a card game class is the second most frequent class, however much less likely than an alcoholic beverage component. For the word *vodka*, which is un-ambiguous, we observe that the representation captures again *aspects* of its meaning; specific to this word are RUSSIA and COMMERCIAL classes. The method finds that specific to *brandy* are a RECIPE class (perhaps reflecting the use of brandy as a cooking ingredient) and a CELEBRITY class (possibly reflecting *Brandy* occurring as a girl's name).

The similarity scores assigned by humans disagree to a large degree from those assigned by our model in a number of cases. The most systematic error we encounter suggests an inadequacy of the newspaper domain input corpus. Consider for example the pair (*planet*, *star*) in Table 5.8. The word *star* is predominantly used with the meaning of *celebrity* in the input corpus, and for this reason the high similarity that humans assign to this pair is not reflected in the representations we induce.

Other examples of this are (*magician*, *wizard*) (with *wizard* predominantly occurring in its software sense[10]) or (*tennis*, *racket*) (with *racket* occurring frequently in its *illegal business* sense).

	Prob.	Class	Word distribution
planet	0.18	PLANET	*earth, surface, atmosphere*
	0.07	FIND	*find, know, discover, show*
star	0.04	ENTERTAINMENT 1	*film, movie, show, like, star*
	0.03	ENTERTAINMENT 2	*actor, do, play, director, star*
wizard	0.10	SOFTWARE	*software, computer, program*
	0.08	ENTERTAINMENT 1	*film, movie, show, like, star*
magician	0.08	VERB	*like, do, make, see, get*
	0.02	NAME	*david, mary, howard, clark*

Table 5.8: Meaning representations of word pairs (*planet*, *star*) and (*magician*, *wizard*) which are assigned high similarity by humans and low similarity by the system.

This points to an interesting aspect of this data: when given ambiguous words, human participants seem to rate their similarity by assigning meanings which bring the two words closer together. In the out-of-context scenario, our models are however designed to return an average similarity score, over *all* the possible meanings. This observation indicates that the performance on this task can be improved by using a different similarity computation strategy[11]. An optimal similarity computation strategy may depend, however, on the concrete application requiring the comparison of words. For this reason, it is out of our scope here to further adapt the similarity computations to the task of predicting human similarity judgments.

[10]A software wizard is a user interface element.

[11]In the same line, Reisinger and Mooney [2010] propose a MaxSim strategy. In their work words are represented as sets of clusters and the idea is to return the similarity between the two most similar clusters (corresponding to the most similar meanings that the two words have).

5.5 Summary

Throughout this section we have tested the use of LDA and NMF as dimensionality reduction methods and compared them against a set of baseline methods for the task of assessing word similarity. In particular, we have focused on determining optimal settings, both for the probabilistic methods as well as for the baselines, which we further employ on the task of lexical substitution in Chapter 6.

For the probabilistic methods, we have observed that larger K values perform better, however a basic mixture model in which all K settings receive equal weight achieves very good performance. The advantage of using a mixture model is that it turns parameter tuning unnecessary, as an optimal K value need not be determined.

A second observation we make is that probabilistic methods perform better on this task when using earlier iterations of their corresponding algorithms. In future work we plan to investigate if this is an overfitting effect, determined by the high complexity of our models.

Finally, we have qualitatively analyzed the data and exemplified the meaning representations obtained with the probabilistic models. We observe that, as expected for a data-driven method, the classes induced reflect *aspects* or *components* of meaning, rather than distinct dictionary-like senses.

Chapter 6

Lexical Substitution

This chapter contains the main evaluation of the proposed framework, on the task of computing lexical similarity in context.

While traditionally, distributional models of semantics are targeted at representing isolated words, most end-user NLP applications need to model the meaning of words *occurring in context.* For this reason, distributional meaning representations that are sense-specific or context-sensitive have been in the focus of recent work. These aim at modeling meanings beyond the level of isolated words while remaining within the distributional representation paradigm and retaining its advantages.

Throughout this chapter we address the task of computing word similarity in context by making use of the SemEval 2007 Lexical Substitution Task (LST) benchmark dataset (McCarthy and Navigli [2007]), in which systems are required to find appropriate substitutes for target words occurring in sentential context.

This task has been originally proposed as an alternative to the Word Sense Disambiguation task. In language technology, Word Sense Disambiguation (WSD) has been proposed as a stand-alone task to help develop computational methods for assigning senses to words occurring in context. The desired goal is to develop ambiguity resolution modules to be integrated in end-user applications such as machine translation or information retrieval. The implicit assumption of the WSD task is the view on word meaning as a list of separate senses, out of which one or more are triggered by an occurrence of the word. The foundations of this assumption have been subject to criticism both from a theoretical point of view, by lexicographers, as well as from an application perspective due to the observed difficulties encountered by computational models for WSD. Thus, it is commonly agreed upon that WSD, at least in its traditional formulation, has failed to prove itself useful in end-user applications (Resnik [2006]). It has been, however, argued that the need for word sense resolution is in fact the need for assessing context-aware meaning similarity. This has lead to the proposal of lexical substitution and cross-lingual lexical substitution as stand-alone tasks. In these tasks, the goal is to find context-appropriate substitutes for a word either within a mono-lingual setting (e.g. for information retrieval) or in a cross-lingual setting (e.g. for machine translation). These tasks circumvent the need for sense inventories or explicit sense assignment by casting the meaning resolution problem as a (context-aware) similarity computation problem, which is at the core of most language technology applications.

6.1 Experimental setup

Data We make use of the SemEval 2007 Lexical Substitution Task (LST) benchmark dataset of McCarthy and Navigli [2007]. In the lexical substitution task, a system is provided with a target word occurring in sentential context and has to identify appropriate substitutes for it. The LST dataset contains 200 target words, namely nouns, verbs, adjectives and adverbs, each of which occurs in 10 distinct sentential contexts. The total set contains 2,000 sentences. Five annotators were asked to provide substitutes for these target words.

The following two sentences are instances of the target adjective *still* in the LST data.

(1) It is important to apply the herbicide on a *still* day, because spray drift can kill non-target plants.

(2) A movie is a visual document comprised of a series of *still* images.

For sentence (1), the annotators provide the following substitutes: *calm* (provided 5 times), *not windy* (1) and *windless* (1). In sentence (2) the adjective has a completely different meaning, which is reflected in the corresponding substitutes: *motionless* (3), *unmoving* (2), *fixed* (1), *stationary* (1) and *static* (1).

We use the test portion of this data set (containing 90% of the total data). Further on we prune this data set by eliminating multi-word expressions provided as substitutes, as well as words that are not in our vocabulary[1]. Sentences that are left with no valid substitutes are eliminated.

Table 6.1 summarizes the data used. We give the total number of substitutes provided for each part of speech tag, on average, as well as the average number of substitutes that are correct for each instance. As it can be observed, the less ambiguous data is the adverb data where one third of the candidate substitutes are correct. At the other extreme, verbs are the most difficult part of speech, with approximately 15% percent of the substitutes being correct.

[1]The input matrix contains the 30000 most frequent words found in GigaWord. Some of

	Adv	Adj	Noun	Verb	All
#Sentences	286	464	490	435	1675
#Total subst.(avg.)	10.7	18.0	16.5	19.7	16.7
#Correct subst.(avg.)	2.9	3.7	3.2	3.5	3.4

Table 6.1: Lexical substitution data

The typical approach to the LST task involves two steps. Initially, a set of *candidate* substitutes is generated with the use of inventories such as WordNet, while in a second stage such candidate lists are ranked. Our goal is not that of building a complete lexical substitution system, but rather on computing context-sensitive similarities for solving the paraphrase ranking subtask.

For this reason we use the LST data similarly to Erk and Padó [2008] and to subsequent work. In this setting, *all* the (gold) substitutes for each target are pooled together, which creates a total list of possible substitutes for each target word. Following this, the model has to produce a ranking of this total set of substitutes, given a specific sentential context. Ideally, the context-appropriate substitutes will be ranked high in detriment of the substitutes corresponding to different senses of the target word.

Evaluation metrics and significance testing The original task proposes precision out of ten as a scoring method. However, as shown in Thater et al. [2009], this evaluation metric is not appropriate for this experimental setting. For this reason, we use as evaluation metrics a rank correlation measure, Kendall τ, as well as Generalized Average Precision (GAP) in order to facilitate comparison with previously reported results. We further detail on these two measures.

Kendall τ is a non-parametric measure for computing rank correlation. Kendall τ_a is defined as:

$$\tau_a = \frac{n_c - n_d}{n(n-1)/2}$$

where

n_c	=	number of concordant pairs
n_d	=	number of discordant pairs
n	=	total number of pairs

A concordant pair is a pair of data instances for which the two ranks agree, i.e. their relative order is the same in both lists, while a discordant pair is a pair for which the ranks disagree. If one of the ranks is the gold standard rank, the measure can be seen as penalizing inversions of the relative order of any two elements in this gold rank.

Kendall τ_b correlation is simply adjusted to cases in which ranks contain tied

the LST words are not in this list.

pairs. Given two ranks t and u, this is defined as:

$$\tau_b = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}$$

where

n_0	$=$	$n(n-1)/2$
n_1	$=$	$\Sigma_i t_i(t_i - 1)/2$
n_2	$=$	$\Sigma_i u_i(u_i - 1)/2$
t_i	$=$	Number of tied elements in the i'th group of tied elements of rank t
u_i	$=$	Number of tied elements in the i'th group of tied elements of rank u

The quantity $(n_0 - n_1)$ stands for the number of element pairs in the first rank for which a strict order relation holds (i.e. elements sharing the same position in the rank do not form such a pair, therefore their cardinality n_1 is discounted). The quantity $(n_0 - n_2)$ represents the corresponding number of "valid" pairs in the second rank.

We also report results using a second metric, Generalized Average Precision (GAP)(Kishida [2005]). Average precision is a measure typically employed in information extraction and is defined as follows:

$$AP = \frac{\Sigma_{i=1}^{n} x_i p_i}{R} \qquad\qquad p_i = \frac{\Sigma_{k=1}^{i} x_k}{i}$$

where x_i is a binary variable indicating whether the ith item as ranked by the system is in the gold standard or not, R is the size of the gold standard, and n the number of candidate substitutes to be ranked by the system.

Since the gold standard of our data also associates weights to substitutes, we use *generalized average precision* instead which extends average precision to:

$$GAP = \frac{\Sigma_{i=1}^{n} I(x_i) p_i}{R'} \qquad\qquad R' = \Sigma_{i=1}^{R} y_i$$

where $I(x_i) = 1$ if x_i is larger than zero, zero otherwise, and $y_1, \ldots, y_R$, the frequencies of paraphrases 1 to R of the ideal ranked list of paraphrases.

The GAP measure differs to rank correlations as it also takes confidence weights into account, these being given by the number of people that suggest a particular substitute word.

For significance testing, we use random shuffling (Yeh [2000]). Randomized testing has the advantage of making no assumptions about the distribution of the data, being one of the most reliable methods of significance testing. These methods are very computational-intensive but usable on our data due to the relatively small number of instances available.

In more detail, given two system outputs, the null hypothesis (i.e., that the two predictions are indistinguishable) is tested by randomly mixing the individual instances (in our case sentences) of the two outputs. We run this for a standard number of one million iterations. We count how many times the difference in performance between the two (shuffled) outputs is equal or greater than the difference observed between the original two outputs. This divided by the number of iterations gives the significance level p. This measures the probability that the same (or larger) difference in performance is observed under the null hypothesis that the two systems are actually indistinguishable.

Models In this section, we overview the models tested on the lexical substitution task. In particular, we test two sets of baselines (context-ignoring and context-sensitive baselines) against the probabilistic models obtained in our framework.

Context-ignoring baselines The first baseline we report is a simple vector space (SVS) that does not use any contextual information. This baseline returns the same ranking of the substitute candidates for each instance, based solely on their similarity with the target word. We test the two best performing settings for the word similarity task: if weighting with cosine similarity and pmi weighting with Lin similarity.

Context-sensitive baselines Our baselines for similarity in context are vector addition and vector multiplication of Mitchell and Lapata [2008]. In Mitchell and Lapata [2009] the authors show that multiplication approximates the intersection of the meaning of two vectors, whereas addition approximates their union.

More precisely, given a target word t and context word c, represented in a J-dimensional vector space as:

$$v(t) = (t_1, ..., t_J)$$

$$v(c) = (c_1, ..., c_J)$$

the composed vector representations are obtained as:

$$v_{Add}(t, c) = v(t) + v(c) = (t_1 + c_1, ..., t_J + c_J)$$

$$v_{Mult}(t, c) = v(t). * v(c) = (t_1 * c_1, ..., t_J * c_J)$$

Any input vector representation can be used with these composition methods and for this reason we test 1) the simple semantic space (SVS) (as originally proposed by the authors) as well as 2) dimensionality-reduced representations obtained from LSA, LDA and NMF. We use notation such as Add_{LSA} to stand for addition as composition method and LDA as underlying vector representation.

The reason for choosing this baseline is twofold. Firstly, it has been shown that it is a competitive model, when applied both on measuring contextual similarity and on language modeling. A second motivation stems from the fact that this method is very easy to replicate; we can therefore compare all the models in an identical setting, in which differences in results can be explained solely by the their performance and not by differences in the experimental set-up. A comparison with other *reported* results on this data set is performed in Section 6.2

Interaction between composition methods and weighting schemes Mitchell and Lapata show that multiplication outperforms addition, when using the weighting scheme: $if(t,c) = \frac{P(t,c)}{P(t)P(c)}$, with t-target word and c-context word. The intuition behind this is that multiplication approximates meaning intersection, as it results in probabilities of context features given both target and context word[2].

However, component-wise composition operations cannot be interpreted independently of the weighting scheme used. For example, the effect of multiplication on if weighting is obtained by performing *addition* on *pmi* weighting. More precisely, as we have pointed out in Chapter 2, the *pmi* weighting scheme is identical to if on logarithmic scale:

$$pmi(v(t)) = log(if(v(t)))$$

For this reason, component-wise addition on *pmi* weights corresponds to multiplication on if weights:

$$pmi(v(t)) = (pmi(t_1), ..., pmi(t_J)) = (log(if(t_1)), ..., log(if(t_J)))$$

$$\begin{aligned} pmi(v(t)) + pmi(v(c)) &= (pmi(t_1) + pmi(c_1), ..., pmi(t_J) + pmi(c_J)) \\ &= (log(if(t_1)) + log(if(t_1)), ..., log(if(t_J)) + log(if(t_J))) \\ &= log(if(v(t)). * if(v(c))) \end{aligned}$$

In turn, different similarity measures are best associated with these weighting schemes. Lin similarity, which is defined in an additive fashion, is used with *pmi* weighting, while cosine similarity which is defined in a multiplicative fashion is better suited for if weighting:

$$\cos(v,w) = \frac{\Sigma_i v_i w_i}{\sqrt{\Sigma_i v_i^2}\sqrt{\Sigma_i w_i^2}}$$

$$\text{lin}(v,w) = \frac{\sum_{i \in I(v) \cap I(w)} (v_i + w_i)}{\sum_{i \in I(v)} v_i + \sum_{i \in I(w)} w_i}$$

where $I(v)$ gives the indexes of the positive values of a vector v.

[2]This aspect is detailed in Chapter 2.

Probabilistic models Our models use the following contextual representation, as detailed in Chapters 3 and 4:

$$v(t, c) = (P(z_1|t, c), ..., P(z_K|t, c))$$

where the components $P(z_k|t, c)$ are estimated using LDA and NMF.

We test the two mixture models, which we denote as $\text{Cont}_{\text{NMF}_{\text{MIX}}}$ and $\text{Cont}_{\text{LDA}_{\text{MIX}}}$. Parameter tuning is not necessary for these models as they return the average similarity score over the predictions of each K value. We also test the best individual models as determined on the word similarity task in Chapter 5. These are K=1400 with early stopping: 300 iterations for the LDA models and 70 iterations for NMF. We denote these models as Cont_{NMF} and Cont_{LDA}.

Treatment of context Our definition of context is identical for all methods. Context words are considered to be words occurring within a 5-word window around the target word. The probabilistic methods use only the most frequent 3000 words as context features; for this reason we can only use the context words that are in this list, rather than all 10 context words that are extracted for each instance. This results in using on average four context words for each instance; all methods tested in this section use these same context words. Table 6.2 exemplifies the data used as input for the sentences in (1) and (2). We list the entire candidate substitutes list, with the correct ones indicating the number of people suggesting them in parentheses.

Target word	Context words	Candidate substitutes
still	apply day	calm(5), windless(1), static, hushed, arranged, serene, smooth, fixed, stationary, motionless, quiet, peaceful, tranquil, unruffled
still	image series	calm, windless, static(1), hushed, arranged, serene, smooth, fixed(1), stationary(1), motionless(3), quiet, peaceful, tranquil, unruffled

Table 6.2: Input data for target word *still* in sentences in (1) and (2).

To create a ranking of the candidate substitutes, we compose the vector of the target with its context and compare it with each substitute vector. Given a set of context words, we contextualize the target using each context word at a time and multiply the individual similarity scores. We contextualize just the target word and not the candidate substitutes. This approach has been shown to bring higher discriminative power to the models, as opposed to performing comparisons with both target and substitute embedded in an identical context (Thater et al. [2010]).

6.2 Results

The results on lexical substitution are shown in Tables 6.3 (baselines) and 6.4 (probabilistic methods) .

Baselines Context-ignoring vector space models can be very competitive baselines on this task, as previously observed by Erk and Padó [2008] and by subsequent studies. In particular, we observe that an adequate choice of weighting scheme and similarity measure further improves the performance of such a baseline. The best simple vector space model (SVS) is obtained using *pmi* weighting and Lin similarity. As a comparison, this context-ignoring method scores 41.29 in GAP which is significantly higher than previously reported context-ignoring baselines on the LST data [3].

Model	Setting	Kendall's τ_b	GAP
Random		0.0	31.61
SVS	pmi, lin	12.09	41.29
Add_{SVS}	pmi, lin	**15.14**	**43.41**
Add_{LSA}	cos	11.03	39.02
Add_{LDA}	cos	12.19	40.08
Add_{NMF}	cos	13.79	41.84
Mult_{SVS}	if, cos	**14.67**	**42.14**
Mult_{LSA}	cos	0.0	31.61
Mult_{LDA}	cos	12.04	40.13
Mult_{NMF}	cos	14.61	41.79

Table 6.3: Baseline results on LST data.

Model	Similarity	Kendall's τ_b	GAP
Cont_{NMF}	JS	**17.57**	**44.94**
Cont_{LDA}	sp	14.60	42.18
$\text{Cont}_{\text{NMF}_{\text{MIX}}}$	JS	**17.38**	**44.67**
$\text{Cont}_{\text{LDA}_{\text{MIX}}}$	sp	15.66	42.66

Table 6.4: Probabilistic methods results on LST data.

The additive and multiplicative compositional models on the simple vector representations (Add_{SVS} and Mult_{SVS}) significantly improve over this baseline. We observe that the two best combinations are addition with *pmi* weighting and Lin similarity and multiplication with *if* weighting and cosine similarity. As discussed in Section 6.1, this result is to be expected as these two combinations approximate the same type of composition, the one proposed in Mitchell and Lapata [2008]. Other combinations of weighting schemes and similarity measures delivered significantly lower results.

[3] 33.04 GAP in Thater et al. [2009], 34.6 in Erk and Padó [2010] and 37.65 in Thater et al. [2010]

We also test the additive and multiplicative compositional models on dimensionality-reduced representations obtained by performing LSA, LDA and NMF. The scores are lower when using these representations over the SVS one. In particular, we observe that multiplication and LSA performs random. Addition and LSA, while not achieving high performance, does significantly improve over random. This corroborates previous work, in which LSA representations have been composed in an additive fashion to obtain sentence representations.

Probabilistic models We list the results of the probabilistic methods in Table 6.4.

Unlike on the word similarity task, we observe that the choice in similarity measure does not make a dramatic difference in the performance of the models. In particular, the LDA models work best with scalar product while the NMF ones prefer the use of inverse JS divergence. These two measures slightly outperform the use of cosine in both types of models.

We observe that both NMF models outperform the best baseline compositional models. We perform significance testing to compare the two NMF runs with the two best baselines: Add_{SVS} and Mult_{SVS}. The differences are highly significant at levels $p < 0.001$[4].

LDA models achieve similar performance to the best baseline composition methods. As discussed in Chapter 5, the computational complexity of the Gibbs sampling LDA algorithm determines us to use considerably less input corpus data for LDA, in comparison to the other methods. In Dinu and Lapata [2010b] we show that when using the same, smaller amount of input data for all methods, LDA also significantly outperforms these baselines.

As shown in Table 6.4, using LDA and NMF representations as input for the *baseline* addition and multiplication composition methods does not lead to performance gains. Thus we can conclude that the performance of our framework is not solely due to the use of LDA or NMF as (superior) dimensionality reduction methods.

Comparison to other reported results The remainder of this section compares the proposed probabilistic methods with previously reported results on the LST dataset.

A number of methods for computing context-sensitive distributional similarity have been tested on the Lexical Substitution data, however a direct comparison is not straightforward due to the differences in data subsets as well as in the

[4]In general, we observe that even small difference in GAP scores correspond to statistically significant differences. This has also been observed in Erk and Padó [2010], where the authors note that, as a rule of thumb, differences of 0.7 in GAP correspond to highly significant $p < 0.01$ levels.

experimental settings specific to each of these methods. In this section, we focus on the comparison with the best reported results on this data, in Thater et al. [2010] (henceforth TFP). This has been made possible by the fact that the exact test data was made available to us by the authors.

The main difference to the data set in Table 6.4 is the way the context words are extracted from the sentential context. More precisely, TFP10 propose a syntactic model, in which composition is guided by syntactic dependencies. Context features are selected to be all the words which stand in a syntactic relation with the target word, in a given sentence. This differs to the previous experiments in which the context words were extracted from a window of size 5 around the target word[5].

We re-run our experiments in order to test the proposed probabilistic methods on this data set. As described in Chapter 5, for computational reasons, the first experiment reduces the number of context features to the most frequent 3000 words. In order to be able to use *all* syntactic neighbors as context words, for this second experiment we use an input matrix in which the context features are no longer chosen to be the top most frequent words, but the context words present in the TFP dataset. This way, the identical context words used by the TFP method can also be used in our experiment[6]. We use our methods with the settings determined in Chapter 5; more precisely we re-train a single $K = 1400$ NMF model and the mixture LDA one. NMF uses JS similarity and LDA scalar product similarity.

