

Automatic Extraction of Facts, Relations, and Entities for Web-Scale Knowledge Base Population

Ndapandula T. Nakashole

A dissertation presented for the degree of
Doctor of Engineering
in the
Faculty of Natural Sciences and Technology
UNIVERSITY OF SAARLAND

MAX PLANCK INSTITUTE FOR INFORMATICS
· Saarbücken, Germany 2012 ·

Dean: Prof. Mark Groves
Faculty of Natural Sciences and Technology
Saarland University
Saarbücken, Germany

Colloquium: 2012-12-20
Max Planck Institute for Informatics
Saarbücken, Germany

Advisor and
First Reviewer: Prof. Gerhard Weikum
Department of Databases and Information Systems
Max Planck Institute for Informatics
Saarbücken, Germany

Second Reviewer: Dr. Fabian Suchanek
Department of Databases and Information Systems
Max Planck Institute for Informatics
Saarbücken, Germany

Third Reviewer: Prof. Tom M. Mitchell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA

Research Assistant: Dr. Rainer Gemulla
Department of Databases and Information Systems
Max Planck Institute for Informatics
Saarbücken, Germany

Chairman: Prof. Jens Dittrich
Department of Computer Science
Saarland University
Saarbücken, Germany

Declaration

I hereby solemnly declare that this work was created on my own, using only the resources and tools mentioned. Information taken from other sources or indirectly adopted data and concepts are explicitly acknowledged with references to the respective sources. This work has not been submitted in a process for obtaining an academic degree elsewhere in the same or in similar form.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbücken, December 2012

Ndapandula T. Nakashole

ABSTRACT

Equipping machines with knowledge, through the construction of machine-readable knowledge bases, presents a key asset for semantic search, machine translation, question answering, and other formidable challenges in artificial intelligence. However, human knowledge predominantly resides in books and other natural language text forms. This means that knowledge bases must be extracted and synthesized from natural language text. When the source of text is the Web, extraction methods must cope with ambiguity, noise, scale, and updates.

The goal of this dissertation is to develop knowledge base population methods that address the aforementioned characteristics of Web text. The dissertation makes three contributions. The first contribution is a method for mining high-quality facts at scale, through distributed constraint reasoning and a pattern representation model that is robust against noisy patterns. The second contribution is a method for mining a large comprehensive collection of relation types beyond those commonly found in existing knowledge bases. The third contribution is a method for extracting facts from dynamic Web sources such as news articles and social media where one of the key challenges is the constant emergence of new entities.

All methods have been evaluated through experiments involving Web-scale text collections.

KURZFASSUNG

Maschinenlesbare Wissensbasen sind ein zentraler Baustein für semantische Suche, maschinelles Übersetzen, automatisches Beantworten von Fragen und andere komplexe Fragestellungen der Künstlichen Intelligenz. Allerdings findet man menschliches Wissen bis dato überwiegend in Büchern und anderen natürlichsprachigen Texten. Das hat zur Folge, dass Wissensbasen durch automatische Extraktion aus Texten erstellt werden müssen. Bei Texten aus dem Web müssen Extraktionsmethoden mit einem hohen Maß an Mehrdeutigkeit und Rauschen sowie mit sehr großen Datenvolumina und häufiger Aktualisierung zurechtkommen.

Das Ziel dieser Dissertation ist, Methoden zu entwickeln, die die automatische Erstellung von Wissensbasen unter den zuvor genannten Unwägbarkeiten von Texten aus dem Web ermöglichen. Die Dissertation leistet dazu drei Beiträge. Der erste Beitrag ist ein skalierbar verteiltes Verfahren, das die effiziente Extraktion hochwertiger Fakten unterstützt, indem logische Inferenzen mit robuster Textmustererkennung kombiniert werden. Der zweite Beitrag der Arbeit ist eine Methodik zur automatischen Konstruktion einer umfassenden Sammlung typisierter Relationen, die weit über die in existierenden Wissensbasen bekannten Relationen hinausgeht. Der dritte Beitrag ist ein neuartiges Verfahren zur Extraktion von Fakten aus dynamischen Webinhalten wie Nachrichtenartikeln und sozialen Medien. Insbesondere werden Lösungen vorgestellt zur Erkennung und Registrierung neuer Entitäten, die bislang in keiner Wissensbasis enthalten sind.

Alle Verfahren wurden durch umfassende Experimente auf großen Text- und Webkorpora evaluiert.

SUMMARY

Knowledge bases are data resources that encode knowledge in machine-readable formats. Such knowledge bases are enabling progress in semantic search, machine translation, question answering and other formidable challenges in artificial intelligence. Knowledge bases are populated by means of information extraction techniques. This dissertation contributes to three aspects of information extraction: fact extraction, relation extraction, and entity extraction.

Fact extraction is central to knowledge base construction. It entails finding *subject-predicate-object* triples in text. State-of-the-art approaches combine pattern-based gathering of fact candidates with logical reasoning over feasibility constraints. However, they face challenges regarding the trade-offs between precision, recall, and scalability. Techniques that scale well are susceptible to noisy patterns that degrade precision, while techniques that employ deep reasoning for high precision cannot cope with Web-scale data. The approach presented in this dissertation aims at a scalable system for high-quality fact extraction. For high recall, a new notion of ngram-itemsets for richer patterns is introduced. For high precision, the method makes use of MaxSat-based constraint reasoning on both the quality of patterns and the validity of fact candidates. For scalability, all parts of the approach are developed as distributed algorithms in the form of MapReduce algorithms. Collectively, these components form PROSPERA, a system which reconciles high recall, high precision, and scalability.

Relation extraction identifies phrases that denote binary relations which are deemed interesting for fact extraction. Relations in existing knowledge bases are mostly hand-specified and limited to around 1000 or less. On the other hand, systems that yield a large number of relational textual patterns do no attempt to organize these according in a meaningful scheme. This dissertation presents PATTY, a method for extracting textual patterns that denote binary relations. In a departure from prior work, the patterns are semantically typed and organized into synonym sets and in a subsumption taxonomy. In particular, the method defines an expressive family of relational patterns, which combines syntactic features (S), ontological type signatures (O), and lexical features (L). The novelty is the addition of the ontological (semantic) dimension to patterns. PATTY also comes with efficient and scalable algorithms that can infer patterns and subsumptions at scale, based on instance-level overlaps and an ontological type hierarchy. On the Wikipedia corpus, PATTY produced 350,569 pattern synsets with 84.7% precision.

Entity extraction identifies noun phrases in text that define entities. Large-scale knowledge bases are often restricted to popular entities such as famous people and locations that are found in encyclopedias such as Wikipedia. Therefore, knowledge bases are never complete in the entities they contain. However, when extracting facts from rapidly updated data sources, such as news articles and social media, new entities emerge all the time. Simply considering all noun phrases as entities would introduce noise and make it difficult to relationships between new entities. To overcome this limitation, this dissertation presents PEARL, a method for enhancing previously unseen entities with type information. PEARL introduces an ILP-based method for typing out-of-knowledge-base entities that leverages type disjointness constraints. Typing new entities enables the fact extractor to generate more accurate facts. The value of extracting facts from rapidly updated sources is shown through the application of identifying emerging stories. Prior work on story identification generate stories as clusters of related entities, without specifying how the entities are related. In contrast, the method presented here succinctly states how a set of entities in a story are semantically related.

ZUSAMMENFASSUNG

Wissensbasen sind Datenquellen, die Wissen in maschinenlesbarer Form darstellen. Sie unterstützen damit Entwicklungen im Bereich der semantischen Suche, des maschinellen Übersetzen, des automatischen Beantwortens von Fragen und ähnlich komplexer Fragestellungen der Künstlichen Intelligenz. Die Erstellung von Wissensbasen erfolgt mit Techniken der Informationsextraktion. Diese Dissertation trägt zu den folgenden drei Aspekten der Informationsextraktion bei: Extraktion von Fakten, von Relationen und von Entitäten.

Die **Extraktion von Fakten** ist eine zentrale Aufgabe bei der Erstellung von Wissensbasen. Dies beinhaltet das Auffinden von Subjekt-Prädikat-Objekt-Tripeln in Texten. Neuere Verfahren kombinieren dazu musterbasierte Methoden zum Auffinden von potentiellen Fakten mit logischen Inferenzmethoden zur Sicherstellung von Konsistenzbedingungen. Allerdings existieren hier Zielkonflikte bezüglich Güte, Auffindbarkeit und Skalierbarkeit. Einerseits sind Verfahren, die sich durch eine gute Skalierbarkeit auszeichnen, anfällig gegen Rauschen, was wiederum die Güte negativ beeinflusst. Andererseits skalieren Methoden zumeist nicht, wenn sie Konsistenzbedingungen durch logisches Schließen berücksichtigen. Der in dieser Arbeit vorgestellte Ansatz zielt darauf ab, hochwertige Fakten zu extrahieren. Um gute Auffindbarkeit zu gewährleisten, wird ein neuer Ansatz unter dem Einsatz semantisch bedeutungsvollerer Muster auf Grundlage von N-Grammen eingeführt. Zum Erzielen einer hohen Güte wird ein logisches Inferenzverfahren auf der Basis eines MaxSat-Approximationsalgorithmus verwendet, das sowohl die Qualität der Muster als auch die Gültigkeit der Fakten bewertet. Die Implementierung in Form verteilter Algorithmen auf einer MapReduce-Plattform skaliert sehr gut. Die Kombination dieser Verfahren bildet das PROSPERA-System, das sich durch ein hohes Maß an Auffindbarkeit, Güte und Skalierbarkeit auszeichnet.

Bei der **Relationsextraktion** werden zunächst Phrasen identifiziert, die als binäre Relationen für die Extraktion von Fakten interpretiert werden können. In existierenden Wissensbasen sind diese Relationen in der Regel manuell spezifiziert worden; daher gibt es dort typischerweise nur 1000 oder weniger Relationen. Auf der anderen Seite gibt es Systeme, die automatisch auf eine große Zahl relationaler Muster abzielen. Diese sind jedoch nicht in der Lage, die gesammelten Muster in semantischer Weise zu organisieren. Diese Dissertation präsentiert die PATTY-Methode zur Extraktion und semantischen Organisation textueller Muster, die binäre Relationen darstellen. Abweichend von früheren Arbeiten sind die hier verwendeten Muster semantisch typisiert und sowohl in Synonymgruppen als

auch in einer hierarchischen Taxonomie eingeordnet. Der hier vorgestellte Ansatz bietet damit ausdrucksstarke relationale Muster, die syntaktische Eigenschaften (S), ontologiebasierte Typsignaturen (O) und lexikalische Charakteristika (L) zu SOL-Mustern kombinieren. Neuartig an PATTY ist vor allem die Dimension einer ontologiebasierten (semantischen) Komponente für relationale Muster. Die Algorithmen zur Konstruktion der Relationen und ihrer Synonymgruppen sind hochgradig skalierend und basieren auf Überlappungsmaßen zwischen Instanzmengen. Auf dem Volltext von Wikipedia erzeugte PATTY 350,569 relationale Synonymgruppen bei einer Güte von 84,7%.

Zur **Entitätenextraktion** werden zunächst Nominalphrasen in Texten identifiziert, die Entitäten beinhalten. Große Wissensbasen sind zumeist auf populäre Entitäten wie berühmte Persönlichkeiten, Orte und Organisationen eingeschränkt, wie sie zum Beispiel in Wikipedia vorkommen. Daher sind diese Wissensbasen inhärent unvollständig in Bezug auf die enthaltenen Entitäten. Wenn man Fakten aus häufig aktualisierten Webinhalten wie Nachrichtenartikeln und sozialen Medien extrahiert, stößt man ständig auf neue Entitäten. Ein denkbarer Ansatz, dem entgegenzuwirken, wäre, jede neu auftauchende Nominalphrase als Entität zu interpretieren, was jedoch semantisch unbefriedigend ist. Um dieses Dilemma aufzulösen, entwickelt diese Dissertation die PEARL-Methode, die bis dato unbekannte Entitäten mit Typinformation anreichert. PEARL basiert auf einer ganzzahligen linearen Optimierung, die u.a. die Disjunktheit bestimmter Typen berücksichtigt. Durch das zeitnahe Extrahieren neuer Entitäten in sich schnell ändernden Datenquellen werden neue Anwendungsfelder erschlossen, beispielsweise das automatische Zusammenstellen von Stories aus einer Fülle von Einzelnachrichten. Frühere Arbeiten, die sich mit Story-Konstruktion befassen, haben einfach häufig zusammen auftretende Entitäten gruppiert, ohne die Relationen zwischen den Entitäten zu interpretieren. Im Gegensatz dazu stellt das hier vorgestellte Verfahren den semantischen Zusammenhang einer Gruppe von Entitäten in einer Story her.

ACKNOWLEDGEMENTS

This work has been made possible through the outstanding guidance and support of my advisor Prof. Gerhard Weikum. I am fortunate and grateful to have had the opportunity to work with him. Members of the Databases and Information Systems group provided a stimulating working environment, I thank them. I thank my co-authors Martin Theobald and Fabian Suchanek with whom I collaborated on parts of this work. I thank Marc Spaniol for translating the abstract and summary to German. I acknowledge with gratitude, financial assistance from the International Max Planck Research School for Computer Science (IMPRS-CS). Last but not least, I express gratitude to my parents, David and Secilia Nakashole, who have always encouraged me.



Table of Contents

1	Introduction	19
1.1	State-of-the-Art	20
1.2	Contributions	22
1.3	Dissertation Organization	23
2	Background, Preliminaries, & Related Work	26
2.1	Background	26
2.1.1	Template Filling	26
2.1.2	Message Understanding Conferences (MUC)	27
2.1.3	Pattern Learning	28
2.1.4	Open Relation Discovery	28
2.1.5	Exploiting Semi-Structured Sources	29
2.1.6	The Semantic Web	30
2.2	Knowledge Base Preliminaries	31
2.2.1	Knowledge Base	31
2.2.2	Ontology	31
2.2.3	Classes	31
2.2.4	Entities	32
2.2.5	Relations	32
2.3	Storage and Access Management	32
2.3.1	Relational Databases	32
2.3.2	NoSQL Databases	33
2.3.3	Triplestores	33
2.4	Related Work	34
2.4.1	SOFIE	34
2.4.2	ReadTheWeb/NELL	35
2.4.3	TextRunner/Reverb	35
2.4.4	Probase	36
2.4.5	WebTables	36
2.4.6	YAGO, DBpedia, Freebase	37
2.5	Summary	37