We report on a number of methods have been tested on the LST dataset:

- EP08: Method of Erk and Padó [2008], as reported by Erk and Padó [2010]. Available only for verb data.
- EP10: Erk and Padó [2010]
- ML08-1: This is identical to the baseline setting Mult_{SVS} from Section 6.2; it uses if weighting and cosine similarity, as originally proposed by Mitchell and Lapata [2008].
- ML08-2: Identical to the baseline setting Add_{SVS} from Section 6.2; it uses *pmi* weighting and Lin similarity, and can be seen as a variant of the composition method of Mitchell and Lapata.
- TDP09: Thater et al. [2009] (Verb data)
- TFP10: Thater et al. [2010]

[5]There is a second, less relevant difference in the actual data instances and candidate substitutes used. Just as in the previous experiments, some of the data is eliminated due to constraints imposed by the experimental setting.

[6]Approximately 50% of the context features used are also in the top 3000 most frequent words.

- Cont_{NMF}: single $K = 1400$ NMF model with early stopping and JS similarity.
- $\text{Cont}_{\text{LDA}_{\text{MIX}}}$: mixture LDA model, early stopping and scalar product similarity.

However, it should be pointed out that, for a number of reasons, all these methods, with the exception of ML08-1 and ML08-2 , are not directly comparable to our method. Similarly to EP10, our models do not use syntactic information. However Cont_{NMF} uses an input corpus which is one order of magnitude larger than the one used by EP10 (GigaWord, not scaled, vs. BNC). However, the best result of EP10, which we report here, is obtained when tuning the model's parameters on test data, which puts the rest of the methods at significant disadvantage. Furthermore EP10, unlike all the other methods here, is not a vector space model, as the input data is not a matrix which sums up over all occurrences of words. This model stores and uses *individual* occurrence vectors, which is a richer, more informative, representation in comparison to that of vector space models.

Cont_{NMF}, TFP10 and TDP09 are comparable with respect to the size of the input data, as they both use the same input corpus. TFP10 and TDP09 models make use, however, of syntactic information, input representation which is richer than the bag-of-words data used both in our experiments as well as in EP10 and ML08. EP08 is also a syntactic method, however trained on the smaller BNC corpus. Cont_{LDA} is at disadvantage both due to the amount of input corpus data used, as well as with respect to its simple bag-of-words input representation.

We report the results in Table 6.5. In this table all the methods use identical test data, that of TFP10, with the exception of EP10.

	Random score	System score
EP10*	28.5	38.6
SVS	29.9	41.34
ML08-1	∼	42.17
ML08-2	∼	44.89
TFP10*	∼	44.39
Cont_{NMF}	∼	**46.49**
$\text{Cont}_{\text{LDA}_{\text{MIX}}}$	∼	43.58

Table 6.5: Results on the LST data. (* reported)

As it can be observed the score of the context-ignoring SVS baseline is very similar to the one obtained in the previous experiment and outperforms the EP10 score. ML08-1 and ML08-2 outperform this SVS baseline and perform significantly better than in the previous experiment. This can indicate that the use of syntactic neighbors as context features, rather than words within a surrounding window of the target, has a positive impact on performance.

The TFP10 method outperforms ML08-1 and performs similarly to ML08-2. Cont_{NMF} outperforms all of these while Cont_{LDA} scores similarly to ML08 and TFP10, despite the smaller amount of data it is trained on.

We further break down the results by individual parts of speech. These results are given in Table 6.6.

Model	Adv	Adj	Noun	Verb
EP08*	-	-	-	27.4
EP10*	-	-	-	39.9
TDP09*	-	-	-	36.54
ML08-1	47.89	42.21	44.17	36.85
ML08-2	52.06	44.24	46.89	39.31
TFP10*	48.19	39.41	46.38	**45.17**
Cont_{NMF}	**54.30**	**46.89**	**47.69**	40.55
$\text{Cont}_{\text{LDA}_{\text{MIX}}}$	51.29	42.43	44.80	39.02

Table 6.6: Results on the LST data, broken down by part of speech. (* reported)

The table also reports the results obtained by methods of EP08, EP10, and TDP09, which have been tested solely on verb data. These are outperformed both by the methods we have implemented in this experiment (proposed methods and baselines) as well as by the TFP10 system.

The results also highlight qualitative differences between our methods and the method of TFP10. TFP10 is better than all the other systems on verbs, scoring 5% better than Cont_{NMF}. However for all other parts of speech, Cont_{NMF} outperforms TFP10 by a large margin. A similar behavior can be observed for ML08-1 and ML08-2, which, similarly to our method, do not make use of syntax. This result seems to indicate that operating on a syntactic level, which is specific to TFP10, is beneficial for the treatment of verb data, and perhaps not particularly useful for other parts of speech.

6.3 Discussion

Finally, we also examined how the context words influence the class distributions of target words using examples from the lexical substitution dataset and the output of an individual LDA model. In many cases, a target word starts with a distribution spread over a larger number of senses, while a context word shifts this distribution to one majority sense. Consider, for instance, the target noun *jam* in the following sentence:

(3) With their transcendent, improvisational jams and Mayan-inspired sense of a higher, metaphysical purpose, the band's music delivers a spiritual sustenance that has earned them a very devoted core following.

	Probability	Sense
jam	0.18	TRAFFIC
	0.04	MUSIC
	0.04	FAN
	0.04	VEHICLE
(jam, band)	0.88	MUSIC
	0.08	FAN
(jam, traffic)	0.73	TRAFFIC
	0.06	VEHICLE
(jam, fruit)	0.92	RECIPE

Table 6.7: Class distributions for word *jam* in different contexts.

Classes	Word Distributions
TRAFFIC	*road, traffic, highway, route, bridge*
MUSIC	*music, song, rock, band, dance, play*
FAN	*crowd, fan, people, wave, cheer, street*
VEHICLE	*car, truck, bus, train, driver, vehicle*
RECIPE	*1, cup, 2, add, salt, fresh, sugar*

Table 6.8: Word distributions of latent classes.

Table 6.7 shows the latent classes activated for *jam*, while Table 6.8 lists the five most likely words associated to them[7].

As it can be seen, initially two traffic-related and two music-related senses are activated, however with low probabilities. In the presence of the context word *band*, we obtain a much more "focused" distribution, in which the MUSIC sense has 0.88 probability. The system ranks *riff* and *gig* as the most likely two substitutes for *jam*. The gold annotation also lists *session* as a possible substitute. When given a context word such as *traffic*, the most probable class is that of TRAFFIC followed by VEHICLE. Finally, in the context of *fruit* the most predominant sense is the one indicated by a RECIPE class.

In a large number of cases, the target is only partially disambiguated by a context word and this is also reflected in the resulting distribution. An example is the word *bug*, which initially has a distribution triggering the SOFTWARE (0.09, *computer, software, microsoft, windows*) and DISEASE (0.06, *disease, aids, virus, cause*) senses. In the context of *client*, *bug* remains ambiguous between the senses SECRET-AGENCY (0.34, *agent, secret, intelligence, FBI*) and SOFTWARE (0.29):

(4) We wanted to give our client more than just a list of bugs and an invoice — we wanted to provide an audit trail of our work along with meaningful productivity metrics.

[7]Class names are again assigned in an attempt to best describe their associated word distributions.

There are also cases in which the contextualized distributions are not correct, especially when senses are domain specific. An example is the word *function* occurring in its mathematical sense with the context word *distribution*. However, the classes that are triggered by this pair all relate to the "service" sense of *function*. This is a consequence of the newspaper corpus we use, in which the mathematical sense of *function* is rare. We also see several cases where the target word and one of the context words are assigned senses that are locally correct, but invalid in the larger context such as in the following example:

(5) Check the shoulders so it hangs well, stops at hips or below, and make sure the pants are long enough.

The pair *(check, shoulder)* triggers classes INJURY (0.81, *injury, left, knee, shoulder*) and BALL-SPORTS (0.10, *ball, shot, hit, throw*). However, the sentential context ascribes a meaning that is neither related to injury nor sports. This suggests that our models could benefit from more principled context feature aggregation.

Generally, verbs are not as good context words as nouns. To give an example, we often encounter the pair *(let, know)*, used in the common "to inform" meaning. The classes we obtain for this pair, are, however, rather uninformative general verb classes: {*see, know, think, do*} (0.57) and {*go, say, do, can*} (0.20). An interesting question is if this type of error can be eliminated in a space designed to only use context features which are relevant for the meaning of target words, for example by pruning context features based on pmi values.

6.4 Summary

In this section we have tested two instantiations of the framework (using NMF and LDA for inducing the latent structure) for computing lexical similarity in context on the task of lexical substitution.

Throughout this section, we have shown that a simple vector space model ignoring context, as well as the straightforward component-wise operations proposed by Mitchell and Lapata, are very strong baselines when instantiated with appropriate weighting schemes and similarity measures. We have shown, however, that in identical, completely comparable experimental settings, our methods outperform these baselines. Furthermore, we also outperform the best reported results on this data set, obtained by Thater et al. [2010], model which uses richer, syntactic information both in input representations as well as in guiding the vector compositions.

A number of future work directions are possible. Conceptually, we have defined our model in an asymmetric fashion, i.e., by stipulating a difference between target words and contextual features. However, in practice, we have used vector representations that do not distinguish the two: target words and contextual features are both words. This choice was made to facilitate comparisons with the popular bag-of-words vector space models. However, differentiating target from context representations may be beneficial particularly when the similarity computations are embedded within specific tasks such as the acquisition of paraphrases, the recognition of entailment relations or thesaurus construction.

Also note that our model currently contextualizes target words with respect to individual contexts. Ideally, we would like to compute the collective influence of several context words on the target. We plan to further investigate how to select or to better aggregate the entire set of features extracted from a context.

Finally, we would also like to investigate the use of more advanced models/algorithms for the stage of latent structure induction.

Chapter 7

Contextual Preferences for Inference Rules

This chapter tests the framework we have proposed on the task of assessing the contextual appropriateness of vector space model (VSM)-based paraphrases.

More precisely we operate on the representations introduced by Lin and Pantel [2001b], who propose one of the most accurate fully distributional paraphrasing methods. However, as for most distributional methods, it has been observed (Lin and Pantel [2001a]) that meaning ambiguity is an issue. Paraphrases extracted this way, are, typically, correct only in *some* of the contexts the may occur in. Following this observation, a number of subsequent studies have focused on assessing the correctness of a paraphrase rule given a context. This problem can be reduced to that of computing context-sensitive similarities in the vector space model of this specific paraphrasing method. In this chapter we show that the framework we have proposed can be used to accurately assess the context-appropriateness of paraphrase rules.

We overview the problem in Section 7.1. Section 7.2 describes previous work on this task while Section 7.3 shows how it can be approached within our framework. Section 7.4 provides an experimental evaluation and Section 7.5 discusses directions for future work.

7.1 Overview

A long line of research has studied methods for automatic paraphrasing, in which the focus is not on assessing similarity of words, but of larger, more informative units of text. Paraphrases are pairs of phrases which convey similar meaning, and can therefore substitute each other without changing the meaning of the sentences they occur in.

The purpose of paraphrase induction is that of addressing the challenge posed to most NLP applications by natural language variability: in natural language, the same meaning can expressed through a variety of different surface realizations. Question answering is one of the target applications of paraphrasing as it is often the case that questions and answer-containing sentences express the same meaning in different ways. Consider for example the following question-answer sentence pair.

(1) Who is Tom Cruise married to?

(2) Actor Tom Cruise and his wife Nicole Kidman accepted "substantial" libel damages on Thursday from a British newspaper.

The sentence in (2) contains the answer to the question in (1), however this requires the knowledge that *X's wife, Y* entails *X is married to Y*.

The intuitive advantage of using larger phrases over simple word similarity for the variability issue is straightforward: in many cases the similarity of phrases cannot be determined based on the meaning of the individual composing words. For example, it is not evident what approach can be used in order to learn that *X pay attention to Y* $\approx$ *X attach importance Y*, based solely on the composing words. However, as we will further detail in this chapter, when two these phrases are treated as *units*, there is enough distributional evidence to asses them as similar in meaning.

A number of methods for paraphrasing have been proposed in the literature, many of them relying on the use of comparable corpora (Barzilay and McKeown [2001], Pang et al. [2003], Sekine [2005]). Comparable corpora may consist of two translations of the same text into the same language, or news articles from different sources, which cover the same events. Comparable corpora is not widely available in all languages or in all domains, restricting thus the use of such methods.

The DIRT algorithm (Lin and Pantel [2001a]) introduces a method for extracting inference rules based on distributional representations, which makes use solely of large amounts of monolingual text. The terminology used by the authors is that of *inference rules.* As opposed to a paraphrase, an inference rule consists of a pair of phrases for which meaning entailment holds in *at least one* direction and in *some* contexts, not necessarily all. This definition relaxes the notion of paraphrase, and, as the authors argue, in many applications such rules are just as needed as tighter paraphrasing rules[1].

In the DIRT algorithm, a phrase is a binary relation (with two noun arguments), extracted as a path in a dependency graph. Such a path is usually called a

[1]Throughout this thesis we will use the terms inference rules/paraphrases interchangeably to refer to these type of rules. A distinction between the two terms is, however, not intended, as it is not relevant in the context of our work.

pattern. An inference rule consists of two such patterns for which a meaning entailment relation holds. The entailment may be bidirectional (i.e. a paraphrase) such as in: $X \xleftarrow{subj} resolve \xrightarrow{obj} Y \approx X \xleftarrow{subj} solve \xrightarrow{obj} issue \xrightarrow{nn} Y$. The meaning entailment may also hold in only one direction, as in: $X \xleftarrow{subj} win \xrightarrow{obj} Y \Rightarrow X \xleftarrow{subj} play \xrightarrow{obj} Y$.

The core of the method is a vector space model for computing similarity between two patterns. In this vector space model, each pattern is represented by the co-occurrence with left hand side (X) and right hand side (Y) noun fillers in a large corpus. Two patterns are compared in the X-filler space, and correspondingly in the Y-filler space by using the Lin similarity measure, preceded by pmi weighting. The final similarity score between two patterns is obtained by multiplying the X and Y similarity scores. Further on, this similarity is used for extracting paraphrases. A large collection of patterns is extracted from a corpus and each of these patterns can be paraphrased by returning its top most similar patterns, according to the similarity score.

The DIRT algorithm is relatively accurate for a mostly unsupervised method[2], as its accuracy is estimated to lie around 50% for the most confident paraphrases (Szpektor et al. [2007]). However, as for most distributional methods, it has been noted Lin and Pantel [2001a] that meaning ambiguity is an issue.

Consider the following DIRT paraphrase $X \xleftarrow{subj} shed \xrightarrow{obj} Y \approx X \xleftarrow{subj} close \xrightarrow{obj} lower \xrightarrow{nn} Y$[3]. Although unintuitive at first sight, this paraphrase is very correct in a financial context, such as the one in (3):

(3) At the NYSE closing bell on the New York Stock Exchange, here is how the major world indices and major U.S. stock indices ended the trading session on the world markets as well as the emerging markets including the stock market closing bell price: DOW (Dow Jones Industrial Average) **shed** 348.63 points.

In this context *DOW closed 348.63 points lower* is a paraphrase of *DOW shed 348.63 points.* The phrase $X \xleftarrow{subj} shed \xrightarrow{obj} Y$ is, however, ambiguous in meaning and in the following occurrence the paraphrase is inadequate:

(4) Infected cats shed a lot of virus particles into the environment.

It has been suggested that additional clauses, stating semantic constraints on the X and Y fillers, should be appended to the inference rules in order to solve this issue. This way, an instantiation of an inference rule (i.e. a rule in the

[2]With the exception of the parser used, which is trained in a supervised fashion, the method can be considered unsupervised.

[3]From the demo available at *http://demo.patrickpantel.com/demos/lexsem/paraphrase.htm*

context of specific X and Y noun fillers), can be judged as correct or not based on the attached semantic constraints. More precisely, in the above example, a semantic class such as *Market Index* could represent a constraint on the X filler. The *DOW* instantiation of the X filler matches this class and therefore the paraphrase is correct in this context.

A number of specific methods have been proposed for making DIRT rules context-sensitive by building and attaching such semantic classes. Section 7.2 summarizes these.

7.2 Related work

Following the original suggestion of Lin and Pantel [2001a], Pantel et al. [2007] propose a method to build and attach semantic classes to the X and Y filler slots of an inference rule, indicating the contexts in which the rule holds. They formally define the task to be solved as:

Task definition: *Given an inference rule* $p_i \rightarrow p_j$ *and the instance* $< w_X, p_i, w_Y >$ *determine if* $< w_X, p_j, w_Y >$ *is valid.*

Variations of the method proposed by Pantel et al. [2007] have been investigated in Basili et al. [2007] and Szpektor et al. [2008]. Although different in a number of aspects, all these methods propose the same basic architecture, which can be summarized as follows:

1. Acquire a paraphrase collection using the DIRT algorithm.
2. Learn semantic classes for the X and Y filler slots by grouping together semantically similar nouns.
3. For each rule in the paraphrase collection, attach semantic classes to the X and Y filler slots. These are chosen based on the filler nouns common to *both* patterns. For example, when given an inference rule such as *X is charged by Y* $\approx$ *Y announced the arrest of X*, semantic classes attached to the rule may be: *X: Person, Y: Law Enforcement Agent.*
4. Given an inference rule and a context (i.e. X and Y instantiations), decide if the rule holds in this context. The degree to which X and Y belong to the attached semantic classes is used as an indicator of the rule's correctness in this context. For example, the previous rule is correct for the context: *X: Terry Nichols, Y: prosecutors* because *Terry Nichols* belongs to the *Person* class and *prosecutors* to *Law Enforcement Agent.*

A number of confidence scores are estimated during each of these stages, such as the similarity of the original DIRT rule, the confidence of the attached semantic classes or the degree to which a noun belongs to such a class. A final score is computed as an aggregation of these scores.

The methods differ mostly with respect to the step of building semantic classes as well as w.r.t. the computation of confidence score and their aggregation. For example in Pantel et al. [2007], the noun semantic classes are built using WordNet in one case and the CBC clustering algorithm in another. In Basili et al. [2007], the X and Y fillers are clustered for each rule individually using a LSA-vector representation extracted from a large corpus. Both of these methods are used in the framework proposed by Szpektor et al. [2008].

Connor and Roth [2007] take a slightly different approach, as they attempt to classify the context of a rule as correct or not, without building or attaching explicit semantic classes to a set of inference rules.

Their method is however conceptually similar to the other proposals. In more detail, given a sentence S, a pattern v in this sentence and a second pattern u, the task is to decide whether u is a good paraphrase for v in sentence S. They decompose this problem in two stages: 1) decide if u and v are paraphrases and 2) decide if the overlap of contexts that u and v appear in is similar to the current context. Similarly to Pantel et al. [2007], the decisions are made based on statistics measuring overlap of context features, the difference lying in the use of a larger set of features, not just the filler nouns. Unlike related work, they build a classifier for any triple (S, u, v) rather than building a database containing paraphrases and attached semantic classes.

These methods are evaluated against a gold standard annotated by human participants. A number of inference rules are randomly selected from the output of the DIRT algorithm. Specific instances of these rules are selected form corpora and judges are asked to asses weather the rule holds or not for each of these.

All methods show improvement over DIRT. On a common data set, Pantel et al. [2007] and Basili et al. [2007] achieve statistically significant improvements over DIRT (at 95% confidence level) when employing clustering methods to learn semantic classes. Szpektor et al. [2008] propose a general framework for these methods and show that some settings lead to significant (level 0.01) improvements over DIRT, on data derived from the ACE 2005 event detection task.

We list examples of the attached semantic classes obtained with the Pantel et al. [2007] method (short ISP[4]) in Table 7.1. We consider two highly ranked paraphrases of $X \xleftarrow{subj} shed \xrightarrow{obj} Y$, namely $X \xleftarrow{subj} fall \xrightarrow{obj} Y$ and $X \xleftarrow{subj} close \xrightarrow{obj} lower \xrightarrow{nn} Y$.

We list the semantic classes attached to the X and Y fillers of these two paraphrases, in the order in which they are returned by the system. In the ISP system, a semantic class is defined as a set of words that belong to it. We mark classes which can be considered correct in bold fonts.

[4]Available as demo at *http://demo.patrickpantel.com/demos/lexsem/paraphrase.htm*

$X \xleftarrow{subj} shed \xrightarrow{obj} Y \approx X \xleftarrow{subj} fall \xrightarrow{obj} Y$

X	Y
{consumer idx,retail sale,prod. price}	{shirt,jacket,pant}
{revenue,profit,income}	{wake,spite,aftermath}
{unit,subsidiary,division}	{woman,child,man}
{Times Index,Nasdaq Idx,Dow Jones}	{revenue,profit,income}
{high,low,total}	**{about x percent,perc.,about %}**

$X \xleftarrow{subj} shed \xrightarrow{obj} Y \approx X \xleftarrow{subj} close \xrightarrow{obj} lower \xrightarrow{nn} Y$

X	Y
{Times Index,Nasdaq Idx,Dow Jones}	**{five cents,three cents,six cents}**
{consumer idx,retail sale,prod. price}	**{about percent,percent,about %}**
{unit, subsidiary, division}	**{more than x perc.,about four perc.,less than one perc.}**
{Rio Tinto,Bank of Austr.,Austr. Bank}	
{revenue,profit,income}	

Table 7.1: Output of the ISP system: two paraphrases for $X \xleftarrow{subj} shed \xrightarrow{obj} Y$ and the semantic classes attached to them.

As discussed in the beginning of this section, all these previous methods reduce the problem of determining the context-appropriateness of an inference rule to the task of assigning a *context-sensitive* similarity score to the two phrases. Following this observation, it becomes clear that our framework for context-sensitive similarity computations can also be used for this task. Previous work divides the context-sensitive similarity problem in different sub-components, such as defining semantic classes, or attaching these classes to inference rules. As opposed to this, our method provides an unitary approach to this, as it implements a motivated, probabilistic, framework targeted specifically to the goal of measuring similarity in context.

An important observation to be made here is that our method uses solely the input data used by the original DIRT algorithm. All of the previous methods for contextualizing DIRT rules use extra resources such as WordNet, or cluster *additional*, noun-specific distributional data. Thus, another question which is implicitly addressed by this work is if structurally richer information (i.e. paraphrasing *in context*) can be extracted from the same type of input data as the original DIRT method.

Section 7.3 shows how the context-sensitive similarity framework we have proposed can be used for assessing the context-appropriateness of an inference rule while Section 7.4 describes the experimental evaluation. Sections 7.5 and 7.6 discuss directions for future work and summarize the chapter.

7.3 Context-sensitive similarity for inference rules

The task of computing context sensitive similarity between phrases can be defined as follows:

***Task definition**: Given a pair of patterns p_i, p_j, and an instance $< w_X, p_i, w_Y >$ assign a similarity score indicating the meaning relatedness of p_i and p_j in context w_X, w_Y.*

This similarity score can be used to solve the original task of Pantel et al. [2007], that of determining the context-appropriateness of a rule as a binary true/false decision, by imposing a confidence threshold.

The key observation we start from (as discussed in Section 7.1), is that the DIRT algorithm is based on a vector space model for computing phrase similarity. The method we have proposed in Chapter 3 can simply be instantiated on the specific input data of this vector space; we obtain this way representations of patterns as distributions over a total set of latent meaning classes. Given a pattern and a context feature (X or Y filler), we can compute, as described in Chapter 3, posterior distributions which are in turn used to compare isolated patterns or patterns occurring in context. The rest of this section details on the main steps: extracting the input matrix, obtaining contextualized representations and computing similarity.

Input matrix The input data is identical to that of the DIRT algorithm, consisting of syntactic patterns represented by their co-occurrence frequency with left and right noun fillers. Table 7.2 shows a fragment of this input frequency matrix.

	country-X	*government-X*	*problem-Y*	*issue-Y*
(X *solve* Y)	89	82	1088	134
(X *settle* Y)	32	56	582	64

Table 7.2: Fragment of the DIRT-like input frequency matrix for acquiring paraphrases

The filler nouns as context features are made disjoint by adding a postfix to signal their role as X or Y fillers. In this example *country* occurs as a X-filler of pattern *X solve Y* 89 times, while *problem* occurs as a Y-filler 1088 times.

Vector meaning representations We use this input data to induce a set of K latent classes. Given a particular pattern p, its vector meaning representation is given by the likelihood of each inferred class: $P(z_k|p)$, with $k : 1..K$. We build contextualized representations for any given pattern p occurring with a context feature w using the posterior probabilities: $P(z_k|p, w)$, with $k : 1..K$.

Computing similarity The similarity between patterns occurring with two X and Y filler-words is obtained as proposed in Chapter 3, and as proposed by the DIRT algorithm as well, as the product of the similarities obtained when conditioning on the X and Y contexts individually.

We experiment with two ways of computing similarity. In a first method both patterns p_1 and p_2 are contextualized resulting in:

$$\begin{aligned} sim((w_X, p_1, w_Y), (w_X, p_2, w_Y)) =& sim(v(p_1, w_X), v(p_2, w_X))* \\ & sim(v(p_1, w_Y), v(p_2, w_Y)) \end{aligned}$$

where $v(p, w)$ is the contextualized vector representation of p occurring with word w[5].

Alternatively we only contextualize one of the two patterns, following the observation made in Section 6.1, that this can bring higher discriminative power to such methods, when used to differentiate between correct/incorrect paraphrases:

$$sim((w_X, p_1, w_Y)(w_X, p_2, w_Y)) = sim(v(p_1, w_X), v(p_2)) * sim(v(p_1, w_Y), v(p_2))$$

7.4 Evaluation

We test our method for context-sensitive similarity between syntactic patterns on two related tasks: 1) Given a pattern p_1 instantiated in a particular context c and p_2, a second pattern, assess the similarity between the two patterns, given c and 2) Given a pattern p, and a context c, generate the top-N most similar patterns in context c.

The first evaluation scenario is similar to that of the related work on contextual preferences for DIRT. In these evaluations, given an instantiated pattern p_1 and a second phrase, p_2, systems have to decide if the paraphrase holds. Unfortunately, since none of these evaluation data sets were made available to us, a comparison with these previous methods was not possible. To evaluate our proposal, we compare it against the original DIRT algorithm. For this purpose, we build an evaluation data set by adapting the data present in the lexical substitution task.

To our knowledge, none of the previous related work has approached the second, more difficult task, that of inducing context-sensitive paraphrases. This tasks consists of *generating* context-appropriate paraphrases rather than assessing if a given paraphrase is correct or not. Throughout this section we use our framework for generating context-sensitive paraphrases for a set of patterns extracted from question answering (QA) data. In this chapter, we exemplify the main issues we observe when performing paraphrase induction, while in Chapter 9 we induce paraphrases for question expansion in QA.

[5]We follow Lin and Pantel and, for each pattern occurring in the corpus, we also add its inverse, in which the X and Y fillers are interchanged, and the direction of the rule is reversed. This way, one can also identify similar patterns in which the X filler of one pattern matches the Y filler of the second pattern.

7.4.1 Experimental setup

Data We use the lexical substitution data (short LST, described in detail in Section 5.4.1) to build a set of instantiated patterns together with appropriate substitutes. We start by parsing the sentences using the Stanford dependency parser. We extract all dependency paths containing the target word from each LST sentence. An example of such path is $pound \xleftarrow{obj} shed \xrightarrow{subj} dog$ for the target word *shed* and the following sentential context:

(5) Feeding an Overweight Dog [offsite link] To help your overweight dog shed some pounds, you might need to change his eating habits - either what or how much he consumes.

In the next step, we use the **word** substitutes provided by the LST data to build **pattern** substitutes. An example is obtaining the pattern $dog \xleftarrow{subj} lose \xrightarrow{obj} pound$ as a substitute for the pattern $dog \xleftarrow{subj} shed \xrightarrow{obj} pound$. The confidence score assigned to it is given by the number of people that suggested *lose* as a good alternative for *shed*.

Pattern in context	Gold substitutes	Score
$virus \xleftarrow{obj} shed \xrightarrow{prep} to \xrightarrow{pobj} cat$	$\xleftarrow{obj} pass \xrightarrow{prep} to \xrightarrow{pobj}$	2
	$\xleftarrow{obj} give \xrightarrow{prep} to \xrightarrow{pobj}$	2
	$\xleftarrow{obj} transmit \xrightarrow{prep} to \xrightarrow{pobj}$	2
$pound \xleftarrow{subj} shed \xrightarrow{obj} dog$	$\xleftarrow{subj} lose \xrightarrow{obj}$	5
	$\xleftarrow{subj} relinquish \xrightarrow{obj}$	1
	$\xleftarrow{subj} discard \xrightarrow{obj}$	1

Table 7.3: Data instances obtained from the LST data.

Table 7.3 shows instances of target patterns in the obtained data, together with their correct substitutes. A system is presented with such a target pattern together with a total set of substitutes; this has been obtained from pooling together **all** the substitutes for that target word. The similarity scores returned by a system are used to rank this list, ideally with the correct substitutes being ranked at the top.

The particular syntax-based representation that the DIRT method uses is best suited for learning verbal paraphrases, i.e. patterns which are verb-rooted (Lin and Pantel [2001a]). For this reason we only use the verb subset of the LST data.

Models We test our method using LDA for latent class induction against the DIRT algorithm baseline, both using the same input frequency matrix.

The input frequency matrix is extracted from the XIE GigaWord fragment containing approximately 100 million tokens[6]. We parse the text with the Stanford dependency parser to obtain dependency graphs from which we extract patterns together with counts of their left and right fillers. We extract paths containing at most four words, including the two noun anchors. Furthermore we impose a frequency threshold on patterns and words, leading to a collection of a total of ≈80K distinct paths, with filler nouns ranging over a vocabulary of ≈40K words.

We use the LDA model to estimate latent senses using the Gibbs sampling algorithm. As in the previous chapter we set $\alpha = \frac{50}{K}$ and $\beta = 0.01$. We test a set of 5 K values: $\{800, 1000, 1200, 1400, 1600\}$. These are chosen to be large since they represent the global set of meanings shared by all the patterns in the collection. The DIRT method is implemented following the description in Lin and Pantel [2001b].

7.4.2 Results

We start by investigating the effect of parameter K on the performance of the models. Figure 7.1 plots the Kendall τ_b score obtained with each of the five K values. The similarity measure used is scalar product. Similarly to the other experimental evaluations in Chapters 5 and 6, we also build a mixture model which averages the similarity scores returned by each individual K setting.

As it can be observed, the individual LDA models outperform DIRT for all K values. As suggested by the previous experiments in Chapter 6, the mixture model outperforms all of the individual models. This is an advantage, since tuning the parameter K becomes unnecessary.

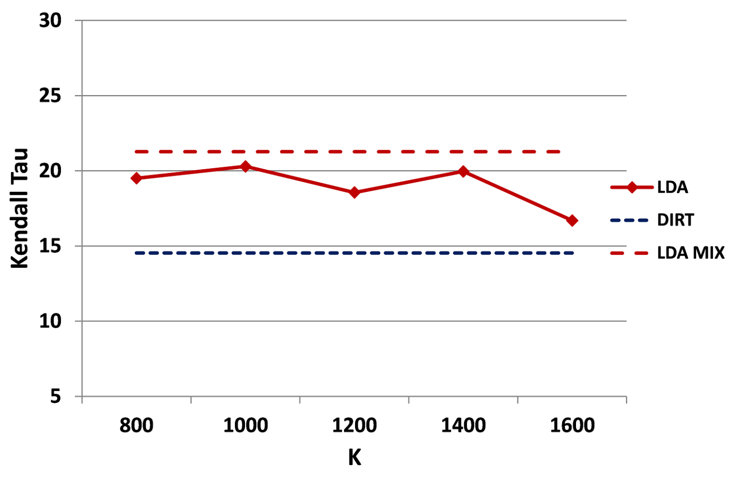


Figure 7.1: LDA and LDA-MIX (scalar product similarity) vs. DIRT

[6]We scale all co-occurrence counts by a factor of 3 in order to speed up computations.

In figures 7.2 and 7.3 we plot the same models, this time using cosine and inverse Jensen-Shannon (JS) as similarity measures. Cosine performs similarly to scalar product, while in the case of JS we notice a significant drop in performance with the $K = 1600$ setting performing slightly worse than DIRT. However, the mixture model still outperforms DIRT.

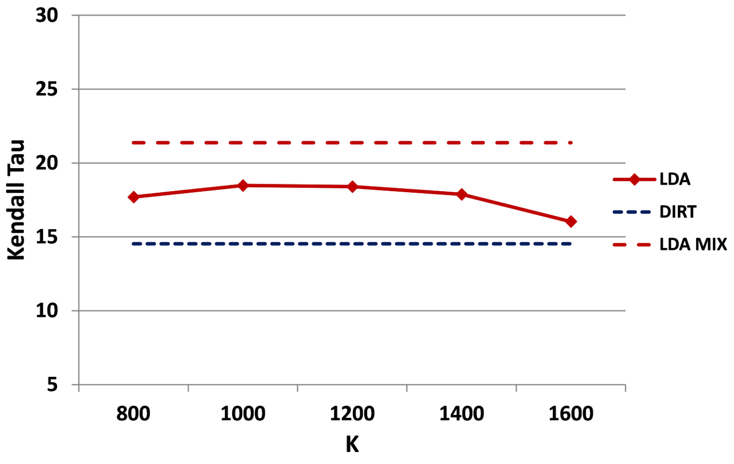


Figure 7.2: LDA and LDA-MIX (cosine similarity) vs. DIRT

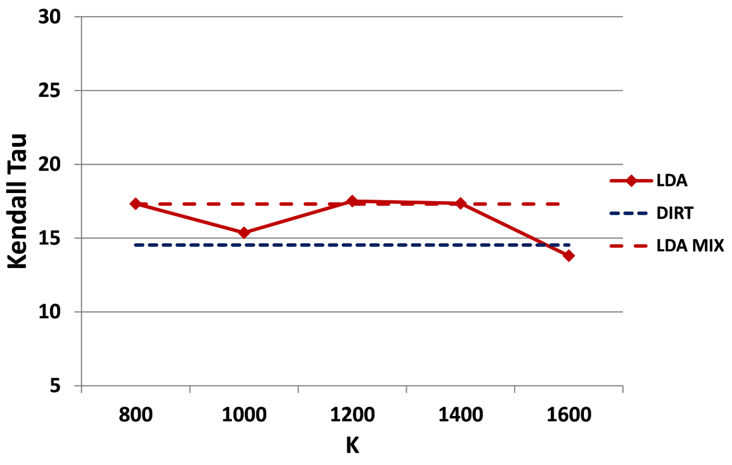


Figure 7.3: LDA and LDA-MIX (JS similarity) vs. DIRT

The results of the LDA MIX model using scalar product, both in Kendall τ_b and in GAP evaluation metrics, are given in Table 7.4. We also test an LDA model ignoring context which scores in the $[11 - 14]$ τ_b interval, depending on the similarity measure used. This scores lower than DIRT (14.5 τ_b), indicating that DIRT is indeed a good method for computing (isolated) pattern similarity. We perform significance testing using randomized shuffling as described in Chapter 5. The LDA methods using context outperform DIRT at significance level $p < 0.005$. Using scalar product as similarity outperforms cosine, which in turn outperforms JS divergence.

Model	τ_b	GAP
Random	0.0	34.91
DIRT	14.53	48.06
LDA$_{\mathbf{sp}}$	**21.27**	**52.37**
LDA$_{\mathbf{cos}}$	**21.38**	**51.12**
LDA$_{\mathbf{JS}}$	**17.31**	**50.06**

Table 7.4: Results on Lexical Substitution data

In Table 7.5 we list the rankings returned for three occurrences of the pattern $X \xleftarrow{subj} shed \xrightarrow{obj} Y$. The contexts considered are (you, blood) extracted from sentence (6), (dog, pound) (sentence (5)) and (study, light) (sentence (7)), each of them illustrating a different sense of the verb *shed.* The gold substitutes are highlighted in bold and the confidence score is given in parentheses.

(6) You have shed blood for us and we thank you .

(7) A mouse study sheds light on the mixed results coming from investigations into the cognitive effects of hormone replacement therapy.

The DIRT method is context-insensitive and therefore returns the same rank for all instances (first column of the Table 7.5). The context-sensitive methods allow us to obtain more informative, instance-specific, rankings. For each of these contexts, the rankings differ to a great extent and favor the context-appropriate substitutes, such as *lose* for *dog shed pound* or *reveal* for *study shed light.* It is interesting to notice that *shed light* is ambiguous as it can also refer to *radiating light*; although *study* dismisses this meaning, it is still reflected in the ranking obtained, as substitutes *give* and *emit* rank second and third.

7.5 Towards context-sensitive paraphrase induction

The DIRT algorithm gives a method for computing similarity between phrases represented as patterns extracted from dependency parses. As discussed in 7.1, similarity computations using DIRT are accurate enough to allow paraphrase induction: given a target pattern, an *entire* collection of patterns is ranked based on the similarity with the target phrase. The highest ranked patterns are selected as paraphrases of the original phrase. An example of this is given in Table 7.6, which lists the top most confident paraphrases for the phrase $X \xleftarrow{subj} acquire \xrightarrow{obj} Y$.

Although determining the context-appropriateness of an inference rule is a task that has been approached in the past, the question of performing context-sensitive paraphrase induction within the DIRT representation framework has

DIRT No context	Context1 (you, blood)	Context2 (dog, pound)	Context3 (study, light)
drop	**give(1)**	drop	**reveal(2)**
lose	throw	**lose(5)**	give
give	**lose(3)**	give	emit
reveal	discard	disperse	**throw(3)**
relinquish	reveal	emit	disperse
throw	transmit	throw	discard
transmit	spread	reveal	spread
pass	emit	**discard(1)**	transmit
emit	relinquish	**relinquish(1)**	relinquish
spread	drop	transmit	pass
discard	pass	spread	drop
disperse	disperse	pass	lose

Table 7.5: Ranked substitutes for the pattern $X \xleftarrow{subj} shed \xrightarrow{obj} Y$. Correct substitutes are marked in bold.

$X \xleftarrow{subj} acquire \xrightarrow{obj} Y$
$X \xleftarrow{subj} get \xrightarrow{obj} Y$
$X \xleftarrow{subj} purchase \xrightarrow{obj} Y$
$X \xleftarrow{subj} buy \xrightarrow{obj} Y$
$X \xleftarrow{subj} use \xrightarrow{obj} Y$
$X \xleftarrow{pobj} to \xleftarrow{prep} sell \xrightarrow{obj} Y$
$X \xleftarrow{pobj} by \xleftarrow{prep} own \xrightarrow{obj} Y$
$X \xleftarrow{obj} provide \xleftarrow{prep} with \xrightarrow{obj} Y$
$X \xleftarrow{pobj} to \xleftarrow{prep} provide \xrightarrow{obj} Y$

Table 7.6: DIRT paraphrases for $X \xleftarrow{subj} acquire \xrightarrow{obj} Y$

not been yet addressed. This task can be defined as follows:

Context-sensitive paraphrase generation: *Given a target pattern p_i and an instance $< w_X, p_i, w_Y >$, return patterns p_j such that p_i and p_j form an inference rule in context w_X, w_Y.*

To exemplify this consider the pattern $X \xleftarrow{subj} shed \xrightarrow{obj} Y$. In a newspaper domain this phrase has a strong bias towards the meaning of *to fall, to drop* as referring to market indexes. However, in contexts such as *shed tear*, *shed blood* or *shed light*, the phrase has completely different meanings. The ISP system of Pantel et al. [2007] identifies adequate semantic classes for *shed*'s most frequent (financial) sense (see Table 7.1); however, we observe that no paraphrase reflecting any of the other senses is obtained in the top 100 paraphrases.

This points to a property shared by all the methods developed to learn contextual preferences for DIRT rules: they are based, and therefore tuned, on the assumption that we are given a high-confidence DIRT rule. It is however not clear how the methods can be adapted in order to discover pairs of patterns which are in turn almost distributionally distinct in an out-of-context scenario.

The method we have proposed can be straightforwardly used for context sensitive paraphrase induction. In this section we perform a second experiment as we use it to paraphrase a set of patterns extracted from QA data. An evaluation for this task is in the purpose of future work and, at the moment, we only summarize a series of observations that can be made when manually inspecting the paraphrases generated. Chapter 9 uses this method for paraphrasing questions for an answer extraction module.

The task of context-sensitive paraphrasing can be expressed naturally in the framework we have proposed: given a pattern p_i in context w_X we return the top N patterns p_j which maximize $sim(vec(p_i, w_X), vec(p_j, w_X))$. For the cases in which we are given two context words w_X and w_Y, we maximize $sim(vec(p_i, w_X), vec(p_j, w_X)) * sim(vec(p_i, w_Y), vec(p_j, w_Y))$.

We use the LDA mixture model which we apply for paraphrasing a set of patterns extracted from questions found in TREC QA data. Only one context word is available for most patterns occurring in questions, as a second argument is often a un-informative question word (e.g. *who, which*). To this set of patterns we add the patterns encountered in the LST data set.

One of the main observations we make is that the paraphrases generated often convey appropriate, context-aware lexical variation, but are not substitutable in the provided context. Consider for example the pattern $X \xleftarrow{subj} appear \xrightarrow{prep} on \xrightarrow{pobj} Y$, extracted from the question:

(8) When did Led Zeppelin appear on BBC?

This is paraphrased by DIRT as in Table 7.7 and by the context sensitive method (JS and cosine similarities) as in Table 7.8.

The context-insensitive substitutes extracted with DIRT are very accurate, with commonly returned patterns such as *be on*, *see on* or *find on.* The LDA method returns *television*-related paraphrases such as *broadcast on bbc*, *broadcast by bbc*, *announced by bbc* or *tell bbc station.* However, as it can be observed, while the variation in lexical items is meaning-appropriate, there is a significant drop in accuracy compared to DIRT, as many of the generated patterns cannot be used as substitutes.

We observe another class of errors, in which the patterns returned are very clearly associated with the given context, however not exhibiting the same meaning as the original pattern. Consider the pattern $X \xleftarrow{subj} appear \xrightarrow{prep}$

$X \xleftarrow{subj} \mathit{appear} \xrightarrow{prep} \mathit{on} \xrightarrow{pobj} Y$
$X \xleftarrow{obj} \mathit{be} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$
$X \xleftarrow{obj} \mathit{appear} \xrightarrow{prep} \mathit{in} \xrightarrow{obj} Y$
$X \xleftarrow{obj} \mathit{release} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$
$X \xleftarrow{obj} \mathit{see} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$
$X \xleftarrow{subjpass} \mathit{be} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$
$X \xleftarrow{obj} \mathit{go} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$
$X \xleftarrow{subjpass} \mathit{find} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$
$X \xleftarrow{obj} \mathit{come} \xrightarrow{prep} \mathit{on} \xrightarrow{obj} Y$

Table 7.7: DIRT paraphrases for $X \xleftarrow{subj} \mathit{appear} \xleftarrow{prep} \mathit{on} \xrightarrow{pobj} Y$

zeppelin, $X \xleftarrow{subj} \mathit{appear} \xrightarrow{prep} \mathit{on} \xrightarrow{pobj} Y$, **bbc**	
LDA - inverse JS	LDA - cosine
$X \xleftarrow{subjpass} \mathit{broadcast} \xrightarrow{prep} \mathit{by} \xrightarrow{pobj} Y$	$X \xleftarrow{pobj} \mathit{in} \xleftarrow{prep} \mathit{channel} \xrightarrow{nn} Y$
$X \xleftarrow{partmod} \mathit{broadcast} \xrightarrow{prep} \mathit{by} \xrightarrow{pobj} Y$	$X \xleftarrow{poss} \mathit{tv} \xrightarrow{nn} Y$
$X \xleftarrow{subj} \mathit{quote} \xrightarrow{obj} \mathit{international} \xrightarrow{nn} Y$	$X \xleftarrow{subjpass} \mathit{announce} \xrightarrow{prep} \mathit{by} \xrightarrow{pobj} Y$
$X \xleftarrow{subj} \mathit{tell} \xrightarrow{obj} \mathit{station} \xrightarrow{nn} Y$	$X \xleftarrow{aposs} \mathit{tv} \xrightarrow{nn} Y$
$X \xleftarrow{subj} \mathit{tell} \xrightarrow{obj} \mathit{program} \xrightarrow{nn} Y$	$X \xrightarrow{prep} \mathit{for} \xrightarrow{pobj} \mathit{television} \xrightarrow{nn} Y$
$X \xleftarrow{partmod} \mathit{broadcast} \xrightarrow{prep} \mathit{on} \xrightarrow{pobj} Y$	$X \xleftarrow{partmod} \mathit{broadcast} \xrightarrow{prep} \mathit{by} \xrightarrow{pobj} Y$
$X \xleftarrow{pobj} \mathit{of} \xleftarrow{prep} \mathit{voice} \xrightarrow{nn} Y$	$X \xleftarrow{partmod} \mathit{broadcast} \xrightarrow{prep} \mathit{on} \xrightarrow{pobj} Y$
$X \xleftarrow{pobj} \mathit{in} \xleftarrow{prep} \mathit{channel} \xrightarrow{nn} Y$	$X \xrightarrow{prep} \mathit{of} \xrightarrow{pobj} \mathit{tv} \xrightarrow{nn} Y$

Table 7.8: Context-sensitive paraphrases for $X \xleftarrow{subj} \mathit{appear} \xleftarrow{prep} \mathit{on} \xrightarrow{pobj} Y$

on $\xrightarrow{pobj} Y$ this time occurring in context *feminist-X, dollar-Y* in the following question:

(9) What American feminist appeared on a silver dollar?