3	Fact Extraction with PROSPERA	39
3.1	Motivation	39
3.2	Contribution	42
3.3	Related Work	43
3.4	Overview of PROSPERA	46
3.5	Pattern Analysis	47
3.6	Reasoning	52
3.7	Distributed Implementation	54
3.8	Evaluation	58
3.9	Summary	68
4	Relation Extraction with PATTY	69
4.1	Motivation	69
4.2	Contribution	70
4.3	Related Work	71
4.4	Overview of PATTY	73
4.5	Pattern Extraction	74
4.6	SOL Pattern Model	75
4.7	Lexico-Syntactic Pattern Generalization	77
4.8	Semantic Pattern Generalization	78
4.9	Taxonomy Construction	78
4.10	Evaluation	84
4.11	Applications	88
4.12	Summary	92
5	Entity Extraction with PEARL	95
5.1	Motivation	95
5.2	Contribution	98
5.3	Related Work	98
5.4	Overview of PEARL	100
5.5	Detection of New Entities	101
5.6	Typing Out-of-Knowledge-Base Entities	102
5.7	Emerging Story Mining	108
5.8	Evaluation	112
5.9	Summary	123
6	Conclusion	125
6.1	Contributions	125
6.2	Future Directions	126
A	PATTY vs. Other Resources	129
B	PEARL vs. Baselines	135
	Bibliography	141



List of Tables

3.1	Example seed patterns with computed confidence values	50
3.2	Performance for the <i>hasAcademicAdvisor</i> relation	59
3.3	Performance for all relations	60
3.4	Precision and recall for the <i>hasAcademicAdvisor</i> relation for varying numbers of seeds	61
3.5	Performance comparison between PROSPERA and NELL on sports relations: number of extractions	62
3.6	Performance comparison between PROSPERA and NELL on sports relations : precision	63
3.7	Performance of PROSPERA: precisions@1000	63
3.8	Sample output from PROSPERA	64
3.9	Sample output from NELL	64
3.10	PROSPERA variants on sports relations (2 iterations).	66
3.11	Sample output from PROSPERA	67
3.12	Extracted facts and estimated precision for academic relations obtained from two iterations of PROSPERA.	67
3.13	PROSPERA variants on academic relations	67
4.1	Pattern Synsets and their Support Sets	79
4.2	Inverted data for Map-Reduce algorithm	83
4.3	Precision of Relational Patterns	85
4.4	Quality of Subsumptions	85
4.5	Coverage of Music Relations	86
4.6	Results for Different Pattern Language Alternatives	87
4.7	Sample Results for Relation Paraphrasing Precision	87
4.8	Sample Results for Relation Paraphrasing	88
5.1	Results for entity typing (NS: <i>not sure</i> , W: <i>wrong</i> , S: <i>satisfactory</i> , VG: <i>very good</i>).	113
5.2	Inter-judge agreement kappa: Fleiss' κ & adapted Cohen's κ	114

5.3	Results for entity typing by PEARL variants (NS: <i>not sure</i> , W: <i>wrong</i> , S: <i>satisfactory</i> , VG: <i>very good</i>).	115
5.4	Inter-judge agreement in entity-typing assessments.	115
5.5	NDCG for various entity-typing methods.	116
5.6	Example source sentences for PEARL results.	119
5.7	Example results from PEARL, HYENA, NNPLB	120
5.8	Sample results for typing out-of-knowledge-base entities	121
5.9	Quality assessment of story representations.	122
5.10	Precision of PEARL stories	122
5.11	A categorization of the evaluated stories.	123



List of Figures

2.1	An excerpt of a MUC-3 template filling task, adapted from [47]. The template is filled by extracting the relevant fields from a terrorist attack story.	27
2.2	An example infobox from Barbara Liskov's Wikipedia page	29
2.3	Wiki-markup corresponding to the infobox in Barbara Liskov's Wikipedia page	30
3.1	The fact extraction problem.	39
3.2	Architecture of the PROSPERA System.	48
3.3	MapReduce pseudo-code for frequent n-gram-itemset mining	55
3.4	MapReduce pseudo-code for seed pattern confidence	56
3.5	Mapper pseudo-code for pattern similarity and fact-candidate extraction.	57
3.6	PROSPERA runtimes on the sports domain (a) and number of extractions across iterations (b).	65
4.1	Architecture of the PATTY System.	73
4.2	An example type system.	74
4.3	Prefix-tree for the Synsets in Table 4.1.	79
4.4	An example type-segmented tree.	82
4.5	PATTY paraphrases for the YAGO relation actedIn	89
4.6	A part of the PATTY taxonomy	90
4.7	Schema-free search for costars of Brad Pitt	91
4.8	Simple join query to find visionaries affiliated with MIT	92
4.9	Explaining relatedness through a direct triple	93
4.10	Explaining relatedness through a join-query	94
5.1	Out-of-knowledge-base entities per day	95
5.2	Example of output from dynamic sources	97
5.3	Architectural overview of the PEARL approach	100

5.4	A fact graph depicting a single story	110
5.5	Precision of entity typing methods at 0.4 minimum fuzzy match score.	117
5.6	Precision of entity typing methods at 0.8 minimum fuzzy match score.	118
5.7	Relative recall for varying degrees of minimum fuzzy matches.	118

Introduction

BUILDING computers with human-level intelligence has been a long-standing goal in Artificial Intelligence. Alan Turing posed a philosophical test for assessing a machine's ability to exhibit intelligent behavior [111]. A machine passes the Turing test if it has the ability to carry on a conversation, in natural language, well enough to be indistinguishable from a person. It is clear that today's computers have not reached that kind of intelligence. Researchers in Natural Language Processing (NLP) have instead focused on specific sub-challenges, including semantic search, question answering, machine translation, and summarization. Literature from these areas shows that knowledge plays a critical role in building intelligent systems [19, 25, 35, 40, 44, 89, 108, 116]. Therefore, it is important to capture knowledge and store it in machine-readable format — this is the goal of knowledge base construction.

With much of human knowledge residing in books and other text documents, large-scale knowledge bases have to be constructed by extracting and synthesizing information from natural language text. Therefore, knowledge base construction requires methods for natural language text understanding. Machine understanding of text refers to parsing and transforming unstructured text into a structured representation. Such a representation has no ambiguity in it — making it suitable for machine reading and machine interpretation.

Encoding text as machine-readable knowledge requires transforming syntactic constructs to semantic constructs. One such transformation maps names in text to real world entities. For example, the word *Apple* may refer to the company *Apple Inc.* or to the apple fruit. Similarly, the name *Madonna* may refer to the popular singer or the Christian painting. Mapping names to entities requires methods for entity disambiguation and co-reference resolution. Another transformation towards semantic data identifies relationships between pairs of entities. For example, simply stating that *Apple Inc* co-occurs a lot with *Cupertino, California* does not convey any semantics about the nature of the relationship. Instead, the goal is to explicitly state that *Apple isHeadquarteredInLocation Cupertino, California*. This requires methods for fact and relation extraction. The last transformation towards knowledge entails shifting from a document-centric view of informa-

tion to a knowledge-centric view by constructing knowledge bases consisting of factual assertions. These factual assertions are entities that exist and the relationships that hold among them. By assimilating relationships and entities from several documents, a knowledge base can unveil not only direct relationships but also complex relationships through inference. For example, a knowledge base may reveal that the relationship between *Apple Inc.* and the disk drive company *Seagate Technology* is that they are both headquartered in *Curpertino*. As basic as it is for a human to uncover such a relationship, a document-centric view of the data would not be able to establish such a connection.