The top paraphrases are dominated by the *dollar* context such as *amount in dollar* or *side of dollar*, none of which having the same meaning as *appear on.* Another example is the phrase *shed blood*, for which highest ranked patterns are expressions which often occur with *blood.* These are *donate blood*, *vessel of blood* or *blood test*, none of which are however similar in meaning with the phrase *shed blood.*

Finally, we also notice that the method is not robust to parsing errors. For example, the path *you* $\xleftarrow{subj}$ *fly* $\xrightarrow{obj}$ *pregnancy* is obtained from an incorrect parse of a QA question. When paraphrasing this, the model returns with average confidence scores, phrases such as *collect from pregnancy*, *receive in* *pregnancy* or *receive for pregnancy.* In the future we plan to investigate the use of our model for detecting such erroneous parse paths. One way of approaching this is by using selectional preferences which can be easily induced from our framework; these can indicate highly unlikely contexts as being most probably generated by parsing errors.

Another aspect to be investigated concerns the amount of input corpus data used by these methods. Previous work shows that the performance of distributional methods can increase significantly with the size of input data. Our method in particular might benefit from this, as we attempt to learn a more complex model than traditional vector space methods. However, it is still an open question if it is possible to obtain an accurate model for context-sensitive paraphrase generation which uses solely distributional information and the DIRT-like representations in particular.

7.6 Summary

Throughout this chapter we have instantiated our framework on the task of assessing the context-appropriateness of distributional paraphrases.

Although conceptually similar to previous work, our method provides a much more unitary approach in which the context-sensitive similarity of phrases is computed in a probabilistic setting. We have shown this to outperform DIRT, the original distributional paraphrasing algorithm, while using the same corpus input data.

Furthermore, the framework can be naturally applied to context-sensitive induction of paraphrases, task which has not been yet approached in previous work. The main observation made is that the lexical variation we obtain when performing paraphrase generation this way is meaning-appropriate, however a significant drop in accuracy from the original DIRT method is observed. In future work we plan to further adapt our framework to this task, with the goal of obtaining high-precision context-appropriate paraphrases. In Chapter 9 we test this preliminary version of the paraphrasing method against DIRT for performing question expansion in a simple question answering pipeline.

Chapter 8

Distributional Paraphrasing for Question Answering

The second part of the thesis investigates the use of distributional paraphrasing for the task of question answering. The current chapter provides the necessary background and proposes an answer extraction module centered around the use of paraphrases for generalizing the meaning representation of a question.

Chapter 9 evaluates this proposal. In particular, it focuses on using two distributional paraphrase methods: the DIRT method introduced by Lin and Pantel and the context-sensitive paraphrasing method developed within the framework proposed throughout this thesis.

8.1 Overview

Question Answering Question answering is an information retrieval task in which a system is provided with a question and the goal is to automatically retrieve an answer to this question by making use of large collections of texts. A typical example is the following question, from the 2002 TREC QA track (Voorhees [2002]):

(1) In what year did Joe DiMaggio compile his 56-game hitting streak?

Here a system has to identify *1941* as the correct answer string. In this particular TREC QA track, systems have to find the answer using a collection of over one million documents, out of which at least one document answers the question.

Most of the existing question answering systems implement three main stages. Initially, the question is analyzed and an expected answer type is determined. For example for a *when* question, candidate answers are strings which represent a date. Document retrieval and passage retrieval are also main components of question answering systems. Typically, in these stages, vector space models or language-model methods are used in order to compare queries with documents, and to shorter text passages, and retrieve those which may contain the answer to the question. The third stage is that of answer extraction. Given a question and a set of candidate passages, the goal of answer extraction is to return the exact string which is the answer to the question.

This work focuses on the answer extraction stage of question answering. More precisely, an answer extraction module is given a question and a set of candidate passages and the goal is to extract a string which is an answer to the question. For example, given the following question-sentence pair, the goal is to extract the correct answer string, "Nicole Kidman":

(2) Who is Tom Cruise married to?

(3) Actor Tom Cruise and his wife Nicole Kidman accepted "substantial" libel damages on Thursday from a British newspaper that reported he was gay and that their marriage was a sham to cover it up.

As opposed to other retrieval tasks, QA, and answer extraction in particular, require higher level of text understanding as similar meaning is often expressed with various surface realizations in questions and in sentences containing the answer. In the example above, the answer can be identified by a system which can infer that Tom Cruise's wife is in fact someone that Tom Cruise is married to. In order to extract answers, one needs to provide a mechanism that generalizes over the different ways in which the same concept can be expressed.

Overview of our approach We propose employing syntactic-level representations for questions and sentences and the use of distributional paraphrasing to provide such means of generalization.

More precisely, our initial focus is the DIRT algorithm, which uses distributional statistics to acquire paraphrases. Despite the fact that DIRT was developed with applications such as QA in mind (Lin and Pantel [2001a]), and even though it is a resource which is easy to acquire and relatively accurate, there has been rather little research on using it to solve the variability problem in NLP, in general, and in QA in particular.

Our second focus is the alternative use of context-sensitive paraphrasing for answer extraction. More precisely, we employ the context-sensitive paraphrasing method detailed in Chapter 7 which is developed in the line of the DIRT framework; this uses the same syntactic-level representation for phrases and the same corpus evidence. The focus is, this time, on investigating the use of paraphrases which are context-appropriate, following the intuition that the accuracy of a paraphrase is highly dependent on the context it is used in.

Our proposal is described and evaluated in Chapter 9. Throughout the current chapter we provide the necessary background and overview previous work. Paraphrases and other resources for question answering are described in Section 8.2. As the use of DIRT in QA is rather limited, in Section 8.3 we discuss previous studies and experiments performed on the related task of Recognizing Textual Entailment (RTE).

8.2 Knowledge resources for QA

Over the years, a number of hand-crafted resources have been employed in QA systems in order to generalize over different ways of expressing the same meaning.

One of the most widely-used resources for question answering is WordNet, which has been used for a variety of tasks in QA such as passage selection (Hovy et al. [2000]), query expansion (Greenwood [2004]) or expected answer typing (Pasca and Harabagiu [2001b], Pasca and Harabagiu [2001a]), to name just a few.

In more recent years, the use of deeper semantic representations has been proposed. For example, studies such as Narayanan and Harabagiu [2004] use PropBank (Palmer et al. [2005]) for identifying predicate-argument structure and FrameNet (Baker et al. [1998]) for frame assignment. Resources such as FrameNet or PropBank are costly to construct, they exist for a handful of languages and suffer furthermore from low coverage. For example, Shen and Lapata [2007] show that the benefits of using FrameNet in answer extraction are severely limited by the low coverage of this resource. Difficulties in using these resources become apparent also in the work of Kaisser and Webber [2007], which address the same issue of using semantic roles for QA. In their experiments, the use of PropBank, FrameNet and VerbNet (Kipper Schuler et al. [2009]) leads to performance gains only when using carefully constructed heuristics and only in specific cases.

A different line of research has focused on the use of automatic paraphrasing for QA. In contrast to hand-crafted semantic resources, many of these methods use minimal supervision and are based on the use of large quantities of text[1].

The focus of our work is the DIRT algorithm which has been developed as an alternative to the use of hand-crafted paraphrase resources in QA. The algorithm does not find just paraphrases (i.e. phrases that can substitute each other in most contexts) but also phrases that stand in a more loose semantic relation: *only one* may a substitute for the other, and this may hold only in *some* contexts. This leads to the authors' preference for using the *inference rule*

[1]See Kaisser [2009] for an overview of some of the previous work on paraphrasing for QA

terminology. The accuracy of the DIRT method is relatively high for a method requiring minimal supervision (supervision is needed for parser training). The method has been evaluated to ≈50% precision for the most confident expansions (Szpektor et al. [2007]).

Furthermore, a number of methods have been developed for improving the algorithm by learning the directionality of rules in Bhagat et al. [2007], or attaching contextual information to the rules (Szpektor et al. [2008], Pantel et al. [2007], Connor and Roth [2007]). Despite the attention that the DIRT method has received, there has been rather little research on using it to deal with the variability problem in QA. To our knowledge it has only been addressed by Poon and Domingos [2009], who report on using DIRT for question answering in a biomedical domain.

However, the DIRT resource has received more attention in the context of a related task, that of Recognizing Textual Entailment. Section 8.3 overviews relevant work in this context.

8.3 Preliminary experiments: DIRT paraphrasing for RTE

Recognizing Textual Entailment (RTE) has been proposed as a stand-alone task to address the issue of variability encountered in many NLP applications.

More precisely, the task has been formulated as that of detecting semantic inference between two fragments of text. The RTE (Dagan et al. [2006]) challenges provide benchmark data sets for this task. In the RTE setting, a system is provided with a text **T** and a hypothesis **H**, and the goal is to detect if **H** can be inferred from **T**. The RTE2 and RTE3 datasets are specifically built to mirror the need for semantic reasoning required by four information retrieval applications: Question Answering, Information Retrieval, Information Extraction and Summarization. For example, the QA subset is created by annotators in the following manner: Annotators are provided with questions and answer passages as returned by QA systems. Out of these passages, they select either a correct or an incorrect answer string, and this is “plugged in” into the question while transforming it into a declarative statement. The original passage is turned into text **T** while the transformed question serves as hypothesis **H**.

The study of Clark et al. [2007] focuses on the type of knowledge required for detecting inference in the RTE **T-H** pairs. This study brings some evidence to confirm the intuition that the type of knowledge encoded in the DIRT resource may be useful for RTE. The authors provide an insightful analysis based on 100 positive entailment pairs which attests that lexical substitution (e.g. synonyms, antonyms) or simple syntactic variation account for the entailment only in a small number of pairs. Thus, they conclude that one essential issue in the task of recognizing semantic inference is to identify more complex expressions which, in appropriate contexts, convey similar meanings.

Several RTE systems, such as those of Bar-Haim et al. [2007], Iftene and Balahur-Dobrescu [2007] or Clark et al. [2007], attempt to use DIRT, however with inconclusive results regarding the usefulness of this resource. Following this line of research, in Dinu and Wang [2009] we perform a series of experiments focusing on using DIRT inference rules for the RTE task. The remainder of this section details on our findings.

In our experiments, we use the DIRT collection provided by the authors which is extracted from 1GB of newswire text. In total, the collection consists of 200K distinct patterns; for each of these, we use the top 40 most confident paraphrases, following Lin and Pantel [2001a].

Initially we investigate how many DIRT-rule instantiations we encounter in the RTE data sets.

The following **T-H** pair is an example of this.

(4) **T:** The sale was made to pay Yukos US$ 27.5 billion tax bill, Yuganskneftegaz was originally sold for US$ 9.4 billion to a little known company **Baikalfinansgroup** which was later **bought by** the Russian state-owned oil company **Rosneft.**
H: Baikalfinansgroup was **sold to Rosneft.**

In this case, we encounter the rule *X bought by Y* $\approx$ *Y sold to X* with X and Y instantiated as *Baikalfinansgroup* and *Rosneft* respectively.

Such instantiations can be identified in only 2% of the RTE pairs. An analysis of these cases in the RTE2 development set shows that indeed, finding a DIRT rule instantiated this way is a very good predictor of meaning entailment, as 80% of these pairs are cases of positive entailment. This is even higher than the estimated accuracy of DIRT, most probably due to the bias of the RTE data sets, which are built such that 50% of the pairs are true entailment.

A closer look at the data reveals that a large number of DIRT rules can be identified as instantiated in a more flexible manner, such as in the following example:

(5) **T:** Libya's case against Britain and the US **concerns** the dispute over their demand for extradition of Libyans charged with blowing up a Pan Am jet over Lockerbie in 1988.
H: One case **involved** the extradition of Libyan suspects in the Pan Am Lockerbie bombing.

In this case, we encounter the rule *X concerns Y* $\approx$ *X involves Y* with X and Y fillers differently instantiated in **T** and **H**. The meaning entailment still holds because *one case* in **H** is implied by *Libya's case against Britain and the US* in **T** and in this context *the extradition of Libyan suspects in the Pan Am Lockerbie bombing* has the same meaning as *the dispute over their demand for extradition of...* Despite the fact that the use of such a resource must allow for flexibility, as shown by this example, most RTE systems, as well as the QA system of Poon and Domingos [2009], use DIRT paraphrase rules only when encountering a strict X and Y filler match.

Following this observation, we perform a second experiment which aims at testing the degree to which the DIRT rules are a predictor for textual entailment when applied in a more relaxed fashion.

We start by extracting matching dependency paths from **T-H** pairs. More precisely we first identify, if present, pairs of matching nouns in text and hypothesis (for example *case* and *extradition* in the above example, which can be found in both **T** and **H**). Following this we extract the dependency path between the two nouns in the text and the dependency path between the two nouns in the hypothesis. These form an example of *matching* dependency paths in text and hypothesis.

After extracting these, the next step is to identify if an inference rule can be applied on them. More precisely, we look for inference rules in which one pattern is a subpath of the text path and the second pattern is a subpath of the hypothesis path. This is a more relaxed application of an inference rule: there is no constraint on the X and Y fillers to be identical. However, we still control the context as we only allow them to be used in matching tree paths, which guarantee some degree of overlap between the two sentences. In the example above, the dependency paths extracted cover the fragment *case concerns the dispute over their demand for extradition* in **T** and the fragment *case involved the extradition* in **H**; the DIRT rule *X concerns Y* $\approx$ *X involves Y* is matched in these two dependency paths.

We select such **T-H** pairs in which we can find matching paths as well DIRT rules. We observe that these still show a strong bias towards positive entailment: $\approx$70% of these pairs are positive, indicating that the occurrences of DIRT rules in these pairs are still a strong indicator of meaning similarity between the two sentences.

We further relax the application of the inference rules even more, by allowing any word in such a rule to be replaced by a WordNet synonym. This obviously introduces a lot of noise as the meaning ambiguity of words will lead to a large number of incorrect inferences. Surprisingly, we observe that even in this scenario, there is a strong bias towards true entailment. The subset of **T-H** pairs extracted this way is considerably larger and we can still predict positive entailment with 65% accuracy on RTE2 and 72% accuracy on RTE3. These are high accuracy scores, as RTE is a difficult task: most of the submitted RTE2 systems scored 55%-61% in accuracy (Bar-Haim et al. [2006]) while most of the

RTE3 systems scored 59%-66% (Giampiccolo et al. [2007]).

These experiments suggest that the full potential of a paraphrase collection such as DIRT can only be realized in a flexible framework. The QA study of Poon and Domingos only uses a very small number of paraphrases, the top three most confident ones, and furthermore does not allow for flexibility: if a sentence contains one of these paraphrases and if the available argument slot in the question is *identical* to the corresponding argument slot in the sentence, the sentence is considered a match

In the following section we propose an answer extraction pipeline which integrates a paraphrase component. In line with the textual entailment observations made in this section (and also made by work such as Marsi et al. [2007]) we focus on flexibility: we allow paraphrases to *reduce* the differences in dependency structures between a question and a sentence containing the answer. This is in contrast to relying on them only when they provide an exact match between the question and the sentence.

8.4 Answer extraction pipeline

This section proposes an answer extraction module which integrates a syntactic-level paraphrasing component. The other components of the QA pipeline, such as passage retrieval, are not in the focus of our work and for this reason their discussion is postponed to the experimental evaluation in Chapter 9.

We build a simple answer extraction system, which uses syntactic representations of questions and sentences in order to find answers. The main components of this pipeline are:

1. Candidate answer extraction (Section 8.4.1). Given a question and a set of text passages, we develop a simple strategy for identifying *all* the text strings which may be the answers to a question.

2. Paraphrase-based question expansion (Section 8.4.2). This stage generalizes the meaning representation of a question by performing dependency-level paraphrasing.

3. Candidate answer ranking (Section 8.4.3). This is the last stage of our answer extraction module, in which the goal is to rank all the candidate answer strings. This is done based on the similarity between text passages and questions, and the string ranked the highest is returned as the answer to the question.

8.4.1 Candidate answer extraction

The input of an answer extraction module is a question together with a set of associated text passages, each typically consisting of one or a few sentences[2].

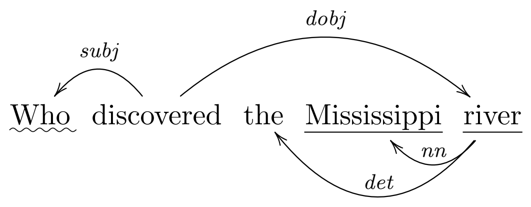
In this section we describe the first step in answer extraction which consists in identifying a set of candidate answer strings. At this stage the focus is on recall, i.e. on obtaining all possible answer strings; the final stage of candidate answer ranking will return a *single* such string as the answer.

We begin with the analysis of the question, which we parse using a dependency parser. Following this, we identify a *question word* and *key words* in the question parse tree. Question words are wh-words (*who, what, when, when, how, which*) and key words are nouns and cardinal numbers. We further extract the dependency paths between any pair consisting of a question word and a key word.

More precisely, given a dependency tree, a dependency path is the sequence of nodes and labeled edges between two nodes. Consider the following question-sentence pair.

(6) Q: Who discovered the Mississippi river?
S: The Mississippi river was discovered by Hernando de Soto in 1541.

For this question we obtain the following dependency parse:



We extract the following dependency paths, connecting the question word *who* to the key words *river* and *Mississippi*:

Who discovered the Mississippi river?
who $\xleftarrow{subj}$ discover $\xrightarrow{dobj}$ river $\xrightarrow{nn}$ Mississippi
who $\xleftarrow{subj}$ discover $\xrightarrow{dobj}$ river

Table 8.1: Question paths

[2]We operate on the assumption that at least one of these passages will contain the answer to the question, however this is highly dependent on the performance of the passage retrieval module.

Next, we move to the analysis of candidate sentences. We parse each candidate sentence and identify keywords that are present in the question. We extract all the paths between such key words and any other nouns or cardinal numbers present in the sentence. For the example above, the key words identified in the sentence are *river* and *Mississippi* and we obtain the sentence paths in Table 8.2.

The Mississippi river was discovered by Hernando de Soto in 1541.
Soto $\xleftarrow{pobj}$ by $\xleftarrow{prep}$ discover $\xrightarrow{dobj}$ river $\xrightarrow{nn}$ Mississippi
Soto $\xleftarrow{pobj}$ by $\xleftarrow{prep}$ discover $\xrightarrow{dobj}$ river
1541 $\xleftarrow{pobj}$ in $\xleftarrow{prep}$ discover $\xrightarrow{dobj}$ river $\xrightarrow{nn}$ Mississippi
1541 $\xleftarrow{pobj}$ in $\xleftarrow{prep}$ discover $\xrightarrow{dobj}$ river

Table 8.2: Sentence paths

These paths connect a question key word to other words in the sentence. We treat these words as *answer candidates.* In the current example, the answer candidates are *Soto* and *1541.*

This method will extract answer candidates that are single words, each corresponding to a node in the syntactic parse. In most of the cases, however, the correct answer is a string containing more than one word. In the example above, the answer is the entire string *Hernando de Soto.* For this reason, as the final step in candidate answer extraction we generate (longer) answer candidate strings from candidate words. We implement the following simple POS tag-based heuristic: if we find the fragment the_D $Spanish_{JJ}$ $explorer_{NN}$ $Hernando_{NNP}$ de_{NNP} $Soto_{NNP}$, then the following strings will be generated: *de Soto* and *Hernando de Soto.* This is because all the POS tags preceding the word form *Soto* are proper nouns or numbers. The total set of candidate answers will contain the original single word strings as well as the expanded strings.

8.4.2 Question expansion using paraphrases

In this step, we generalize the meaning representation of a question by making use of paraphrases.

The question and sentence meaning representations as sets of syntactic dependency paths, are not general enough: they encode particular choices of lexical items and syntactic constructions. We abstract away from these particulars by using paraphrases obtained solely from distributional evidence. This simple approach stands in contrast with representations which explicitly posit and use abstract dependencies such as those of Frame Semantics (Fillmore [1982], Baker et al. [1998]), as distributional paraphrasing requires only large quantities of text and a parser. In comparison, such paraphrase resources are much easier to build and the extraction method can be extended to languages other than English.

The intuition behind this approach is to include many realizations of the same meaning in the question representation, such that the chance that it will closely match a sentence containing the answer increases.

We associate an initial path set with each question, obtained as described above and exemplified in Table 8.1. We rewrite these paths using paraphrases. More precisely we allow paraphrase substitution by matching the whole path as well as any sub-path (i.e. a valid subsequence of it). The reason for this is that, in many cases, we encounter question paths which are too long to be paraphrased as units as they do not occur sufficiently (or at all) in the input corpus. Finally, we add these expanded paths to the original question paths.

Consider, for example, the question path $X \xleftarrow{subj} discover \xrightarrow{dobj} river \xrightarrow{nn} Y$. We substitute this pattern by rewriting the fragment $\xleftarrow{subj} discover \xrightarrow{dobj}$ with the paraphrases returned by the DIRT algorithm. Table 8.3 shows the top 10 paraphrases extracted by DIRT for this path.