The above-mentioned transformations are challenging due to the inherent ambiguity of natural language. There is ambiguity in entity names and in relational phrases. Determining when a sentence mentions an entity or relation at all is not trivial. Also determining which of many relations a piece of text refers to is not simple. For example, it may not be clear if the text refers to the relation *isHeadquarteredInLocation* or the relation *hasOfficesInLocation*. The multiplicity of relations between entity pairs is also a challenge, Apple and Seagate may not only be headquartered in the same location, they might also have been led by the same executives at different time periods. Because of these factors, knowledge base construction is a best-effort process. It may produce conflicting and erroneous facts. In spite of these challenges, large-scale knowledge bases have been constructed in academia and industry. Such knowledge bases contain millions of entities and hundreds of millions of facts about them. Example academic projects include: *opencyc.org*, *dbpedia.org* [8], *knowitall* [38, 10], *stat-snowball* [134], *readtheweb* [20], and *yago-naga* [108]. Industry has produced products such as : *wolframalpha.com*, *freebase.com*, and *trueknowledge.com*.

Although these projects underline progress made to date, much remains to be done. Thus, the goal of this dissertation is to move forward the state-of-the-art in knowledge base construction research. It brings together ideas from previous work while also proposing new ideas for better coverage and quality.

1.1 STATE-OF-THE-ART

Information extraction (IE) [104] is the technology for extracting knowledge from text by transforming unstructured text into a structured format. The output of information extraction is usually facts in the form: *subject-predicate-object* triples. For example, in the triple *Apple-isHeadquarteredIn-Cupertino*, *Apple* is the *subject*, *isHeadquarteredIn* is the *predicate*, and *Cupertino* is the *object*. In extracting facts from text, there are two important questions to ask. The first is: what constitutes a subject and object? In other words, which noun phrases in text are worth extracting facts about? The second question is: which relations should be extracted? In other words, which verb phrases or other types of phrases denote relations between entities? In answering these questions, two paradigms of information extraction have emerged, each at a different end of the spectrum of the conceivable answers to these questions. The two paradigms are: *Ontology-based IE* [109] and *Open IE* [10].

In Ontology-based IE, at least one ontology is used to guide the extraction pro-

cess. The ontology serves the purpose of providing a dictionary of entities. It also defines relations of interest in a specific domain, such as books and their authors or companies and their headquarters. The relations in ontologies are generally hand-specified by either developers of the ontology [108] or by domain experts [43]. What this means for ontology-based IE is that there is a major limitation in the number of relations that can be populated. Open IE aims to address this limitation by not making assumptions about the existence of entities or relations. Instead, all noun phrases are considered to be entities. For relations, the most open form of Open IE considers any phrase appearing between pairs of entities to be a relation. While Open IE counters the recall issue, it is highly susceptible to noise [70], due to the lack of tightly enforced semantics on relations and entities. Evidently, each of these IE paradigms have their respective strengths and weaknesses as outlined in more detail next.

Ontology-based IE vs. Open IE

Ontology-based IE: Pre-specified relations. The task of specifying all relations of interest is laborious. As a result, Ontology-based IE systems have traditionally only been applied to a small number of relations, typically in the order of hundreds or less [20, 81, 109]. This leads to sparse knowledge in terms of the relations populated. Commonly covered relations are *wasBornInCity*, *wasBornOn-Date*, *isHeadquateredIn*, *isMarriedTo*, *hasWonPrize*. However there are many more interesting relations that are usually not covered.

Ontology-based IE: Dictionary of entities. Dictionaries of entities are never complete and often focus on popular entities. Entities in the long tail such as a cathedral in a small french city may not be in any of the major knowledge bases. On the other hand, Notre Dame in Paris has a high chance of being included in all major knowledge bases. Furthermore, new entities are constantly emerging, for example a new singer or a new hurricane. This and the issue of pre-specified relations point to a general coverage limitation in Ontology-based IE methods.

Open IE: Unspecified relations. Instead of using a set of pre-specified relations, Open IE systems consider every phrase appearing between a pair of entities to denote a relation. This permissiveness addresses the coverage limitation seen in Ontology-based IE. However, it introduces a substantial amount of noise. The issue is partly addressed by recent work, which has attempted to improve precision by restricting relations to specific part-of-speech tag sequences that are presumed to express true relations [41]. However, this does not improve much on the front of tighter semantics, for example, what each of the relations could mean and what constraints apply to such relations. Such semantics and constraints are key to accuracy in Ontology-based IE, for example, using type constraints to determine which types of entities are valid for a given relation [109].

Open IE: No noun phrase left behind. Open IE systems do not use pre-existing dictionaries to define entities. In Open IE systems, every noun phrase is a possible entity. This raises a number of issues. First, the system picks up noun phrases that are not proper entities, such as "John and David" or "three girls". Second, there is no attempt to determine which noun phrases refer to the same entity or what type

of entity a given noun phrase is. Therefore, it is difficult to construct a consistent knowledge base. This may, for example, result in two different birth dates for the same person. Ontology-based IE enables entity disambiguation where each entity is mapped to a unique identifier in the ontology. For example, Wikipedia-derived ontologies use a short descriptive text after ambiguous names; for Michael Jordan the Berkeley professor, the identifier is *Michael_Jordan_(computer scientist)*. Therefore the knowledge base clearly distinguishes which information is related to which Michael Jordan.

Challenges

Given the shortcomings of the two paradigms of information extraction, the approach taken in this dissertation aims for a middle-ground. The goal is to avoid the limitations of the two paradigms while reaping the benefits of both. Aiming for such a middle-ground requires addressing the following limitations:

Efficiency bottleneck at Webscale. One limitation in the state-of-the art is that high accuracy has only been attained by expensive methods that do not scale to large data collections. In order to deal with Web-scale data collections, a scalable architecture for high accuracy fact extraction is needed.

High accuracy only for pre-specified relations. Another limitation in the state-of-the art is that high accuracy has only been attained for pre-specified relations in Ontology-based IE. Overcoming this limitation means discovering and maintaining a *dynamically evolving open set of relations*. This needs to go beyond common relations such as “bornIn” or “isHeadquarteredIn”. For example, major knowledge bases lack potentially interesting relations such as “firedFrom” or “hasGoddaughter”. For completeness, there is a need to automatically discover such relations. At the same time, any approach that simply considers every phrase to be a possible relation should be avoided. In order to boost accuracy, there must be some semantics associated with the relations.

Static Knowledge. Another limitation is that previous methods have largely been snapshot-based. Facts are extracted from a snapshot of a corpus and some weeks or months later, another round of facts is extracted from a new snapshot. This process is repeated at irregular and long intervals resulting in incomplete and partly stale knowledge bases. For comprehensive coverage and to avoid staleness, rapidly updated sources such as news and social media must be brought into scope. One key challenge for this kind of data is that new entities are constantly emerging, methods need to discover new entities as they emerge. At the same time, resorting to treating every noun phrase as a valid entity without any semantic information to verify their validity should be avoided.

1.2 CONTRIBUTIONS

This dissertation presents methods that, to a large extent, address the challenges outlined above. In particular, the dissertation makes the following contributions:

PROSPERA: Fact Extraction. The first contribution of this dissertation is a system named PROSPERA, for high-quality knowledge harvesting at Web-scale. Fact extraction approaches that have achieved the highest accuracy to date [109] combine pattern-based gathering of fact candidates with constraint-based reasoning. However, such approaches face major challenges regarding the trade-offs between precision, recall, and scalability. Techniques that scale well are susceptible to noisy patterns that degrade precision, while techniques that employ deep reasoning for high precision cannot cope with Web-scale data. PROSPERA introduces a new notion of n-gram-itemsets for richer patterns, and uses MaxSat-based constraint reasoning on both the quality of patterns and the validity of fact candidates. PROSPERA incorporates these building blocks into a scalable architecture which is implemented on top of a MapReduce-based distributed platform. PROSPERA's results were presented at WebDB 2010 [79] and WSDM 2011 [80].

PATTY: Relation Extraction. The second contribution of this dissertation is a system named PATTY, for extracting patterns that denote binary relations between entities. PATTY's output is a large resource for relational phrases. Unlike open IE patterns, PATTY's patterns are semantically typed and organized into a subsumption taxonomy. The PATTY system is based on efficient algorithms for frequent itemset mining and can process Web-scale corpora. It harnesses the type system and entity population of large knowledge bases. PATTY extends the state-of-the-art on lexical resources and knowledge bases in two ways. Compared to WordNet-like resources, it provides synsets for binary-relation patterns, in an unsupervised and open-domain manner. Compared to knowledge collections like ReVerb or NELL, it provides much more precise pattern synsets with semantic typing, greatly reducing the noise in extracting relational-pattern instances. PATTY is available for download and for online browsing at <http://www.mpi-inf.mpg.de/yago-naga/patty/>. PATTY's results were presented at EMNLP 2012 [81] and VLDB 2012 [82].

PEARL: Populating Knowledge Bases in Real-time. The third contribution of this dissertation is a system called PEARL, for real-time fact extraction from rapidly updated data sources such as news and social media. PEARL focuses on one of the key challenges of dynamic data — previously unseen entities. PEARL introduces an ILP-based method for typing out-of-knowledge-base entities that leverages type disjointness constraints. Furthermore, PEARL demonstrates the value of fact extraction from news and social media through the application of emerging story identification. Prior work on story identification generate stories as clusters of related entities. The stories PEARL generates state how a set of entities in a story are related. Some parts of the PEARL approach were presented at AKBC-WEKEX 2012 [83].

1.3 DISSERTATION ORGANIZATION

The outline of the rest of this dissertation is as follows. **Chapter 2** begins by summarizing the evolution of the field of information extraction; it then defines some of the central concepts in knowledge base construction and ends with a review of related projects, building a case for the novelty of our methods by outlining the

characteristics of competing methods. **Chapter 3** presents PROSPERA, a system for large scale fact extraction; it discusses how high recall, precision and scalability are brought together in one system. **Chapter 4** presents PATTY, a system for mining a large collection of patterns that denote binary relationships between entities; the chapter presents a thorough evaluation of PATTY, comparing it to similar resources. **Chapter 5** presents the PEARL system for populating knowledge bases with rapidly update data; in particular it focuses on handling new entities as they emerge and showcases a novel application. **Chapter 6** provides concluding remarks and puts forward possible directions for future work.

Background, Preliminaries, & Related Work

Recent years have seen a growing interest in automatic knowledge base construction. However, research in information extraction, the technology for extracting knowledge from text, spans over three decades. This chapter places the contributions of this dissertation in the context of the previous body of work. It first gives a brief history of information extraction. It then introduces concepts central to knowledge base construction. The chapter ends by reviewing some of the ongoing related work.

2.1 BACKGROUND

Information extraction emerged out of the natural language processing community as a way of automatically understanding text. The past two decades have seen it become a challenge of interest to other computer science communities including databases, Web mining, and information retrieval. This section explains the progression of the field from the early days, up to where it currently stands, highlighting some of the defining works.

2.1.1 Template Filling

Early systems defined information extraction as a template filling task [28, 32, 46, 106, 133]. As in contemporary projects, these systems aimed to transform natural language text into tabular structures. Each such tabular structure, referred to as a template, consists of slots to be filled up. Schank et al. [106] described template filling in the context of story comprehension. They argued that stories follow patterns, for example, stories on corporate mergers or management successions have recurring role players. Knowing who the players are for a given story enables understanding other stories. These ideas were implemented in the FRUMP system [32] in the early 1980s. Also in the early 1980s, Cowie [28] developed a system for extracting templates about plants from wild flower guides.

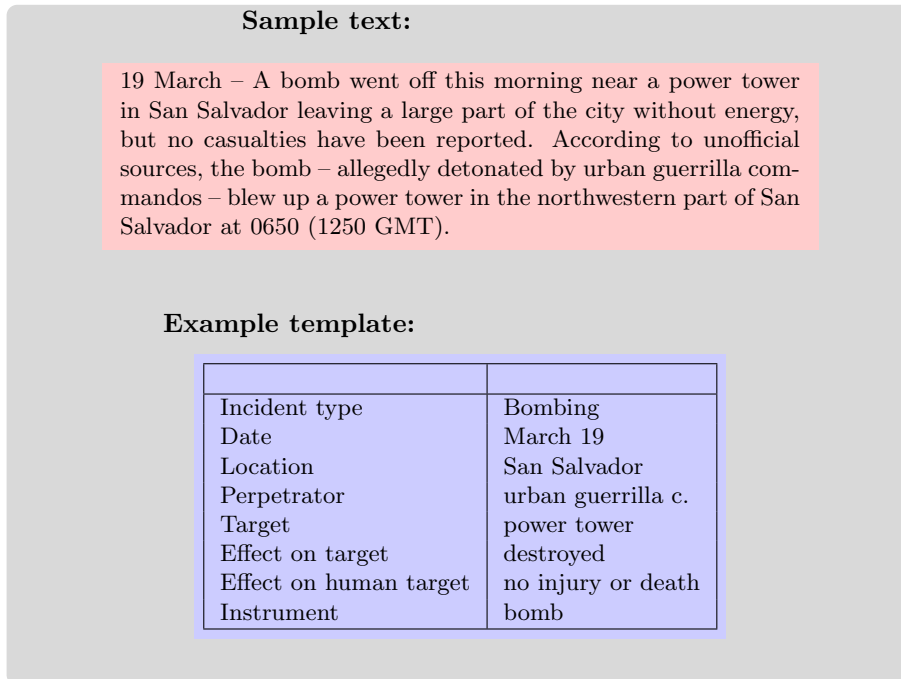


Figure 2.1: An excerpt of a MUC-3 template filling task, adapted from [47]. The template is filled by extracting the relevant fields from a terrorist attack story.

Cowie made use of pre-defined phrases which correspond to various slots of the template being extracted. In the case of plants, the slots correspond to properties of plants. Rules were then applied to match the pre-defined phrases to phrases in text in order to extract values for the slots. Zarri [133] developed a system in the early 1980s, to extract information about the roles of historical figures in French historical documents. The template slots consisted of roles in particular historical events, such as the appointment of a historical figure on a particular date at a given location.

Systems based on template filling were among the first systems in the area of information extraction.

2.1.2 Message Understanding Conferences (MUC)

The next development was the introduction of the DARPA-sponsored Message Understanding Conferences [47, 78, 110]. The MUC conferences first started with institutions in the US that were extracting information from naval messages. The idea was for systems from different institutions to work on common messages as training data and then meet at the conferences to compare performance of the different systems on unseen messages. The MUC conferences focused on extracting domain-specific information such as ship sightings, terrorism in Latin America, management successions, and business joint-venture information. Figure 2.1 is an example template filling task from MUC-3, pertaining to terrorist stories [78]. For each terrorist story, systems had to extract the type of attack

(e.g., bombing, arson, etc.), the date, location, targets, etc. Systems were comparatively assessed against human performance for the same task. Thus, the conferences provided a means of measuring performance of language understanding systems.