X $\xleftarrow{subj}$ *discover* $\xrightarrow{dobj}$ Y
X $\xleftarrow{subj}$ *find* $\xrightarrow{dobj}$ Y
X $\xleftarrow{pobj}$ *by* $\xleftarrow{prep}$ *discover* $\xrightarrow{subjpass}$ Y
X $\xleftarrow{subj}$ *uncover* $\xrightarrow{dobj}$ Y
X $\xleftarrow{pobj}$ *by* $\xleftarrow{prep}$ *find* $\xrightarrow{subjpass}$ Y
X $\xleftarrow{subj}$ *detect* $\xrightarrow{dobj}$ Y
X $\xleftarrow{subj}$ *use* $\xrightarrow{dobj}$ Y
X $\xleftarrow{subj}$ *seize* $\xrightarrow{dobj}$ Y
X $\xleftarrow{subj}$ *unearth* $\xrightarrow{dobj}$ Y
X $\xleftarrow{subj}$ *recover* $\xrightarrow{dobj}$ Y
X $\xleftarrow{subj}$ *develop* $\xrightarrow{dobj}$ Y

Table 8.3: Top 10 paraphrases for X $\xleftarrow{subj}$ *discover* $\xrightarrow{dobj}$ Y

8.4.3 Candidate answer ranking

Candidate answer ranking is the final stage of the answer extraction module. In this stage, all candidate strings are ranked, and the top string is returned as the answer to the question.

We reduce candidate answer ranking to the task of computing similarity between paths in question and paths in the retrieved sentences. Intuitively the sentence paths which match the question best will contain the answer to the question. QA systems relying on syntactic-level representations have been previously shown to be effective in a number of studies, such as Punyakanok et al. [2004], Cui et al. [2005], Shen and Klakow [2006], motivating thus our choice for a syntax-based answer ranking module.

The candidate answer ranking is done first at sentence level, to return an answer string for each sentence, and at question level, to return an answer for the entire set of sentences associated to a question. The rest of this section describes these two stages.

Sentence-level ranking

We rank the list of candidate answers by using two main ranking keys:

1. Expected answer type
2. Path similarity

If two strings score the same on these two ranking keys, we use a third key which gives preference to longer strings. This rule is introduced to handle cases such as the one previously discussed, in which *Hernando de Soto* is the complete correct answer to the question rather than just *Soto*. The two main ranking keys are detailed for the rest of this section.

Expected answer type In a first step we rank all possible candidates in terms of the expected answer type, with those matching the expected type being ranked at the top.

Typically, in QA systems, expected answer types are deduced based on a named entity recognition/classification component. For example, a *who* question implies an expected answer of type *Person*. We approximate this component with a POS-tag-based heuristic. More precisely, for each question type, we define a set of POS tags which are characteristic of that question type; for example for a *when* question, we expect the answer string to contain a cardinal number. This heuristics is summarized in Table 8.4.

The second key ranks candidate answers based on comparison of sentence paths and question paths. This is the most relevant step and we continue with its description.

Language-model-based path similarity Similarity between paths in syntactic trees can be used as a base for identifying the correct answer string. For example the similarity between $who \xleftarrow{subj} discovered \xrightarrow{obj} river \xrightarrow{nn} Mississippi$

Question type	Expected POS in answer string
who	NNP || NNPS
which	NNP || NNPS || CD
what	NNP || NNPS || CD
where	NNP || NNPS
how	CD
when	CD
other	NNP || NNPS

Table 8.4: Simple POS-tag based heuristic for expected answer typing

and $Soto \xleftarrow{pobj} by \xleftarrow{prep} discover \xrightarrow{dobj} river \xrightarrow{nn} Mississippi$ indicates *Soto* as the correct answer to the question. We propose the use of a language model-based approach to compute path similarity. We start with a brief overview of language models and continue with their use for computing path similarities.

Language models A language model assigns probabilities to sequences of words as estimated from a collection of data, such as a sentence, a document or an entire corpus. Intuitively, these models give a characterization of a language in terms of how likely it is that a string of words has been generated in that language.

Language models have been used in many NLP areas such as speech recognition or information retrieval. In information retrieval the goal is to rank documents d in terms of their relevance to a query q. One common approach to this is to learn language models for each of the documents θ_d. Documents are then ranked based on the probability of the query given the language model of the document $P(q|\theta_d)$. The intuition behind this is that the more likely it is for a query to be generated by the language model of a document, the more relevant the document is to the query.

Unigram language models assume that words in a sequence are generated independently: $P(w_1, ..., w_n) = \prod_i P(w_i)$. n-gram models with $n \geq 2$ are more informative as the occurrence of a word is conditioned on the past $n-1$ words. For example in a bigram model the probability of each word is conditioned on the previous word: $P(w_1, ..., w_n) = \prod_i P(w_i|w_{i-1})$.

Language models can be estimated from direct frequency counts, however, these are unreliable and all language models use some type of smoothing. The purpose of smoothing is to overcome the data sparseness problem and to obtain probability estimates which are more accurate. This is typically done by discounting the probabilities of seen events, and by assigning probabilities larger than 0 to unseen events.

One way to evaluate language models is through perplexity measurements. More precisely, given an unigram language model p, learned from some input data, and an unseen test sample from the same data, $x_1, ..., x_N$, p can be evaluated based on the perplexity with respect to this new sample. This is defined as:

$$PP_p(x_1, ..., x_N) = 2^{-\sum_{i=1}^{N} \frac{1}{N} \log_2 p(x_i)}$$

The exponent is in fact the cross-entropy $H(\hat{p}, p)$ between the language model probability p and the empirical distribution $\hat{p}$ extracted from the sample. A low perplexity score indicates a small cross-entropy between the sample distribution and the model distribution. This shows that the test sample is predictable given the estimated language model, indicating a good language model.

Language models for computing path similarity For computing path similarity, we use a bigram language model with absolute discount smoothing which is trained on question paths.

More precisely, we start by collecting all the paths in a question q connecting the question word and a particular keyword kw. We denote this set by $\mathcal{P}^q(kw)$; this will consist of just one path in the baseline setting (i.e. no paraphrasing) and it will contain a larger number of paths when employing the paraphrase component described in the previous section. We use this set of dependency paths to learn a language model which we denote $\theta_{\mathcal{P}^q(kw)}$. In particular, we use a bigram language model trained with absolute discounting smoothing (Zhai and Lafferty [2004]) with discount parameter $\delta = 0.7$[3].

We denote the set of paths connecting a keyword kw to an answer candidate a_i in sentence s, by $\mathcal{P}^s(kw, a_i)$. Given a sentence path $\pi^s \in \mathcal{P}^s(kw, a_i)$, we compute its perplexity with respect to $\theta_{\mathcal{P}^q(kw)}$ language model. The general perplexity in a bigram model is given by equation 9.1. For our model this results in the formula in equation 9.2.

$$PP_p(x_1, ..., x_N) = 2^{-\sum_{i=1}^{N} \frac{1}{N} \log_2 p(x_i|x_{i-1})} \tag{8.1}$$

$$PP_{\theta_{\mathcal{P}^q(kw)}}(\pi^s) = 2^{-\sum_{i=1}^{N} \frac{1}{N} \log_2 P_{\theta_{\mathcal{P}^q(kw)}}(\pi_i^s|\pi_{i-1}^s)} \tag{8.2}$$

where $\pi^s \in \mathcal{P}^s(kw, a_i)$ is a sentence path of length N, and $P_{\theta_{\mathcal{P}^q(kw)}}(\pi_i^s|\pi_{i-1}^s)$ is the probability of the i^{th} element of the path given the previous one, according to the language model trained on the question paths $\theta_{\mathcal{P}^q(kw)}$.

Intuitively, we compute perplexity as an indicator of how predictable a sentence path is, given the question language model. The lower the perplexity, the higher the chances that the path contains an answer to the question. We therefore use the inverse of the perplexity as a similarity score between question and sentence paths.

One answer candidate a_i typically occurs in more than one sentence path; in this case we consider the path that minimizes the perplexity:

[3]Zhai and Lafferty [2004] show that absolute discounting smoothing performs well for the information retrieval task; they also find that $\delta = 0.7$ is a robust parameter value as it is optimal for a range of various experimental settings.

$$dist_{(q,s)}(a_i) = \min_{\pi^s \in \mathcal{P}^s(kw, a_i)} PP_{\theta_{\mathcal{P}^q(kw)}}(\pi^s)$$

where $dist_{(q,s)}(a_i)$ measures the the inverse of the confidence level of candidate a_i, extracted from sentence s, being an answer to question q.

The intuition behind the paraphrase expansion is that the language model learned on an expanded set of paths will represent the general meaning of a question, rather the particular lexical or syntactic form choices. Consider, for example, the language model learned from the question *Who discovered the Mississippi river?* after the expansion with the phrases in Table 8.3. In this language model, the word *find* will have a high probability and an answer-containing sentence such as *De Soto found the Mississippi river* will have a lower perplexity score under this expanded language model.

We also use the specific rank of a paraphrase, as returned by the paraphrasing algorithm. More precisely, when training the language model on a path set, we weigh the contribution of a path to the model by $\frac{1}{\log_2(\mathrm{R})}$ where R is the path's rank.

Question-level ranking

For question-level scoring, we pool together all the answer candidate strings from a set of question-relevant sentences and compute the overall ranking. We denote the set of question-relevant sentences by $S(q)$; this simply contains sentences which may be relevant to the question, as returned by the passage retrieval component of a question answering system.

In order to aggregate these scores we first transform the distance scores into their inverses. This way, a large distance is associated to a small score and vice-versa. For a particular answer candidate, for each sentence in which it does not occur, we assign a score of 0.

$$\mathrm{score}_{(q,s)}(a_i) = \begin{cases} 0 & \text{if } a_i \text{ not extracted from } s \\ \frac{1}{\mathrm{dist}_{(q,s)}(a_i)+1} & \text{otherwise} \end{cases}$$

We add the term 1 to the nominator to act as a positive "smoothing" parameter which avoids division by zero. The candidate answers are then ranked according to the sum of their sentence-level scores:

$$\mathrm{score}_q(a_i) = \sum_{s \in S(q)} \mathrm{score}_{(q,s)}(a_i)$$

This concludes the presentation of the proposed answer extraction system. Chapter 9 provides the experimental evaluation of this system, when instantiated with DIRT and context-sensitive paraphrasing.

Chapter 9

Evaluation

In this chapter we perform the experimental evaluation of the answer extraction method proposed in Chapter 8. Section 9.1 details on the experimental setup. We integrate and evaluate paraphrases obtained from two sources: 1) the output of the classic DIRT algorithm (Section 9.2) and 2) as returned by the context-sensitive paraphrasing method described in Chapter 7 (Section 9.3).

9.1 Experimental setup

Data We test the baseline pipeline system together with the paraphrase extension module described in the previous chapter on the set of questions in the QASP (Kaisser and Lowe [2008]) data set.

This resource was built using a subset of questions from TREC QA track datasets from years 2002-2006. For each TREC factoid question for which an answer could be found in the AQUAINT corpus, QASP provides the set of answer sentences, together with the corresponding answer string[1].

This data set allows us to evaluate our system for answer extraction both at question level, as standard in QA evaluation, and at sentence level. The question-level evaluation tests a system's performance in retrieving the correct answer for each given question. The sentence-level evaluation is concerned with the actual number of sentences/passages in which the systems identifies the correct answer. The sentence-level evaluation on this data set is particularly suited for the answer extraction task, as all sentences contain the answer, setting a recall upper bound of 100%. The question-level evaluation is not informative on the QASP resource, as in this setting all the candidate sentences contain an

[1]In QASP, questions which in TREC where grouped into series about a certain topic are reformulated such that they can be answered in isolation – that is anaphoric references to the topic were replaced with the topic phrase itself.

answer to the question, which makes the retrieval of the question-level answer much easier. For this reason, we use a second data set containing candidate sentences that are *automatically* retrieved, on which we perform question-level evaluation.

For this latter scenario, for each question, we take all the documents where at least one answer sentence was given in QASP, and run sentence retrieval on the full text of these documents. We use a language-model-based retrieval method as in Ponte and Croft [1998] with Dirichlet smoothing (Zhai and Lafferty [2004]). For each question, we use all sentences with perplexity lower than a threshold of 2000, or the 10 top-ranked sentences if the perplexity is higher than this threshold.

Table 9.1 summarizes these two data sets which we denote as $\text{QASP}_{\text{GOLD}}$, which is the original QASP resource, and $\text{QASP}_{\text{RETRIEVED}}$, which contains automatically retrieved sentences. $\text{QASP}_{\text{GOLD}}$ provides on average approximately 4 sentences per question. The sentence retrieval method previously described results in approximately 15 candidate sentences per question, in the $\text{QASP}_{\text{RETRIEVED}}$ dataset.

	TREC02	TREC03	TREC04	TREC05	TREC06
#Questions	429	354	204	319	352
#$\text{QASP}_{\text{GOLD}}$	2002	1446	865	1456	1405
#$\text{QASP}_{\text{RETRIEVED}}$	8018	5168	3189	5202	5004

Table 9.1: Number of questions and sentences in the QASP data sets.

Evaluation metrics We use accuracy and mean reciprocal rank (MRR) as evaluation metrics. Accuracy reports the percentage of cases for which the correct answer has been extracted and ranked first. In the case of sentence-level evaluation we compute the average accuracy for each sentence, while at question-level we compute a single accuracy score for each question.

The MRR is the reciprocal of the rank of the correct answer, averaged over all sentences/questions. The reciprocal rank (RR) is the inverse of the position of the correct answer in the returned rank:

$$RR(s) = \frac{1}{rank(a)}$$

where s is a sentence and a the correct answer string; $rank(a)$ is the position of this answer string in the candidate answer ranking computed for sentence s. If the answer extraction component has failed to extract the correct answer string, the associated RR score is 0. If the correct answer has been ranked first the RR reaches its maximal value 1. The motivation behind using the MRR metric is that it measures the performance of a system in greater detail than the accuracy metric.

Parsing The system we have described in the previous chapter requires dependency parsing.

Due to limitations of currently available parsing technology, an automatically obtained best parse tree for a naturally occurring sentence has a high probability of differing in at least one way from the analysis that a human expert would assign to it. The problem is exacerbated in question answering, compared to other NLP applications, by the fact that the most widely used training resource for English parsing, the Penn Treebank, contains a very small number of questions. In order to overcome this problem, we represent a question by the set of dependency paths which are extracted not just from the single best parse, but from n-best parses.

We use the Stanford parser (Klein and Manning [2003]) to obtain dependency parses, running it in n-best, unlexicalized, mode for parsing questions. For parsing sentences we only use the best parse obtained from the lexicalized mode.

9.2 DIRT paraphrasing

We acquire DIRT-style paraphrases using a 100-million-word portion of Gigaword (Graff et al. [2003]), which we again parse using the Stanford dependency parser.

We expand each question by paraphrasing noun-ending paths in its dependency graph. In order to acquire paraphrases for a particular path, we use the DIRT method (Lin and Pantel [2001b]) which has been detailed in previous chapters.

9.2.1 Model Selection

A first parameter to be determined is the optimal number of parses used for n-best parsing of questions. Our hypothesis is that a single best parse leads to low scores due to the lack of correct parses from which informative dependency paths can be extracted. A larger number of parses might however introduce noise into the data.

A second parameter to be determined is the number of paraphrases to be used for question expansion. The DIRT algorithm provides a confidence value which we use to control the amount of paraphrases used. A large confidence threshold allows only a small number of high-precision substitutes to be used; a low threshold will result in considerable decrease in precision with a trade-off for larger recall.

We use the TREC 2002 portion of the $\text{QASP}_{\text{GOLD}}$ data set for development and the remaining data for testing. Throughout this section we report on *sentence-level* scores, which, as previously discussed, are much more informative than question-level scores on this data set.

Number of parses We use the development set to see the effect of increasing the number n of parses used in the n-best parsing mode. When using n alternative parses, all the paths extracted from all parses are pooled together leading to a potentially very noisy representation of a question.

We test the effect of parsing on the baseline syntactic model, which does not employ paraphrasing. Figure 9.1 plots the accuracy and MRR scores, as functions of the number of question parses considered.

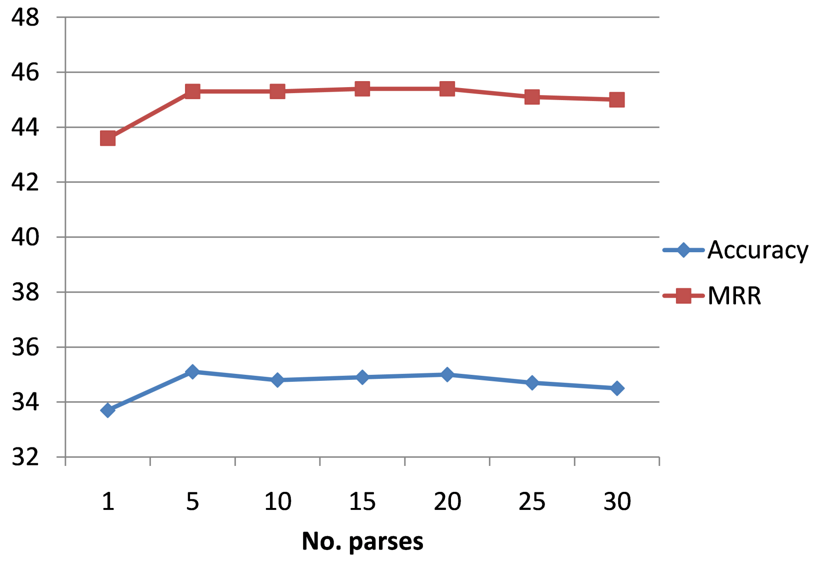


Figure 9.1: Effect of increasing the number n of n-best parses in question parsing. Sentence-level scores on TREC02 portion of $\text{QASP}_{\text{GOLD}}$.

We observe that the accuracy and the MRR scores of the baseline method increase significantly with the number of parses. This confirms our initial hypothesis that the parser is not reliable in finding the best single parse of questions. An example of this is the first question occurring in the data: *Who is Tom Cruise married to?* From the best parse we can extract the path $who \xleftarrow{dep} married_{\text{JJ}} \xrightarrow{subj} cruise$, where *dep* stands for unknown dependency relation. Only the fifth parse results in the correct dependency path $who \xleftarrow{dobj} marry_{\text{V}} \xrightarrow{subj} cruise$.

This increase in performance further indicates that our method is robust with respect to the noise that these parses invariably introduce. We assume that this is the case due to the fact that erroneous parse paths are unlikely to closely match paths in answer sentences. Most of the gain in performance is obtained by using the top 5 parses as opposed to the single best, however slight improvements are obtained when increasing the number of parses to as much as 15. The use of more than 15 parses keeps the performance stable followed by a decrease when using 25 or more parses.

We observe, however, that the performance of the method is limited by the simple heuristics employed for answer extraction. More precisely, we also compute an oracle score on this data set. The oracle score gives the percentage of sentences in which the correct answer is extracted by the candidate answer extraction component. This is the upper bound on the performance of the ranking method, as failure in extracting the correct answer string as a candidate cannot be corrected by any of the subsequent components of the system. While the upper bound on the QASP$_{\text{GOLD}}$ data is 100%, as all sentences contain answers to the questions, we only extract the correct answer string as an answer candidate in 74% percent of the cases; this becomes the upper bound for the ranking method.

Paraphrase confidence threshold We investigate the precision/recall trade-off of the paraphrasing component by testing a number of confidence thresholds. We test a confidence threshold parameter τ with values in the set: $\{0.30, 0.20, 0.10, 0.05, 0.04, 0.03, 0.02, 0.01\}$. These values are chosen based on the confidence score distributions of the paraphrases obtained: a threshold of 0.3 results in generating, on average, one paraphrase, while 0.01 results in expanding with approximately 250 paraphrases on average.

In Tables 9.2 and 9.3 we plot the accuracy and MRR scores against the threshold values, including the no-paraphrasing baseline. We test the effect of paraphrasing when using the 1-best parse and the 15-best parses settings; the 15-best parses setting was the optimal one in the previous experiments.

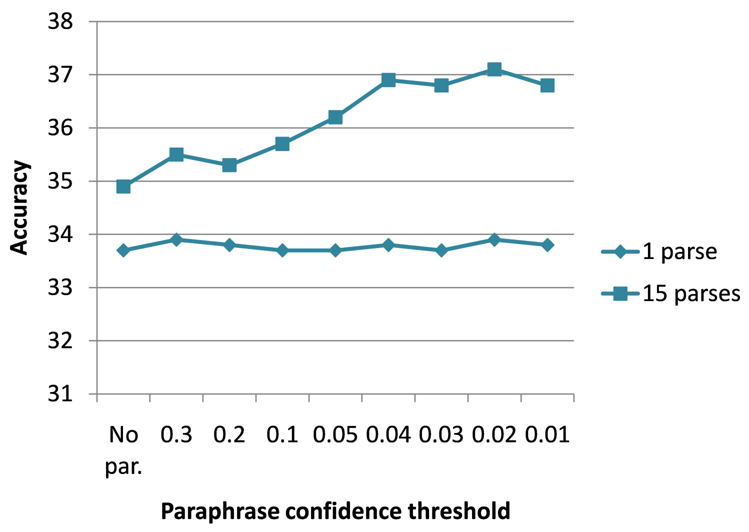


Figure 9.2: Effect of paraphrasing (1-best and 15-best parses). Sentence-level accuracy on TREC02 portion of QASP$_{\text{GOLD}}$.

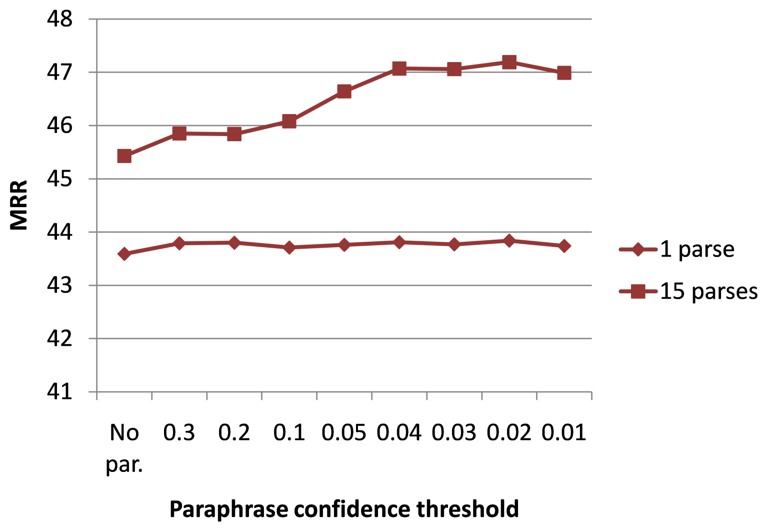


Figure 9.3: Effect of paraphrasing (1-best and 15-best parses). Sentence-level MRR on TREC02 portion of QASP$_{\text{GOLD}}$.