2.1.3 Pattern Learning

In the late 1990s, along with the rapid growth of the Web came the first systems to extract information from Web sources. Among them were the DIPRE [14] and Snowball [3] systems. These systems take as input the relations to be extracted and a few seed example pairs for each of the relations. The goal is to then build-up the initial set of example pairs by extracting more pairs from Web text through pattern learning over multiple iterations. The first step is to spot the example pairs in text and then extract the patterns they occur with as good phrases for expressing the relations. For example, seeds like

teamHasWonTrophy(Germany, FIFA_World_Cup),

which was true in 1954, 1974 and 1990, may lead, after a few iterations, to patterns such as

X won the final of Y

and may then pick up new facts such as

teamHasWonTrophy(Spain, FIFA_World_Cup),

which was true in 2010.

Pattern learning is a powerful technique which is still in use today. It can scale to large data collections. The major limitation of pattern learning systems is that they require manual specification of relations along with seed examples. This is a laborious process, requiring human supervision. While our first contribution makes use of pattern learning in PROSPERA (Chapter 3), our second contribution, PATTY (Chapter 4), shows how to avoid manual specification of relations by automatically learning relations from text [81].

2.1.4 Open Relation Discovery

In an effort to overcome the limitation of pattern learning systems, work has emerged that aims to learn relations without requiring a pre-defined set of relations and their seeds. Among these is the work on preemptive IE [127] and the TextRunner/Reverb system [10, 41]. These systems have the ability to learn a large number of relations by considering all phrases as denoting some kind of relation. However they do not provide any semantics with the phrases. This results in a substantial amount of noise [70]. More importantly, it is not clear how to apply logical reasoning on the resulting output. Logical reasoning needs some specification to reason over: without any semantics attached to the phrases such as domain and range type constraints, the value of the data is reduced, especially for logical reasoning purposes.

2.1.5 Exploiting Semi-Structured Sources


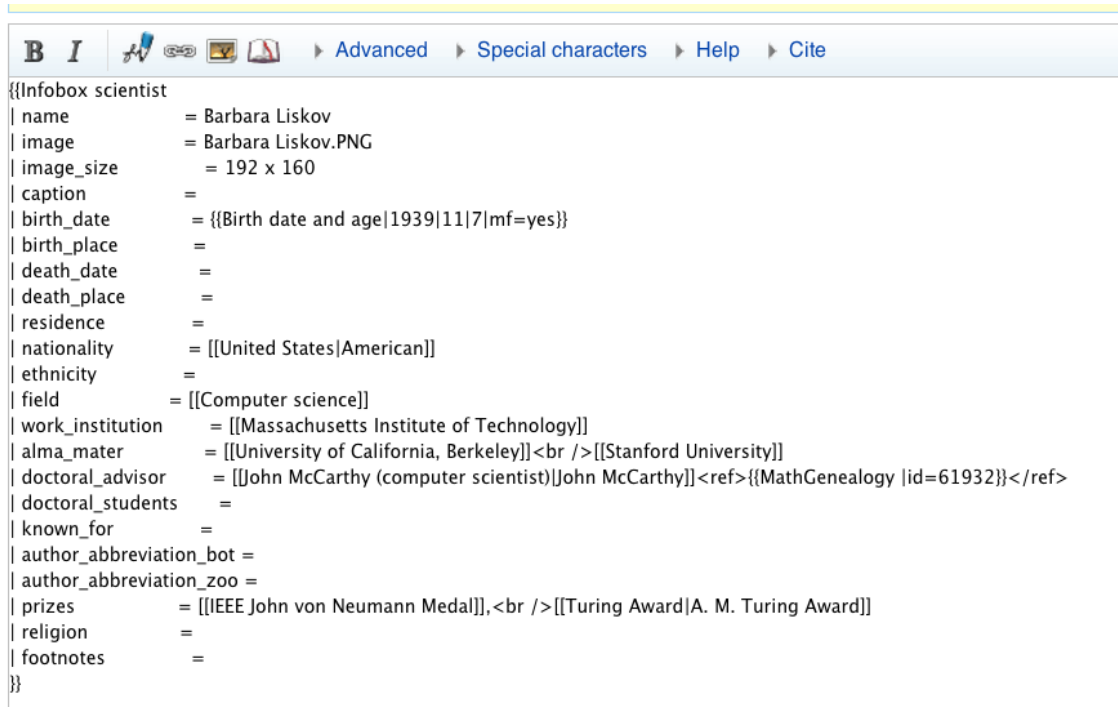
Barbara Liskov	
	
Born	November 7, 1939 (age 72)
Nationality	American
Fields	Computer science
Institutions	Massachusetts Institute of Technology
Alma mater	University of California, Berkeley Stanford University
Doctoral advisor	John McCarthy ^[1]
Notable awards	IEEE John von Neumann Medal, A. M. Turing Award

Figure 2.2: An example infobox from Barbara Liskov’s Wikipedia page

The Internet has made it possible for geographically remote humans to collaborate on massive-scale projects. This phenomenon has led to the development and growth of knowledge sharing communities such as Wikipedia and a plethora of online forums including question answering websites such as *answers.yahoo.com* and *quora.com*. The semi-structured nature of these sources means that knowledge can be extracted with relatively higher precision compared to natural language text. This has been shown by projects such as DBpedia [8], YAGO [108], and Freebase [13]. These projects have transformed Wikipedia into knowledge bases of structured data. The extractors are based on rules that extract relations from Wikipedia infoboxes and categories. Wikipedia infoboxes and categories follow a specific structure based on the Wiki-markup language. Thus the rules employed by these methods are not applicable to general Web text. An example infobox is shown in Figure 2.2 and its corresponding markup is shown in Figure 2.3.



```

{{Infobox scientist
| name           = Barbara Liskov
| image          = Barbara Liskov.PNG
| image_size     = 192 x 160
| caption        =
| birth_date     = {{Birth date and age|1939|11|7|mf=yes}}
| birth_place    =
| death_date     =
| death_place    =
| residence      =
| nationality     = [[United States|American]]
| ethnicity      =
| field          = [[Computer science]]
| work_institution = [[Massachusetts Institute of Technology]]
| alma_mater     = [[University of California, Berkeley]]<br />[[Stanford University]]
| doctoral_advisor = [[John McCarthy (computer scientist)|John McCarthy]]<ref>{{MathGenealogy |id=61932}}</ref>
| doctoral_students =
| known_for      =
| author_abbreviation_bot =
| author_abbreviation_zoo =
| prizes         = [[IEEE John von Neumann Medal]],<br />[[Turing Award|A. M. Turing Award]]
| religion       =
| footnotes     =
}}

```

Figure 2.3: Wiki-markup corresponding to the infobox in Barbara Liskov’s Wikipedia page

2.1.6 The Semantic Web

The World Wide Web Consortium (W3C) ’s Semantic Web vision aims to advance the Web by augmenting documents with machine-readable data [12]. Towards this end, the Semantic Web strives to develop standards that enable data publishers on the Web to enrich their data with metadata. It is envisioned that by relying on common standards, Web applications can interoperate with one another to perform complex tasks on behalf of users. In comparison to the Web of documents, the uptake of the Semantic Web has been relatively slow [105]. While speculation as to why this is the case is beyond the scope of this dissertation, requiring additional work from content publishers likely plays a role. Therefore, knowledge base construction aim for semantic web capabilities from a different perspective: develop tools to parse unstructured or semi-structured data and convert it to structured data. Rather than have a uniform schema, knowledge base construction projects have been evolving in parallel, with different knowledge bases constructed at multiple sites each with its own schema.

2.2 KNOWLEDGE BASE PRELIMINARIES

2.2.1 Knowledge Base

Formally, a knowledge base is a special-purpose database used for the collection and management of knowledge in the form of logical statements. Some knowledge bases are manually constructed; for example lexical knowledge bases such as WordNet [43] are compiled by domain experts. Other manually constructed knowledge bases include Cyc [69], health databases like MeSH and SNOMED, and semantic search engines like Wolfram Alpha (www.wolframalpha.com). Though manually constructed knowledge bases are of high quality, especially those authored by domain experts such as WordNet, they tend to be small in size. In this dissertation, the focus is on developing tools for *automatically* populating knowledge bases.

2.2.2 Ontology

Ontology is a term borrowed from philosophy where it is defined as the study of the nature of being, existence, or reality. In Computer Science and knowledge representation, Gruber [48] made the first attempt at a definition. He defines an ontology as a (logical) description of the concepts and entities that exist in a domain and the relationships that hold among them. Different from other kinds of knowledge bases, ontologies have the ability to reason about entities and relationships within a domain.

To encode knowledge in ontologies, several knowledge representation languages have emerged. Ontology languages vary in structure. Early ontology languages were frame-based. There has also been languages based on first-order logic such as CycL and KIF. Description logic-based languages produced the most commonly used ontology language today, RDFS/OWL. RDFS/OWL is a standard of the World Wide Web Consortium (W3C). In RDFS/OWL all entities and relations are defined as RDF resources, and identified by Uniform Resource Identifiers (URIs). With RDF, data is stored as subject-predicate-object (SPO) *triples*.

Contemporary ontologies are structurally similar, even though they adapt different ontology languages. Most ontologies consist of classes, entities and relations.

2.2.3 Classes

Each ontology defines a type system around which its entities are organized. These classes (unary predicates) may include common types such as *person*, *politician*, *scientist*, *location*, *etc.* They may also include finer grained types such as *female person*, *computer scientist*, *governor*, *etc.* In addition, subsumption relations among classes may also be specified. Examples of subclass-superclass are:

```
subclassOf (governor, politician),  
subclassOf (scientist, person),  
subclassOf (computer scientist, scientist),  
...
```

2.2.4 Entities

Each ontology also defines individual entities. For example, it contains instances of people, companies, and so on. Entity type information is provided in order to specify the classes to which each of the entities belongs. For example, the *instanceOf* relation may specify the following:

instanceOf (Arnold Schwarzenegger, governor),
instanceOf (Bill Clinton, politician),
instanceOf (Bill Clinton, philanthropist),
 ...

Additionally, a way of mapping names to entities is provided. Real-world entities can have multiple names including nicknames. These are different from the entity's full name. Natural language text documents typically introduce the entity by its full name and thereafter refers to it by a shorthand form (e.g., a person's last name). The official name of an entity in the ontology, is used as a canonical representation of the entity. Therefore, names are mapped to entity identifiers through some kind of "means" relation. For example, the "means" relation of a given ontology may specify the following mappings of names to entities.

means ("Bill Clinton", William Jefferson Clinton),
means ("President Clinton", William Jefferson Clinton),
means ("Paris", Paris, France),
means ("Paris", Paris Hilton),
 ...

2.2.5 Relations

Ontologies define sets of relation types that are deemed interesting. The list of relations varies from ontology to ontology. Common relations include: *hasWonPrize(person, award)*, *headquarteredIn(organization, location)*, *isMarriedTo(person, person)*. The number of relations in ontologies is a well-known shortcoming. They are typically less than 1000 relations in each ontology. There are often many interesting relations that are not covered, for example, *competedAgainst(team, team)*, *nominatedForPrize(person, award)*. As discussed in Chapter 4, the second contribution of this dissertation, PATTY, addresses this limitation.

2.3 STORAGE AND ACCESS MANAGEMENT

Knowledge representation languages provide a conceptual description or model of knowledge data. RDF models data as subject-predicate-object (SPO) *triples*. Once the data is modeled as RDF triples, it needs to be made persistent for later use and access.

2.3.1 Relational Databases

A relational database is a common choice for storing data. It is created using the relational model by Codd [27]. The relational model organizes data in tables

where rows correspond to distinct real-world entities, and columns are attributes of the entities. For example, a company might store information about its employees in a database table where each row has information about a different employee and each column stores a particular employee attribute (name, address, salary, etc.).

A relational database requires a schema to be defined upfront before loading the data. The schema describes and prescribes how the data will be organized in the database, specifically how it is divided into database tables. There are multiple ways to organize the same data into a database. For RDF triples, the simplest schema is to store the data in a single table, consisting of three columns, one column for subject, one for predicate, and one for object. The problem with this approach is that, accessing the data may be slow if the query requires multiple self-joins on the same table. An alternative schema is to organize the data into a wider, sparser table, containing one column for each unique predicate. So far the dominant way of storing RDF in relational databases is the three-column-table. The ontologies used and developed in this dissertation are typically stored in relational databases, following a three-column-table schema.

2.3.2 NoSQL Databases

The requirement for a pre-defined schema in relational databases makes it difficult to deal with highly dynamic and unpredictable data. Data in certain domains is not amenable to fixed schemas. As a result, recent years have seen a new type of database management systems emerge — key-value oriented NoSQL databases [90]. These types of databases are not built on tables as the relational model, consequently, they do not use SQL for data manipulation. Instead, NoSQL databases store structured data as documents with dynamic schemas. For example, MongoDB[9] is based on lightweight XML(JSON)-like documents with dynamic schemas.

NoSQL databases started to gain ground in major Internet companies such as *google.com*, *facebook.com* and *twitter.com*, which have to deal with huge quantities of data. NoSQL databases are specifically designed to deal with large volumes of data as they can be seamlessly deployed on distributed hardware architectures such as clusters of multiple machines. Data can be partitioned among different servers. The system can easily scale out by adding more servers and failure of a server can be managed.

We use the MongoDB[9] system to store the large number of triples produced along with the relations extracted in Chapter 4.

2.3.3 Triplestores

Triplestores [102] are special-purpose databases designed for storing and retrieving triples. Example triplestores include Apache Jena, RDF-3X and Sesame. Some triplestores are built on top of relational databases, others have built their own in-house database engines. For example, Freebase [13] runs on a database created by Metaweb based on a graph model. The underlying data structures are

not based on tables, instead Freebase uses a set of nodes and a set of links that establish relationships between the nodes. Queries to the database are made in Metaweb Query Language (MQL).

2.4 RELATED WORK

There are many contemporary projects on knowledge base construction. Collectively, our contributions aim for a Web-scale, yet high-accuracy approach to knowledge base construction. The following characteristics set apart our approach from previous work.

1. **Web-scale.** For scaling to large data collections, our algorithms are developed as distributed MapReduce algorithms which scale to large data collections.
2. **Canonical entities.** In order to ensure consistency among the extracted facts, we extract facts between canonical entities. This is in contrast with approaches that extract facts between noun phrases without any attempt to detect which noun phrases refer to the same entities and which noun phrases refer to different entities.
3. **Newly emerging entities.** Because we map noun phrases to canonical entities, in the event of a new entity we may not be able to process and accept facts pertaining to such previously unseen entities. Our contributions include a method for adding type information to new entities.
4. **Open set of relations.** Our methods extract facts pertaining to a large set of binary relations.
5. **Dynamic knowledge.** The above four characteristics enable fact extraction from dynamically updated data sources such as news and social media.