We observe that adding n-best parses and increasing the number of paraphrases has a cumulative effect. Adding paraphrases to the single best parse brings a modest improvement of 0.2 gain in accuracy and 0.25 in MRR. These are obtained when using a confidence threshold of 0.02. This amounts to using, on average, 200 paraphrases for each question path.

The results indicate that the paraphrase component is severely limited by the use of a single best parse as the same confidence threshold of 0.02 brings significantly larger performance gains (2.2% in accuracy and 1.8% in MRR) when using 15 parses.

In order to get more insight into the effect of the paraphrasing component, we analyzed the development set in more detail. More precisely, we are interested in investigating how many times the paraphrase expansions determined a change in the rank of the correct answer string. If adding paraphrases has no effect on the ranking of the correct answer, it is safe to assume that the paraphrase expansions available were not relevant to that question-sentence pair.

The results obtained are plotted in Figure 9.4. We count the number of sentences in which the ranking was altered and compute the MRR scores on these sentences, both without paraphrasing, which is the baseline setting, and with paraphrasing (+ par.). We use 15 parses, and range over different paraphrase thresholds.

We observe that the paraphrases are actually used only in a very small portion of the total number of sentences. The numbers are particularly small considering that answers to the questions are extracted from most of the sentences in this total set, as indicated by the 74% oracle score.

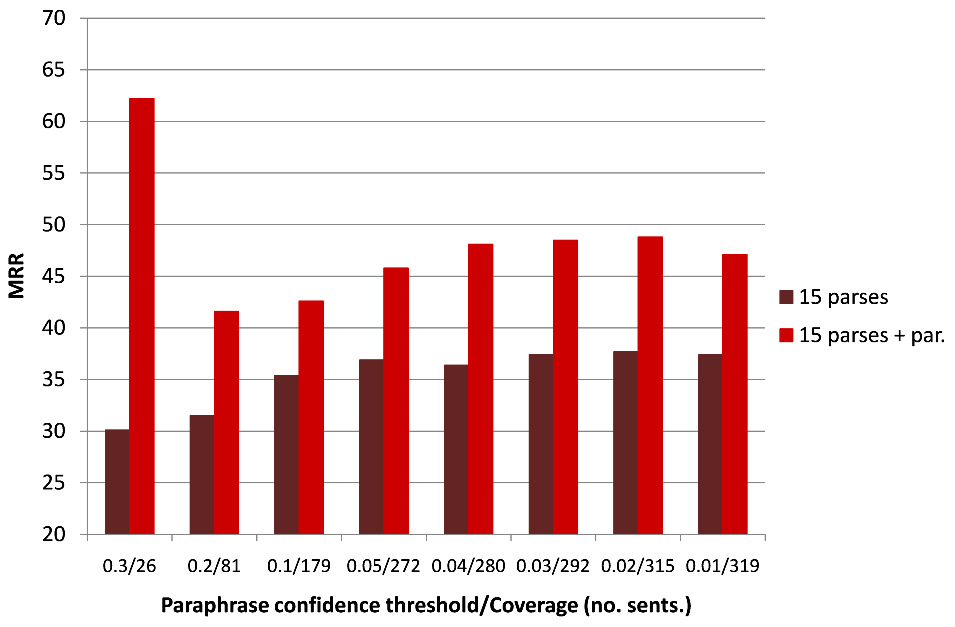


Figure 9.4: Performance on data subsets in which paraphrases are used (out of 2002 total sentences) as a function of paraphrase confidence threshold.

When using a very high confidence threshold of 0.3 we obtain a small number of expansions, which are used only in 1% of the sentences. However, the results indicate that these expansions are very accurate as the absolute gain in MRR score is as high as 30%. As the confidence threshold decreases, we observe that the paraphrases are still beneficial, as they increase performance by ≈10%. However, they are used in a small number of cases, not more that 16% of the data (319 sentences), when using the smallest confidence threshold of 0.01. We also observe that paraphrases are used for the more difficult sentences as the baseline method scores 10% lower on these subsets, as compared to the overall average performance.

9.2.2 Results

We use the optimal development settings on the test data: n-best parses with $n = 15$ and 0.02 paraphrase confidence threshold. We use as test data the TREC03-TREC06 portions of the QASP$_{\text{GOLD}}$. We also test these settings on the automatically retrieved sentences in QASP$_{\text{RETRIEVED}}$.

The notation we use throughout the rest of the chapter is the following:

- **Oracle** (sentence-level/question-level): percentage of sentences/questions in which the correct answer is part of the set of strings extracted by the candidate extraction module.

- **1-best parse**: baseline setting, using the single-best parse for questions
- ***n*-best parses**: 15-best parses for questions
- ***n*-best parses + par.**: 15-best parses for questions plus paraphrasing component with a confidence threshold of 0.02

Sentence-level MRR scores on the gold data are shown in Table 9.2. The oracle scores range between 64%-72%, limiting the performance of the overall answer extraction system. Next, we list the baseline setting (1-best parse and no paraphrases), the *n*-best parses setting and *n*-best parses together with paraphrasing. As it can be observed, the development data results carry over to the

	TREC03	TREC04	TREC05	TREC06
Oracle	65.9	64.3	69.4	72.5
1-best parse	35.5	38.4	41.7	41.2
n-best parses	38.8	40.9	42.6	42.6
n-best parses + par.	39.2	42.2	43.3	43.7

Table 9.2: Sentence-level MRR on QASP$_{\text{GOLD}}$: TREC03-TREC06

test data. Adding *n*-best parses brings significant improvements in all data sets, scores which are further increased by the use of paraphrasing.

Both the use of *n*-best parses and the use of paraphrases add a considerable amount of noise to the representations of the sentences. This is not an issue when testing the method on gold sentences where we obtain significant increase in performance.

We further test our method on the automatically retrieved data. The goal is to investigate which results carry over when using typical question answering data. In this data, the majority of the passages retrieved do not contain the correct answers, making the method potentially less robust to the noise introduced by multiple parses and by paraphrasing. We list the question-level results obtained on the TREC02-TREC06 QASP$_{\text{RETRIEVED}}$ data in Table 9.3.

	TREC02	TREC03	TREC04	TREC05	TREC06
Oracle	74.6	67.2	66.2	70.8	72.7
1-best parse	37.3	31.6	28.0	32.0	32.4
n-best parses	39.4	32.6	28.8	31.6	33.2
n-best parses + par.	41.0	33.6	29.6	32.4	33.8

Table 9.3: Question-level MRR on QASP$_{\text{RETRIEVED}}$

We observe significant improvement on this data set again both from the use of *n*-best parses and from that of paraphrases. The improvements obtained from the paraphrasing component are particularly surprising as we observe that the paraphrases were in fact used only in approximately 3% of the sentences. Although much less informative on this data set, we also list the sentence-level results together with the sentence-level oracle score in Table 9.3; we observe that the question-level improvements carry over from sentence-level ones.

	TREC02	TREC03	TREC04	TREC05	TREC06
Oracle	16.3	15.7	15.0	18.0	16.9
1-best pars.	9.68	8.10	9.02	10.08	9.48
n-best pars.	10.24	8.80	9.35	10.14	9.71
n-best pars. + par.	10.59	8.91	9.52	10.39	9.90

Table 9.4: Sentence-level MRR on QASP$_{\text{RETRIEVED}}$.

9.2.3 Discussion

In order to better understand the effect of paraphrasing, we manually inspected a subset of the sentences in which paraphrases help improve performance and a subset of those in which they harm performance. We use the QASP$_{\text{RETRIEVED}}$ data set for this purpose.

An example of performance gain is the following question-sentence pair, from the TREC02 QASP$_{\text{RETRIEVED}}$:

(1) What year was Alaska purchased?

(2) In Seward, the town named for Secretary of State William Seward, who bought Alaska for $7.2 million in 1867, a multimillion-dollar industry has developed around ships that take visitors to the bird rookeries and glaciers of Kenai Fjords National Park.

DIRT adds a number of paraphrases to the question path X $\xleftarrow{subj}$ *purchase* $\xrightarrow{obj}$Y, including X $\xleftarrow{subj}$ *buy* $\xrightarrow{obj}$Y, *Y* $\xleftarrow{pobj}$ *to* $\xleftarrow{prep}$ *sell* $\xrightarrow{obj}$ X. Under the language model trained on these expansions, the path 1867$\xleftarrow{pobj}$ *in* $\xleftarrow{prep}$ *buy* $\xrightarrow{obj}$Alaska has the lowest perplexity and the correct answer, 1867, is ranked highest.

We observe a number of similar cases, in which lexical variation is introduced by the use of paraphrasing. For example, the sentence fragment *the 1990 invasion and occupation of Kuwait by Iraq* is used to return *Kuwait* as the answer to *What country did Iraq invade in 1990?* The highest ranked paraphrases influence the language model most, and in this example DIRT returns *X's occupation of Y* and *X's invasion of Y* as the most confident paraphrases of *X invade Y*. Examples of other useful variation that we encounter are *X won Y* $\approx$ *X, the winner of Y*, *X manufacture Y* $\approx$ *Y made by X* or *X governs Y island* $\approx$ *X's* *island of Y.*

We observe, however, that in many cases in which the paraphrases improve performance, their effect is that of allowing more flexibility at the syntactic level, rather than introducing more complex variations. For example, we encounter the question *Who is the director of the Hermitage museum?* The language model trained on paraphrase expansions ranks the sentence path $piotrovski \xleftarrow{appos} director \xrightarrow{poss} museum$ as most similar to the question path $who \xleftarrow{subj} director \xrightarrow{prep} of \xrightarrow{pobj} museum$. This leads to finding the correct answer *Dr. Mikhail Piotrovsky*, however only by allowing variation at the syntactic level.

We also observe that the added syntactic flexibility is also a common source of error. For example $\xleftarrow{pobj} over \xleftarrow{prep} kill \xrightarrow{subj}$ is ranked most similar to $\xleftarrow{dobj} kill \xrightarrow{subj}$ for the question-sentence pair *How many people has ETA killed? - ETA has killed nearly 800 people over the last 30 years.* In this case it leads to the the wrong answer *30 years* being chosen over the correct answer *800*.

Next, we inspect whether the effects of n-best parsing and paraphrasing are uniform over different types of questions. We divide the questions into different classes based on the first word present in the question. We obtain six categories: *who, what, which, how, when, where* and we label as *other*, questions which do not match any of these. Figure 9.5 shows the MRR scores broken down per question type on the
TREC_{RETRIEVED} data set.

We observe that for all types of questions, multiple parses improve the baseline performance with the exception of *when* questions where we observe small decrease in MRR. In general, the effect of paraphrasing is not uniform over the different types of questions. The highest gains are observed for *who* and *what* questions. The latter form more than one third of the questions in this data set, which explains the overall increase in performance. For the other question types we observe either insignificant gains or decrease in MRR scores. In particular, *which* questions seem to suffer from the use of paraphrases.

A closer look at the *which* question reveals that in many cases the answers to *which* questions are relatively close to one of the question keywords, and information present further away in the sentence is usually not informative. The language model method has no bias towards returning shorter or longer paths as answer-containing paths; in the case of *which* questions, these shorter paths seem however preferable.

Bias towards smaller paths can be easily introduced in our system. We rank paths based on perplexity scores which compute cross-entropy between the empirical distribution, in our case obtained from the candidate path, and the language model distribution learned from the question representation. The cross-entropy term can be adjusted to favor smaller paths by replacing $\frac{1}{N}$ with

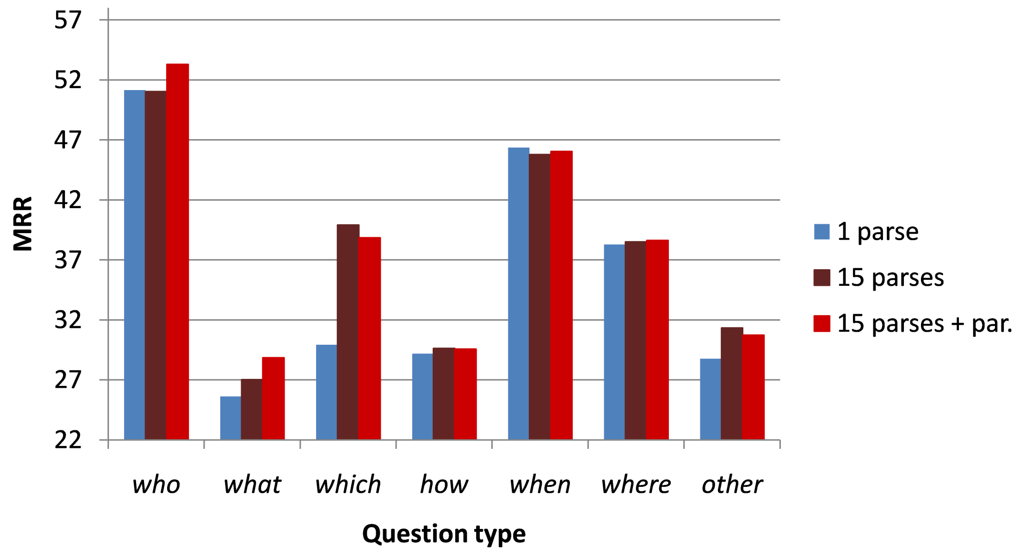


Figure 9.5: Performance by question type on QASP$_{\text{RETRIEVED}}$.

$\frac{1}{N+d}$, where we call d the perplexity discount parameter. The adjusted perplexity becomes:

$$PP_p(x_1, ..., x_N) = 2^{-\sum_{i=1}^{N} \frac{1}{N+d} \log_2 p(x_i|x_{i-1})} \tag{9.1}$$

$d = 0$ is the original setting which introduces no bias for path length, while larger d values favor shorter paths. Figure 9.6 shows the MRR scores (with/without paraphrasing) when varying the discount parameter d.

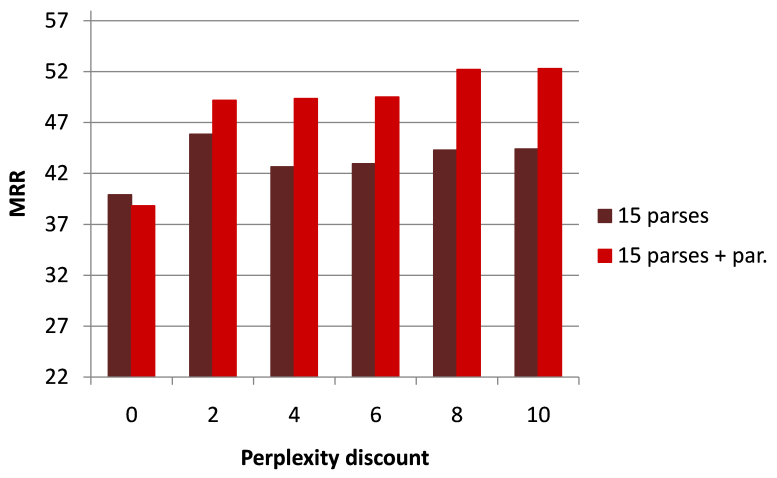


Figure 9.6: Performance for *which* questions when introducing bias for shorter paths on QASP$_{\text{RETRIEVED}}$.

We observe that giving preference to shorter paths brings significant improvements. The paraphrase component becomes beneficial in this setting and in total we see an MRR gain of approximately 15% over the original un-biased setting. This is mirrored by more than 10% gain in accuracy. However, when employed for other question types, this bias determines no significant performance changes from the original method.

This experiment seems to indicate that tailoring the settings to the properties of individual question types may determine better performance and allow for higher gains obtained through paraphrasing.

9.3 Context-sensitive paraphrasing

In a second series of experiments we integrate our context-sensitive paraphrasing component in the baseline answer extraction pipeline. Intuitively, paraphrases that are generated this way should allow for more accurate question expansions compared to the DIRT method.

The context-sensitive paraphrasing component is integrated similarly to the DIRT method, by expanding the set of question paths. Consider for example the question:

(3) Q: What Spanish explorer discovered the Mississippi River?

The DIRT method paraphrases the path $X \xleftarrow{subj} discover \xrightarrow{dobj} Y$ independently of a given context, as in Table 8.3. In this second experiment, we use paraphrases which are specific to the given context of X:*explorer* and Y:*river*. Naturally, other occurrences of this path, i.e. with different X and Y instantiations, will be paraphrased differently; in contrast, DIRT returns the same expansions for all of these.

In our experiments, we use the LDA-based method described in Chapter 7. More precisely, the similarity between paths p_i and p_j in context w_X, w_Y is computed as $sim(vec(p_i, w_X), vec(p_j, w_X)) * sim(vec(p_i, w_Y), vec(p_j, w_Y))$, where $vec(p, w)$ is the contextualized representation of path p with context word w. We employ, as in the previous chapters, scalar product (sp), cosine (cos) and inverse Jensen-Shanon divergence (JS) as similarity measures. Table 9.5 lists the top ranked paraphrases for $explorer \xleftarrow{subj} discover \xrightarrow{dobj} river$ when using sp and JS as similarity measures.

In general, we observe that the precision of context sensitive (henceforth CS) paraphrasing is lower than DIRT's. Many of the expansions of a path are not appropriate as substitutes. Some of the phrases obtained when performing the substitution are unlikely to occur in natural language, such as *river institute of explorer* or *river academy of explorer* in the example above. However, in many cases, the lexical variation encoded in these expansions is appropriate to the

explorer, $X \xleftarrow{subj} discover \xrightarrow{dobj} Y$, **river**	
sp similarity	JS similarity
$X \xleftarrow{subj} find \xrightarrow{prep} by \xrightarrow{pobj} Y$	$X \xleftarrow{subj} unearth \xrightarrow{dobj} Y$
$X \xleftarrow{pobj} of \xrightarrow{prep} expedition \xrightarrow{nn} Y$	$X \xleftarrow{subj} excavate \xrightarrow{dobj} Y$
$X \xleftarrow{pobj} of \xrightarrow{prep} institute \xrightarrow{nn} Y$	$X \xleftarrow{subj} find \xrightarrow{prep} at \xrightarrow{pobj} Y$
$X \xleftarrow{subj} dig \xrightarrow{dobj} Y$	$X \xleftarrow{partmod} study \xleftarrow{dobj} Y$
$X \xleftarrow{pobj} of \xrightarrow{prep} civilization \xrightarrow{nn} Y$	$X \xleftarrow{subj} discover \xrightarrow{prep} in \xrightarrow{pobj} Y$
$X \xleftarrow{pobj} by \xleftarrow{prep} design \xrightarrow{partmod} Y$	$X \xleftarrow{prep} with \xleftarrow{pobj} academy \xrightarrow{nn} Y$
$X \xleftarrow{pobj} by \xleftarrow{prep} complete \xrightarrow{nsubjpass} Y$	$X \xleftarrow{prep} from \xleftarrow{pobj} academy \xrightarrow{nn} Y$
$X \xleftarrow{pobj} at \xrightarrow{prep} committee \xrightarrow{nn} Y$	$X \xrightarrow{partmod} work \xrightarrow{prep} for \xrightarrow{pobj} Y$
$X \xleftarrow{partmod} study \xleftarrow{dobj} Y$	$X \xleftarrow{subj} find \xrightarrow{prep} with \xrightarrow{pobj} Y$
$X \xleftarrow{subj} unearth \xrightarrow{dobj} Y$	$X \xleftarrow{pobj} by \xrightarrow{prep} discover \xrightarrow{nsubjpass} Y$

Table 9.5: Context-sensitive paraphrases for $X \xleftarrow{subj} discover \xrightarrow{dobj} Y$

given context. In the current example, *civilization*, *expedition*, *study* or *institute* are lexical items indicative of the context word *explorer*. In contrast, the DIRT expansions of this same path, shown in Table 8.3, are highly accurate and the lexical variants, such as *find* and *detect* are appropriate to most meanings of *discover*.

In the remainder of this section, we present the results obtained when using the context-sensitive paraphrasing component; in particular, we are interested in investigating whether the answer extraction method benefits from the variation introduced this way, despite the large amount of noise present in the CS expansions.

9.3.1 Model selection

Similarly to the previous section, we use the TREC02 portion of QASP$_{\text{GOLD}}$ for development.

In particular, we investigate the changes in performance as a function of the number of paraphrases employed. Unlike for DIRT paraphrasing, confidence level thresholds are difficult to determine in the case of CS paraphrasing: the absolute similarity scores vary from one similarity measure to another as well as within different paraphrasing strategies. Different paraphrasing strategies are determined by different ways of contextualizing. More precisely, if a particular context word is not encountered often enough in the input corpus (e.g. some proper names), it will not be present in our input matrix and, therefore cannot be used as a context feature. In general, we only use the available context words, which leads to using no context at all if none of the X or Y filler words are available. To vary the amount of paraphrases, we simply use the top-n

most confident expansions and vary the value of n rather than using confidence thresholds.

Throughout this section we use the following notation for different paraphrasing methods:

- **DIRT**: paraphrases obtained using the DIRT algorithm of Lin and Pantel [2001b] (no context information is used)
- **CS**$_{sim}$: Context Sensitive paraphrases obtained with the LDA-based method of Chapter 7, using similarity measure $sim \in \{\text{sp}, \cos, \text{JS}\}$.
- **NC**$_{sim}$: No Context (i.e. context-ignoring) paraphrases obtained with the LDA-based method of Chapter 7 and using similarity sim.

The LDA-based method we have proposed can be used both for context-sensitive paraphrasing as well as in a context-ignoring mode, denoted CS and NC, respectively. We vary the number of top-n expansions used with n ranging from 5 to 250, when added to the single best parse and 15 best-parses settings, using the three similarity measures listed above. In Figure 9.7 we plot the results obtained with the cosine similarity measure.

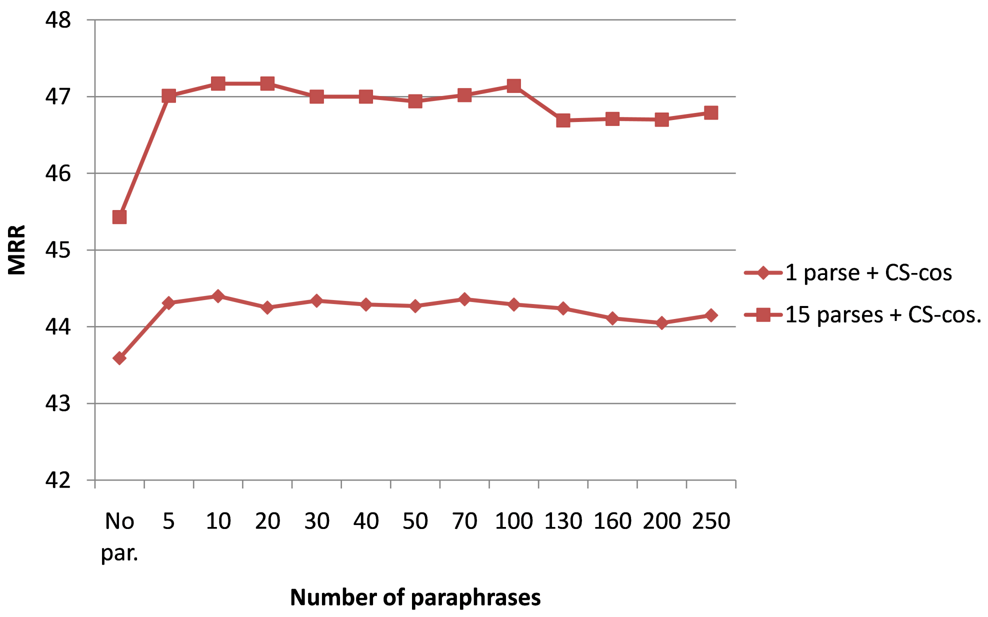


Figure 9.7: Effect of CS paraphrasing on 1-best and 15-best parses. Sentence-level MRR on TREC02 portion of QASP$_{\text{GOLD}}$.

Similarly to the previous results, adding paraphrases and using 15-best parses is more beneficial than adding expansions to the single best parse. The absolute

MRR gains are in the range of 2% for 15-best parsing and 1% for 1-best parse; the best performance is obtained when using 10 expansions, in both settings.

The different similarity measures perform similarly, with scalar product slightly outperforming JS, which in turn slightly outperforms cosine. As the expansions returned by these measures differ to a significant extent form each other, we also test a mixture method which uses top-n sp expansions, followed by top-n, JS and top-n cosine. Despite its simplicity, this mixture method outperforms the use of all individual similarity measures.

Figures 9.8 and 9.9 show the MRR scores of the CS (mixture) method against the DIRT method, when varying the number of paraphrases. In the single best

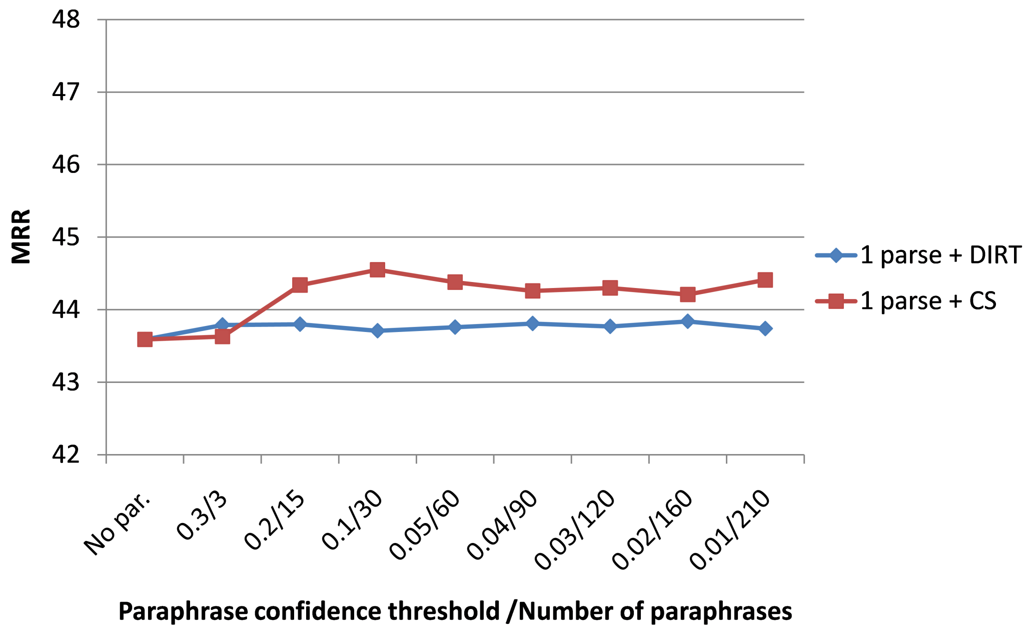


Figure 9.8: CS paraphrasing (mixture over the three similarity measures) vs. DIRT on 1-best parse. Sentence-level MRR on TREC02 portion of QASP$_{\text{GOLD}}$.

parse setting, we observe that CS paraphrases are beneficial as opposed to the DIRT ones which bring almost no improvement in this setting. In the 15-parses setting, we also observe significant differences between the two methods. The CS method only slightly outperforms DIRT, however the maximal performance gain is obtained when using as little as 30 paraphrases (10 from each similarity measure) while DIRT's performance peaks at a threshold of 0.02 which amounts to approximately top-200 paraphrases on average[2].

[2]In figures 9.8 and 9.9, the confidence thresholds for DIRT and the number of CS paraphrases are approximately aligned: for example, 0.01 confidence threshold for DIRT corresponds to using slightly less than 250 paraphrases on average. This is aligned to using 210 paraphrases of the CS method

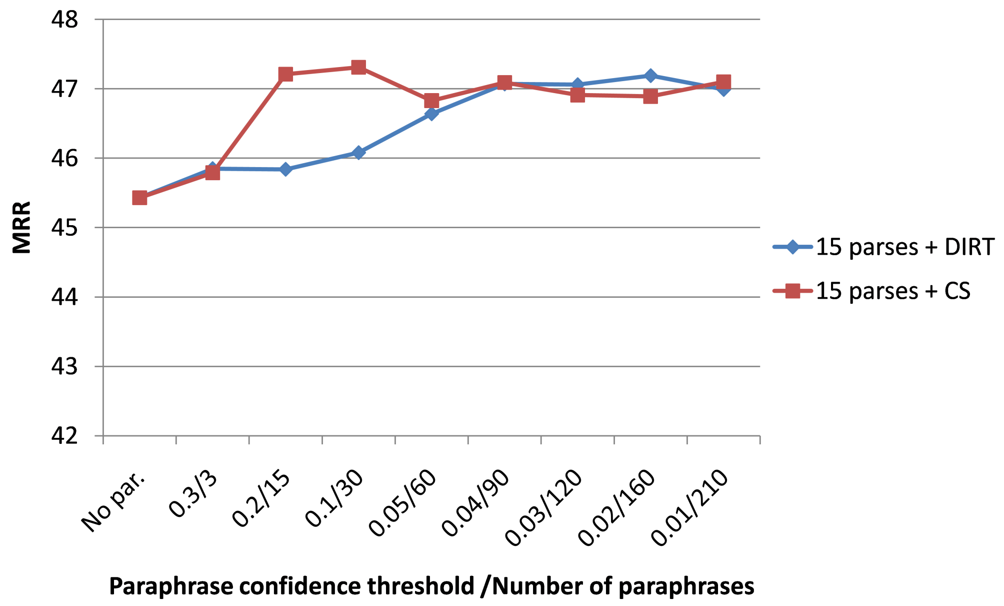


Figure 9.9: CS paraphrasing (mixture over the three similarity measures) vs. DIRT on 15-best parses. Sentence-level MRR on TREC02 portion of $\text{QASP}_{\text{GOLD}}$.

The results suggest that beneficial, appropriate expansions are found at the very top of the CS paraphrases, while in order to achieve the same goal, a much larger number of general-meaning DIRT paraphrases are required.

In order to verify that the differences observed are caused by the use of context, we also test the non-contextualized LDA-based method. The similarity between paths p_i and p_j is computed as $sim(vec(p_i), vec(p_j))$, where $vec(p)$ is the LDA dimensionality-reduced representation of path p. We use the notation NC (No Context) for this model. The results are plotted in Figure 9.10 (using cosine as similarity measure).

Similarly to DIRT, the NC method benefits from using a large number of paraphrases, and achieves the best performance only when as many as 200 expansions are generated for each question path. This confirms the intuition that the context-sensitive method ranks the useful expansions at the very top while context-ignoring methods such as DIRT and NC require many more expansions to achieve the same goal.

These results are summarized in Table 9.6, where we list the MRR values obtained on the development set when employing the different paraphrasing methods.

Overall, the best CS method, the mixture setting, outperforms DIRT, however the NC settings employing a large number of expansions have the best

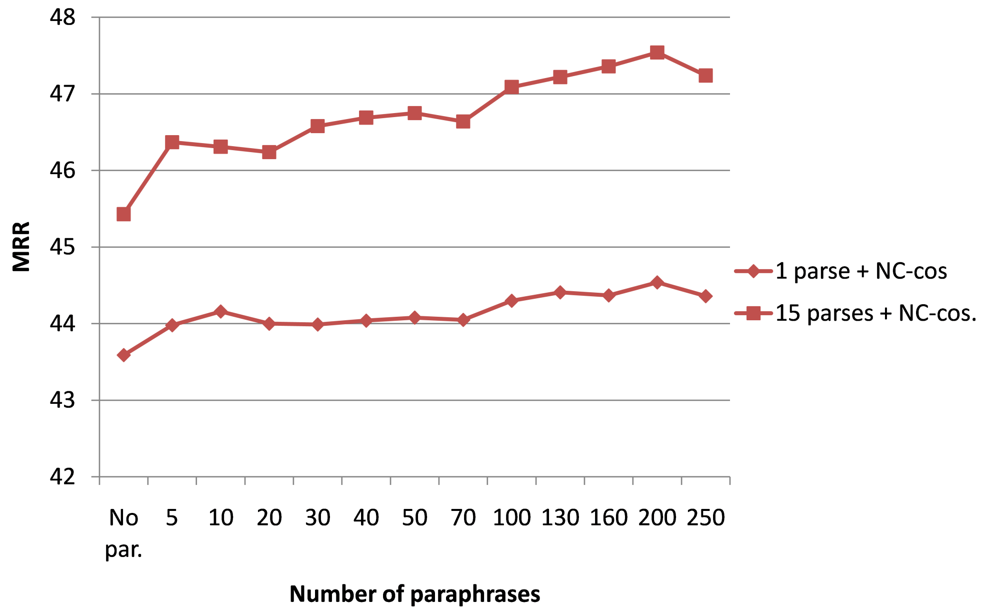


Figure 9.10: Effect of NC paraphrasing on 1-best and 15-best parses. Sentence-level MRR on TREC02 portion of $\text{QASP}_{\text{GOLD}}$.

performance on the development set.

Similarly to the previous section, we investigate in more detail what is the effect of CS paraphrasing on the development set. More precisely, we compute in how many sentences the paraphrase expansions determine a change in the ranking of the correct answer string (we use the term coverage to stand for this percentage of sentences). The results are shown in Figure 9.11. We give the coverage when varying the number of expansions as well as the performance gains on the covered data subsets.

Similarly to DIRT, we observe that paraphrases bring significant improvements over the baseline setting, in the range of 5%-15% absolute MRR gains. We again observe, however, that these are used in a minority of cases, only in 15% of the sentences when using the most relaxed setting; the upper bound is set by the number of sentences in which the correct answer is extracted as a candidate, which is ≈75%. The major difference from DIRT is that a similar coverage is obtained by using much fewer expansions. For example, the coverage obtained using the top-3 most confident CS paraphrases is only achieved when using top-30 DIRT paraphrases. This finding matches with the overall MRR scores shown Table 9.6: for CS paraphrasing, both coverage and MRR performance peak when using a small number of expansions.

Method	Confidence level setting	MRR	Using Context
DIRT	$\tau = 0.02$ (≈top-200)	**47.19**	
CS_{sp}	top-10	47.17	x
CS_{JS}	top-20	46.92	x
CS_{cos}	top-20	46.85	x
CS_{MIXT}	3 x top-10	**47.31**	x
NC_{sp}	top-200	47.27	
NC_{JS}	top-200	**47.66**	
NC_{cos}	top-200	47.54	
NC_{MIXT}	3 x top-100	47.41	

Table 9.6: Results for different paraphrasing methods (15-best parses). Sentence-level MRR on TREC02 portion of $QASP_{GOLD}$.

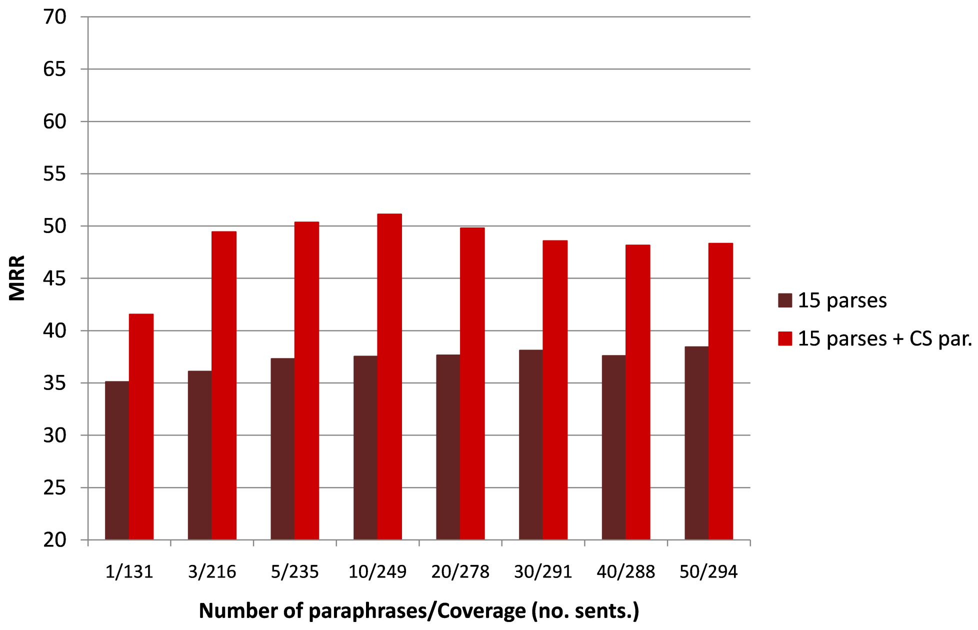


Figure 9.11: Performance data subsets which make use of paraphrases (out of 2002 total sentences) using CS paraphrasing (sp similarity) as a function of the number of paraphrases used.

9.3.2 Results

We use the TREC03-TREC06 portions of $QASP_{GOLD}$ as well as TREC02-TREC06 of $QASP_{RETRIEVED}$ as test data. We test the baseline single best parse and 15-best parses settings, as well as 15-best parses together with the paraphrasing methods.

We compare the three paraphrasing methods using their best development set settings:

- **DIRT**: confidence threshold of $\tau = 0.02$ ($\approx$ 200 expansions on average).
- **CS**: mixture of all three similarity measures, 3×10 expansions in total.
- **NC**: inverse JS similarity measure, 200 expansions.

In Table 9.7 we show the results on the QASP$_{\text{GOLD}}$ test data. For simplicity the four data sets are concatenated and one overall score is reported. The corresponding results on the retrieved data are reported in Table 9.8.

	TREC03-06
1-best parse	39.23
15-best parses	41.27
15-best parses + DIRT par.	42.09
15-best parses + CS par.	42.14
15-best parses + NC par.	41.97

Table 9.7: Sentence-level MRR on TREC03-TREC06 portions of QASP$_{\text{GOLD}}$

	TREC02-06
1-best parse	32.87
15-best parse	33.80
15-best parse + DIRT par.	34.87
15-best parse + CS par.	34.61
15-best parse + NC par.	34.87

Table 9.8: Question-level MRR on TREC02-TREC06 portions of QASP$_{\text{RETRIEVED}}$

We observe no significant difference between the MRR scores obtained with the three paraphrasing methods on the test data. In particular DIRT and NC perform surprisingly similar of both data sets.

9.3.3 Discussion

The results seem to suggest a potential upper bound on the performance gains obtained from paraphrasing in our experimental setting, irrespective of the paraphrase method used. Other than the particular experimental setting in itself, two other potential sources of this limitation may be 1) the extent to which paraphrasing can account for the question-sentences variation encountered in the data and 2) the specific distributional paraphrasing setting used. We further detail on these two aspects.

One important question that remains unanswered concerns the degree to which paraphrasing can explain the variation encountered in the data. In our experiments we observe that all three paraphrasing methods, when used, bring a significant, over 10% absolute gain, however their coverage is severely limited to a small number of cases. This observation corroborates the findings of previous work on QA or recognizing textual entailment. This low coverage may simply be specific to the problem, or may be a real shortcoming of the paraphrasing method. In particular, we think that only a close investigation of QA data focused on analyzing the paraphrasing aspect of this data, can answer this question.

A second upper bound may be set by the input data used to learn the paraphrases. All three methods extract the same total set of syntactic patterns ($\approx$80000), and furthermore, learn to paraphrase from the same evidence, the left and right filler words. Variation that is not covered by the set of patterns used may be required in order to improve on these results. In particular the use of larger amounts of data as well as of larger context, going beyond the filler nouns, may allow the methods to perform better. We expect this to be the case especially for the CS methods, as these methods attempt to extract significantly richer knowledge from the same amount of evidence. And finally, the general limitations of the distributional paradigm may set an upper bound on the performance on these methods.

Significant qualitative differences between context-sensitive paraphrasing and the two context-ignoring paraphrasing methods can still be observed, despite their similar performance. A small number of paraphrases is sufficient for the CS method in order to obtain the same results as NC (30 expansions vs. 200) suggesting that the variation obtained by the CS paraphrasing is more precise, more specific. Furthermore, we have performed an analysis of the results per question type, which reveals a different pattern for the CS method as opposed to the no-context settings. For example the precision on *who* questions, which benefit most from NC paraphrasing, is harmed by the use of CS expansions. This seems to suggest again, that adapting the approach to handle variation specific to different question types may prove itself helpful.

9.4 Summary

In this chapter we have evaluated the proposed syntax-based method for dealing with lexical-syntactic variation in question answering. Dependency path matching, as well as paraphrasing components for passage retrieval or answer extraction have been used before in the literature. Unlike previous work, we have focused on enhancing a baseline syntactic system solely with knowledge acquired in an unsupervised fashion. The goal was to build a robust model, in which the paraphrasing is used not only to provide an *exact* match between a question and a candidate sentence (such as Poon and Domingos [2009], to our knowledge the only other account on using DIRT for QA) but rather to allow the representations of questions and answer-containing sentences to be brought closer.

We use a paraphrase component together with multiple parses for the questions in order to expand the representation of a question's meaning. The representations obtained contain, most likely, a number of dependency paths which do not reflect the meaning of the question; obvious error sources are the multiple alternative parses used or sense ambiguity in paraphrasing. However, we obtain more variations than precision-oriented methods, which seems to be helpful for answer extraction, a task for which multiple sources of evidence (e.g. a number of sentences) are aggregated to produce a single answer. We obtain improvements in answer extraction MRR scores both on the QASP gold sentences data set and on a noisier collection of automatically retrieved sentences. These improvements are obtained despite the fact that, similarly to previous work such as Dinu and Wang [2009], we notice that paraphrases are used only in a small number of cases. We plan to further investigate to what extent this phenomenon is caused by the paraphrase resource.

In particular, we test two types of paraphrasing, the previously proposed DIRT method and the context-sensitive method we have developed in the previous chapters. Despite the observation that the context-sensitive expansions are much nosier, in practice they bring similar (in 15-parse setting) or better (in 1-parse setting) improvement than the context-ignoring methods. In contrast to the context ignoring methods, a small number of context-sensitive paraphrases is sufficient in order to achieve the same performance gains, suggesting that this method properly identifies the context-appropriate expansions.

Given the positive results reported in this chapter, we would like to further explore some other research prospects. We would like to attempt to use domain adaptation techniques in order to train a dependency parser which deals better with questions and introduces less noise to the question meaning representation. If fewer errors are propagated this could also mean that we gain more from paraphrasing. Another prospect is that of using information about the question type to guide the system to better answers. As shown in this chapter there are indications that performance gains can be obtained by adapting the general method we propose here to specific question types.

Chapter 10

Conclusions

This chapter summarizes the main contributions of the thesis and discusses directions for future work.

10.1 Summary

This thesis focuses on the use of unsupervised, distributional methods for semantics. In the first part, we address the issue of *context-sensitive* distributional representations and their use for assessing similarity in meaning. The second part of the thesis focuses on the use of distributional paraphrasing for question answering.

Context-sensitive distributional similarity The issue of adapting vector space models, or distributional methods in general, to the computation of context-sensitive similarities has received increased attention in recent years. This has been driven by the observation that a type-level representation, mixing together all the distinct meanings or usages of a word, is not suitable in real applications, which may deal with words occurring in context, disambiguated words. This observation has been made for a number of distributional methods such as vector space models (VSMs) operating at word-level, for assessing lexical similarity (Erk and Padó [2008], Thater et al. [2010]), or for phrase-level VSMs used in distributional paraphrase acquisition (Pantel et al. [2007], Connor and Roth [2007]).

The first part of the thesis addresses this problem by proposing a probabilistic framework for context-sensitive distributional similarity. Specifically, the framework developed represents words, occurring in isolation or in context, as probability distributions over a global set of corpus-induced meaning components, or meaning aspects. When given a word without a context, this representation reflects its a priori meaning as a distribution over a set of latent meaning components. When given a context, a shift in this distribution determines a *disambiguated* representation in which meaning components which are validated by the context become more likely.

In turn, the meaning components themselves are induced form the corpus in an unsupervised fashion. The intuition behind this is that each occurrence of a word together with a contextual feature is explained by a latent meaning, or latent meaning component, and the sum of all occurrences of a word is a mixture over such latent classes. The goal is to induce the set of latent classes that best explain the corpus co-occurrence data. For this, we follow Hofmann's (Hofmann and Puzicha [1989]) framework for unsupervised learning from dyadic data and we use two variations of this method to induce latent classes.

Given a set of latent classes, a word is represented as a distribution over classes, while words occurring in context are represented in terms of posterior distributions, reflecting the probability of each class conditioned on the given context. Such representations can now be compared with a clear interpretation: words, occurring both isolated or in context, have similar meaning if they trigger the same latent components.

Unlike previous work, our framework models the meaning of words in context in a probabilistic setting, in which the meaning representations as well as the similarity computations are obtained in a natural, intuitive fashion. Furthermore, the framework is completely modular, as it can be applied to any type of vector space model and used with any suitable latent variable induction model/algorithm. In this thesis, the framework is instantiated on a word-level VSM, for the task of lexical substitution as well as on a paraphrase acquisition VSM; we obtain promising results on both of these tasks.

Distributional paraphrasing for question answering The second part of the thesis focuses on the application of distributional paraphrasing to question answering. This is driven by the observation that despite the development of methods for unsupervised paraphrase acquisition, reports on using such methods in end-user NLP tasks, or in QA in particular, are almost inexistent. Similarly, most of the more recent work on context sensitive distributional similarity has not been tested so far in an application scenario. For this reason, in this second part of the thesis we build a context-sensitive extension of the paraphrasing method of Lin and Pantel [2001a] and test this, together with the original method, on the task of answer extraction.

Question answering is the task of automatically extracting answers to questions from large collections of text. One of the main issues that QA systems face is caused by the fact that the same meaning is often expressed differently in questions and in answer-containing sentences. The use of paraphrases is one of the most natural ways to address this problem and methods such as Lin and Pantel [2001a] have been developed with applications such as QA in mind.

This part of the thesis focuses on the task of answer extraction, in which a system is provided with a question and a set of candidate sentences and the goal is to extract the exact string answering the question. We propose a language-model based answer extraction module, which can be naturally enhanced with a paraphrasing component. In particular, unlike previous work, we focus on robustness: rather than using the paraphrase rules to obtain exact matches between sentences and questions, we rather allow them to *reduce* the distance between question and answer-containing sentences.

We observe that the use of paraphrasing brings overall improvements, which although significant, are severely limited by low coverage: while large improvements can be observed on the sentences in which paraphrases are used, their number is very small. In future work we plan to investigate in more detail the limitations we observe, in order to identify if these are true limitations of the paraphrasing method, or if they are caused by other factors. The results observed may reflect that paraphrases are in reality not a frequent phenomenon in this setting. A related issues is that of separating the performance of a paraphrasing component from the particularities of the QA system that it is integrated in; the improvements we observe can be very relevant if they carry over to complex, very accurate QA systems, however this question remains open, as the development of a fully-fledged QA system was not the goal of this work.

10.2 Outlook

A number of future work directions have been discussed throughout the thesis, in the context of the individual aspects under consideration. This section highlights the potential use of the work presented here for information retrieval, one of the most widely-used NLP applications.

Context-sensitive similarity computations are potentially relevant to all applications that require similarity in meaning of words or phrases which naturally occur in context. In particular, we are interested in using this framework for the tasks of query expansion/query reformulation in information retrieval.

One of the main issues that information retrieval engines face is that of term mismatch: documents relevant to a user's query may be using different terms than the ones present in the query. Query expansion methods attempt to solve this by identifying alternative, similar terms and adding these to the set of query words; the resulted expanded query is then used by the search engine to retrieve matching documents, instead of the original query. For example, for the query *how to make bombs*[1], expanding the term *make* with its synonym *build* can help retrieve a potentially relevant document entitled: *How to build homemade nuclear weapons for peacful and defensive purposes.* A related task is that of

[1]The query suggestions and retrieval examples given throughout this section are obtained using the Google search engine.

generating query suggestions. This consists in finding alternative, meaning-related reformulations of a user's query which can be further on proposed to the user in order to help them refine the search and facilitate the retrieval of relevant documents. For example, for the query *Sir Arthur Conan Doyle books* (Alfonseca et al. [2009]), possible suggestions are related queries such as *Sir Arthur Conan Doyle works* or *Sherlock Holmes books.*

In both cases, the ability to reliably asses similarity between queries or the similarity of words occurring in query context plays an important role. In query expansions methods, it is important that the term expansions found are appropriate to the query, i.e. that the added terms correspond to the correct meaning of the words in the context of the query. While this is done implicitly by a number of query expansion methods, work such as Riezler et al. [2007] and Riezler and Liu [2010] focuses on obtaining context-sensitive synonyms by making use of machine translation or paraphrasing engines. The idea behind this is that these methods will produce rephrasings of the query; these in turn contain rephrased *terms* which are context appropriate. While all these methods may be used to achieve this goal, the framework developed in this thesis is targeted at exactly this task and in future work we plan to investigate this application.

For the related task of query suggestion, composition in vector space models has already been proposed as a means to estimate the similarity of two queries, in Alfonseca et al. [2009]. Given a target query, the task is to rank the entire set of queries based on the similarity to the target and return the most similar ones. In particular, the authors show that the choice of component-wise geometric mean for composing the representations of words to form a representation of the query performs particularly well. This corroborates the results of Mitchell and Lapata [2008] as well as the experimental results presented in this thesis, in which component-wise multiplication of words has proved to be a very competitive baseline. A future work direction can be to investigate the use of the method developed in this thesis for the comparison of queries.

Appendix A

Dirichlet distribution

The Dirichlet distribution is a K-dimensional distribution, $K \geq 2$, with parameters $\alpha = (\alpha_1, ...\alpha_K)$, $\alpha_1, ..., \alpha_K > 0$ with the following probability density function:

$$f(x_1, ..., x_K; \alpha_1, ..., \alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{A.1}$$

for $x_1, ...x_{K-1} > 0$, $x_1 + ... + x_{K-1} \leq 1$ and $x_K = 1 - x_1 - ... - x_{K_1}$.

The beta function $B(\alpha)$ is a normalizing constant defined in terms of the Γ function as follows:

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\prod_{i=1}^{K} \alpha_i)}$$

The parameters of the two Dirichlet mixtures used in LDA play an important role as they control the mean and the variance of the marginals X_i:

$$E[X_i] = \frac{\alpha_i}{\sum_{i=1}^{K} \alpha_k}$$

$$Var[X_i] = \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)}$$

Let us consider the Dirichlet random variable $X = (X_1, ..., X_K) \sim Dir(\alpha)$ with $K = 3$. The following figures exemplify the probability density function for a 3-dimensional Dirichlet distribution with different α parameters.

If the α parameter is symmetric ($\alpha_1 = \alpha_2 = \alpha_3$) we obtain a figure such as A.1. This plots the probability density function of the Dirichlet distribution $f(x_1, x_2, x_3; 5, 5, 5)$ for all $x_1, x_2, x_3 > 0$ and $\sum_i X_i = 1$. The domain is a plane because $\sum_i X_i = 1$. As it can be seen, since the α parameter is symmetric, the most probable mixtures are those centered around the mean, for which each topic is equally likely. In A.2 the α parameter is non-symmetric, $\alpha = (4, 3, 2)$, and mixtures favoring the first topic over the second, and the second topic over the third, are more likely.

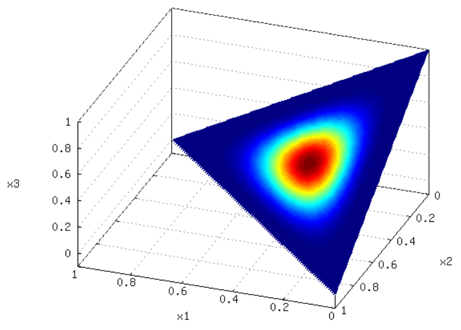


Figure A.1: $\alpha = (5, 5, 5)$

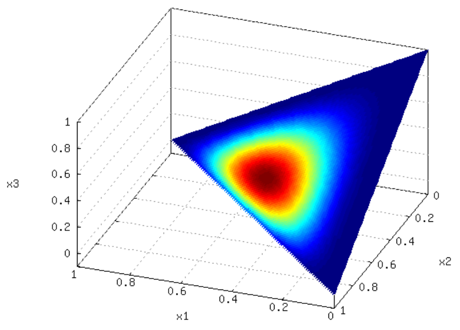


Figure A.2: $\alpha = (4, 3, 2)$

The actual vales of the α parameters control the variance. The relatively large values $\alpha = (5, 5, 5)$ determine low variance, meaning that the probability of mixtures very close to the mean is very high. With smaller α values such as $\alpha = (2, 2, 2)$ in Figure A.3, the probability of mixtures close to the mean decreases. In the Dirichlet(1,1,1) distribution plotted in Figure A.4, all mixtures are equally probable.

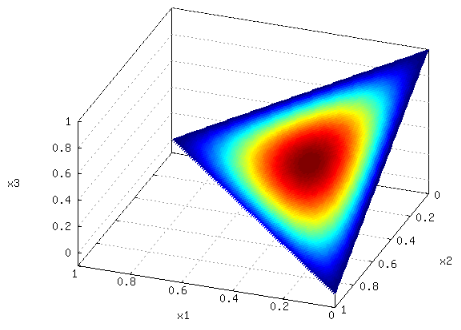


Figure A.3: $\alpha = (2, 2, 2)$

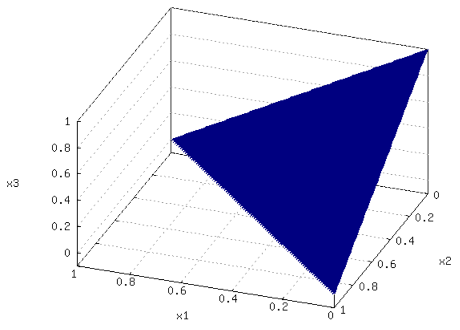


Figure A.4: $\alpha = (1, 1, 1)$

Table of Contents

Bibliography

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27, Morristown, NJ, USA, 2009.

Enrique Alfonseca, Keith Hall, and Silvana Hartmann. Large-scale computation of distributional similarities for queries. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 29–32, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Srinivas S Aswani Kumar, Ch. A note on the effect of term weighting on selecting intrinsic dimensionality of data. *Cybernetics and Information Technologies*, 9:5–12, 2009.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. *Proceedings of the 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90, 1998.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Roy Bar-Haim, Ido Dagan, Iddo Greental, Idan Szpektor, and Moshe Friedman. Semantic inference at the lexical-syntactic level for textual entailment recognition. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 131–136, Prague, June 2007.

Marco Baroni and Alessandro Lenci. One distributional memory, many semantic spaces. In *In Proceedings of GEometrical Models of Natural Language Semantics Worskshop, EACL 2009*, Athens, Greece, 2009.

Regina Barzilay and Kathleen R. McKeown. Extracting paraphrases from a parallel corpus. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 50–57, Toulouse, France, July 2001. Association for Computational Linguistics.

Roberto Basili, Diego De Cao, Paolo Marocco, and Marco Pennacchiotti. Learning selectional preferences for entailment or paraphrasing rules. In *In Proceedings of RANLP 2007*, Borovets, Bulgaria, 2007.

Holger Bast and Debapriyo Majumdar. Why spectral retrieval works. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 11–18, 2005.

Rahul Bhagat, Patrick Pantel, and Eduard Hovy. LEDIR: An unsupervised algorithm for learning directionality of inference rules. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 161–170, Prague, Czech Republic, June 2007.

D. Blei and M. Jordan. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 127–134. ACM Press, August 2003.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. ISSN 1532-4435.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. A topic model for word sense disambiguation. In *Empirical Methods in Natural Language Processing*, 2007.

Samuel Brody and Mirella Lapata. Bayesian word sense induction. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–111, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Jean-Philippe Brunet, Pablo Tamayo, Todd R. Golub, and Jill P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *PNAS*, 101(12):4164–4169, 2004.

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. Nus-ml:improving word sense disambiguation using topic features. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 249–252, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Monica Chagoyen, Pedro Saez, Hagit Shatkay, Jose Carazo, and Alberto Montano. Discovering semantic features in the literature: a foundation for building functional associations. *BMC Bioinformatics*, 7(1):41+, January 2006.

Gzegorz Chrupala, Georgiana Dinu, and Benjamin Roth. Enriched syntax-based meaning representation for answer extraction. In *SIGIR 2010 Workshop: Query Representation and Understanding*, 2010. DK, MP.

Peter Clark, Phil Harrison, John Thompson, William Murray, Jerry Hobbs, and Christiane Fellbaum. On the role of lexical and world knowledge in rte3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 54–59, Prague, June 2007. Association for Computational Linguistics.

S. Clark, B. Coecke, and M. Sadrzadeh. A distributional compositional model of meaning. *QI, College Publications, University of Oxford, March 2008.*, 2008.

Stephen Clark and Stephen Pulman. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction, Stanford, CA, 2007*, pages 52–55, 2007.

Michael Connor and Dan Roth. Context sensitive paraphrasing with a global unsupervised classifier. In *ECML '07: Proceedings of the 18th European Conference on Machine Learning*, pages 104–115, Berlin, Heidelberg, 2007. Springer-Verlag.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings 28th annual international ACM SIGIR conference*, pages 400–407, 2005.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Lecture Notes in Computer Science, Vol. 3944, Springer*, pages 177–190. Quionero-Candela, J.; Dagan, I.; Magnini, B.; d'Alch-Buc, F. Machine Learning Challenges, 2006.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927, April 2008.

Georgiana Dinu and Mirella Lapata. Topic models for meaning similarity in context. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 250–258, Stroudsburg, PA, USA, 2010a. Association for Computational Linguistics.

Georgiana Dinu and Mirella Lapata. Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, Cambridge, MA, October 2010b. Association for Computational Linguistics.

Georgiana Dinu and Rui Wang. Inference rules and their application to recognizing textual entailment. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 211–219, Athens, Greece, March 2009. Association for Computational Linguistics.

Katrin Erk. Supporting inferences in semantic space: representing words as regions. In *IWCS-8 '09: Proceedings of the Eighth International Conference on Computational Semantics*, pages 104–115, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Katrin Erk and Sabastian Padó. A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*, Waikiki, Honolulu, Hawaii, 2008.

Katrin Erk and Sebastian Padó. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, Athens, Greece, 2009.

Katrin Erk and Sebastian Padó. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 92–97, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998.

Charles J. Fillmore. Frame semantics. In *Cognitive linguistics: basic readings*, pages 373–400. Walter de Gruyter, 1982.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, 2002.

Yuan Gao and George Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21): 3970–3975, 2005.

Eric Gaussier and Cyril Goutte. Relation between plsa and nmf and implications. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602, New York, NY, USA, 2005.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 1–9, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

W. R. Gilks. *Markov Chain Monte Carlo In Practice.* Chapman and Hall/CRC, 1999.

David Graff, J. Kong, K. Chen, and K. Maeda. English gigaword. Linguistic Data Consortium, Philadelphia, 2003.

Mark A. Greenwood. Using pertainyms to improve passage retrieval for questions requesting information about a location. In *Proceedings of the Workshop on Information Retrieval for Question Answering (SIGIR 2004)*, Sheffield, UK, 2004.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. Concrete sentence spaces for compositional distributional models of meaning. *Proceedings of the 9th International Conference on Computational Semantics (IWCS11)*, pages 125–134, 2011.

Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA, USA, 1994. ISBN 0792394682.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

Patrick Hanks. Do word meanings exist? *Computers and the Humanities*, 34 (8):3913–3927, April 2000.

Patrick Hanks. How people use words to make meanings. In Bernadette Sharp and Michael Zock, editors, *NLPCS*, pages 3–13. SciTePress, 2010.

Gregor Heinrich. Parameter estimation for text analysis. Technical note version 2 (1: 2005), vsonix GmbH and University of Leipzig, February 2008.

Thomas Hofmann. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI99*, pages 289–296, 1999.

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004. ISSN 1046-8188.

Thomas Hofmann and Jan Puzicha. Unsupervised learning from dyadic data. In *Technical Report, ICSI TR-98-042*, 1989.

Thomas Hofmann, Jan Puzicha, and Michael I. Jordan. Learning from dyadic data. In *Advances in Neural Information Processing Systems 11, [NIPS Conference, Denver, Colorado, USA, November 30 - December 5, 1998]*, pages 466–472, 1998.

Eduard H. Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in webclopedia. In *TREC*, 2000.

Parry Husbands, Horst Simon, and Chris Ding. Term norm distribution and its effects on latent semantic indexing. *Journal of Information Processing and Management*, 41:777–787, July 2005. ISSN 0306-4573.

Adrian Iftene and Alexandra Balahur-Dobrescu. Hypothesis transformation and semantic variability rules used in recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 125–130, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

Michael Kaisser. *Acquiring Syntactic and Semantic Transformations in Question Answering*. 2009.

Michael Kaisser and John Lowe. Creating a research collection of question answer sentence pairs with amazons mechanical turk. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). ISBN 2-9517408-4-0. http://www.lrec-conf.org/proceedings/lrec2008/.

Michael Kaisser and Bonnie Webber. Question answering based on semantic roles. In *DeepLP '07: Proceedings of the Workshop on Deep Linguistic Processing*, pages 41–48, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

Walter Kintsch. Predication. *Cognitive Science*, 25(2):173–202, 2001.

Karin Kipper Schuler, Anna Korhonen, and Susan Brown. Verbnet overview, extensions, mappings and applications. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 13–14, Boulder, Colorado, May 2009. Association for Computational Linguistics.

Kazuaki Kishida. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. *NII Technical Report*, 2005.

D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 2003.

Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999. ISSN 0028-0836.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

Fei-Fei Li, Pietro Perona, and California Institute of Technology. A bayesian hierarchical model for learning natural scene categories. In *CVPR (2)*, pages 524–531. IEEE Computer Society, 2005. ISBN 0-7695-2372-2.

Stan Z. Li, XinWen Hou, HongJiang Zhang, and QianSheng Cheng. Learning spatially localized, parts-based representation. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*, pages 207–212, 2001.

Dekang Lin. An information-theoretic definition of similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998a. Morgan Kaufmann Publishers Inc.

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada, August 1998b. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001a. ISSN 1351-3249.

Dekang Lin and Patrick Pantel. DIRT – Discovery of Inference Rules from Text. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*, San Francisco, CA, 2001b.

Erwin Marsi, Emiel Krahmer, and Wauter Bosma. Dependency-based paraphrasing for recognizing textual entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 83–88, Prague, June 2007. Association for Computational Linguistics.

Diana McCarthy and Roberto Navigli. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 48–53, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, 2008.

Jeff Mitchell and Mirella Lapata. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 430–439, Singapore, 2009.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 2010. To appear.

Srini Narayanan and Sanda Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004.

David Novakovitch, Peter Bruza, and Laurianne Sitbon. Inducing shades of meaning by matrix methods: A first step towards thematic analysis of opinion. In *Proceedings of the 2009 Third International Conference on Advances in Semantic Processing*, pages 86–91, Washington, DC, USA, 2009. IEEE Computer Society.

Sebastian Pado and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106, 2005. ISSN 0891-2017.

Bo Pang, Kevin Knight, and Daniel Marcu. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *HLT-NAACL*, pages 102–109, 2003.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. ISP: Learning inferential selectional preferences. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, New York, 2007.

Marius Pasca and Sanda M. Harabagiu. The informative role of WordNet in open-domain question answering. In *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, pages 138–143, 2001a.

Marius A. Pasca and Sandra M. Harabagiu. High performance question/answering. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 366–374, New York, NY, USA, 2001b. ACM.

Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM New York, NY, USA, 1998.

Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore, August 2009. Association for Computational Linguistics.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577, New York, NY, USA, 2008. ISBN 978-1-60558-193-4.

J.K. Pritchard, M. Stephens, and P. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945, 2000.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. Mapping dependencies trees: An application to question answering. In *In Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics, Fort*, 2004.

Joseph Reisinger and Raymond J. Mooney. Multi-prototype vector-space models of word meaning. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2010)*, pages 109–117, 2010.

Philip Resnik. Wsd in nlp applications. In *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*, pages 299–338. Springer, Dordrecht, The Netherlands, 2006.

Stefan Riezler and Yi Liu. Query rewriting using monolingual statistical machine translation. *Computational Linguistics*, 2010. doi: 10.1162/coli_a_00010.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu O. Mittal, and Yi Liu. Statistical machine translation for query expansion in answer retrieval. In *ACL*. The Association for Computer Linguistics, 2007.

Bryan C. Russell, William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR (2)*, pages 1605–1614. IEEE Computer Society, 2006. ISBN 0-7695-2597-0.

G Salton, A Wang, and C Yang. A vector-space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620, 1975.

Gerard Salton. *The SMART retrieval system - experiments in automatic document processing.* Prentice-Hall, 1971.

Hinrich Schuetze. Automatic word sense discrimination. *Journal of Computational Linguistics*, 24:97–123, 1998.

Satoshi Sekine. Automatic paraphrase discovery based on context and keywords between NE pairs. In *Proceedings of International Workshop on Paraphrase*, pages 80–87, Jeju Island, Korea, 2005.

Dan Shen and Dietrich Klakow. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings COLING/ACL 2006*, pages 889–896, Sydney, 2006.

Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, 2007.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. Instance-based evaluation of entailment rule acquisition. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 456–463, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Idan Szpektor, Ido Dagan, Roy Bar-Haim, and Jacob Goldberger. Contextual preferences. In *Proceedings of ACL-08: HLT*, pages 683–691, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. Ranking paraphrases in context. In *Proceedings of TextInfer ACL 2009*, 2009.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 948–957, Morristown, NJ, USA, 2010. Association for Computational Linguistics.

Kristina Toutanova and Mark Johnson. A bayesian lda-based model for semi-supervised part-of-speech tagging. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1521–1528. MIT Press, Cambridge, MA, 2008.

Peter D. Turney. Similarity of semantic relations. *Comput. Linguist.*, 32:379–416, September 2006. ISSN 0891-2017.

Peter D. Turney and Michael L. Littman. Corpus-based learning of analogies and semantic relations. *Mach. Learn.*, 60:251–278, September 2005. ISSN 0885-6125.

Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141, 2010.

Tim Van de Cruys. Using three way data for word sense discrimination. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 929–936, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

Ellen M. Voorhees. Overview of the TREC 2002 Question Answering Track. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of TREC-2002, 11th Text Retrieval Conference*, Gaithersburg, US, 2002. National Institute of Standards and Technology, Gaithersburg, US.

Dominic Widdows. Semantic vector products: Some initial investigations. *Second AAAI Symposium on Quantum Interaction*, 2008.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA, 2000.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):214, 2004.

Maayan Zhitomirsky-Geffet and Ido Dagan. Bootstrapping distributional feature vector quality. *Computational Linguistics*, 35:435–461, September 2009. ISSN 0891-2017.