We now discuss previous work, focusing on the degree to which they possess the above characteristics.

2.4.1 SOFIE

SOFIE [109] is an *Ontology-based IE* system for fact extraction. SOFIE's main premise is to use logical reasoning to verify the plausibility of potential new facts against prior knowledge already in the YAGO ontology[108]. In particular SOFIE brings together entity-disambiguation, pattern matching and rule-based reasoning in one framework. For entity disambiguation SOFIE first looks up, in YAGO, all the possible entities a noun phrase may refer to. It then determines which of the candidate entities the noun phrase likely refers to. This is done by computing similarity overlaps between the context of the noun phrase and what the ontology knows about the entities. SOFIE has the ability to re-assess disambiguation decisions if further evidence is found that contradicts the chosen entity. In a similar vein, SOFIE can reason about the plausibility of patterns. It reasons about the likelihood that a pattern expresses a given relation based on the pairs of entities the pattern occurs with. SOFIE applies the pattern-fact duality [14] to

reason about patterns. SOFIE also takes into account other background knowledge, such as type constraints of relations, functional constraints on relations, and other domain-specific rules if available.

The experiments reported in [109] dealt with 1,000's of input documents in about 10 hours on a single computer. Therefore, in its original form SOFIE is not designed for *Web scale*. SOFIE produces fact about *canonical entities*. However, it was run for a few hand-specified relations only and does not handle out-of-knowledge base entities; thus it cannot accumulate knowledge for rapidly changing data sources. Our contribution in Chapter 3 builds on SOFIE; we compare SOFIE with PROSPERA in some of our experiments.

2.4.2 ReadTheWeb/NELL

The ReadTheWeb project developed ensemble methods to populate its NELL ontology [20]. The idea behind NELL is a Never-Ending-Language-Learning system which learns new facts 24 hours a day, 7 days a week, and corrects itself over time as it learns to better understand natural language. NELL learns new entities and new facts. It employs a combination of methods in an ensemble manner coupled with constraints for extracting entities and facts from a large Web corpus. Like SOFIE, NELL automatically takes into account type constraints and other domain-specific constraints. The NELL method also learns constraint rules from the facts that it gathers.

NELL scales to Web data as it was applied to a corpus of 500 million pages. In addition, its architecture was deployed on a cluster of multiple machines. NELL extracts facts between pairs of noun phrases instead of *canonicalized entities*. In this sense NELL can tackle new entities as they emerge. NELL was originally based on a fixed set of relations. It was recently extended in *OntExt* [77], to automatically extract new relations (beyond the pre-specified ones) for a given type signature of arguments. However, *OntExt* only learned 250 relations. It is not clear if NELL's set of relations can compile facts from rapidly changing sources in a comprehensive and large-coverage manner, due to its relatively small set of relations.

2.4.3 TextRunner/Reverb

TextRunner [130, 10] is an *Open IE* system which harvests facts by considering all relationships between pairs of noun phrases. *ReVerb* [41] extended TextRunner by constraining patterns to verbs or verb phrases that end with prepositions [41]. The TextRunner method is a three step process:

1. TextRunner first creates a self-supervised classifier which extracts a few facts from the input corpus and then labels them as good or bad based on a few hand-specified linguistic rules. This training portion of the corpus is run through expensive linguistic processing that produces enough information for the classifier to self-train. Example rules for determining if a fact is good or bad include: the relation string contains a verb, or the left-hand entity is the clause subject, etc.

2. Second, TextRunner extracts facts from the entire corpus, using the self-trained classifier to determine if the facts are good or bad.
3. In the third and final step, TextRunner uses frequency statistics to determine if the fact is indeed true or not.

TextRunner is designed for *Web-scale*, it was applied to 9 million Web pages. Like NELL, TextRunner extracts facts pertaining to pairs of noun phrases as opposed to *canonical entities*. Also like NELL, it can tackle new entities as they emerge. TextRunner is not restricted to pre-defined relations, therefore it can likely handle dynamic Web content. However, it must be noted that due to its relaxation on what constitutes an entity or relation, the output of TextRunner is relatively noisy and makes it very difficult to apply any form of consistency reasoning on the extracted data. This generally leads to low quality output.

2.4.4 Probase

Probase [124] is a large probabilistic taxonomy of concepts based on extracting *isA* relations at *Web-scale*. The Probase concept taxonomy is similar to type hierarchies in other knowledge bases. The main difference is that it includes both widely used concepts such as *cities*, *mucisians*, *etc.*, as well as highly specific concepts such as *renewable energy technologies*, *common sleep disorders*, *etc.* Furthermore, there are probabilities attached to each *isA* assertion. Probase extracted concepts from 1.68 billion Web pages, resulting in a taxonomy of 2.7 million concepts.

Probase does not extract facts pertaining to relations other than the *isA* relation. Furthermore, Probase *isA* assertions pertain to noun phrases as opposed to *canonical entities*.

2.4.5 WebTables

There is a line of work that extracts information from tables on the Web. One of the early work along these lines is the WebTables system [18]. Recent work has gone a step further to annotate data extracted from tables with semantics [74, 112]. Venetis et al. [112] add class labels to columns and binary relationships between pairs of columns. They leverage a database of entities, their types and the relationships that hold among them. A table is assigned a class label (semantic type) if a large number of the entities in that column are identified with that label. Similarly, a pair of columns is labeled with a relationship if a large number of pairs from the two columns occur in the facts of that relationship.

The WebTables system extracted over 100 million tables[18]. WebTables and related work primarily focus on HTML tables and are not concerned with natural language text. Furthermore, the output pertains to noun phrases as opposed to *canonical entities*.

2.4.6 YAGO, DBpedia, Freebase

Knowledge bases like YAGO, DBpedia, and Freebase extract facts from specific data formats such as the Wikipedia infobox structure for YAGO and DBpedia. As shown in Figure 2.3, the format of infoboxes is semi-structured. Therefore these systems are not meant for full-fledged natural language processing but instead leverage domain-specific rules. In addition to extracting data from structured sources such as Wikipedia and musicbrainz.com, Freebase also takes human updates in a crowd-sourcing manner similar to the way Wikipedia is authored. Each of the knowledge bases defines its own population of entities, which grow with updates to the knowledge base. All the three knowledge bases are limited to less than a 1000 relations. These limited sets of relations are not amenable to extracting facts from rapidly changing data sources.

2.5 SUMMARY

This chapter provided a brief history of information extraction, starting with early work on IE as a template filling task, proceeding to modern day pattern learning systems, and to even more recent work on open relation discovery. It also presented methods that leverage semi-structured resources. The chapter then defined central concepts in knowledge base construction, including classes, entities and relations. It then discussed some of the storage options for knowledge bases, covering relational databases, NoSQL databases and triplestores. Lastly, the chapter gave a discussion of related projects, comparing them to the contributions of the dissertation with respect to a number of design decisions.

Fact Extraction with PROSPERA

Harvesting relational facts from text is a task central to automatic knowledge base construction. Fact extraction is a formidable challenge, thus state-of-the-art approaches still suffer from a number of limitations. In particular, there are major challenges regarding the trade-offs between precision, recall, and scalability. Techniques that scale well are susceptible to noisy patterns that degrade precision, while techniques that employ deep reasoning for high precision cannot cope with Web-scale data. This chapter presents this dissertation’s contribution toward Web-scale, high-quality fact extraction — the PROSPERA system.

3.1 MOTIVATION

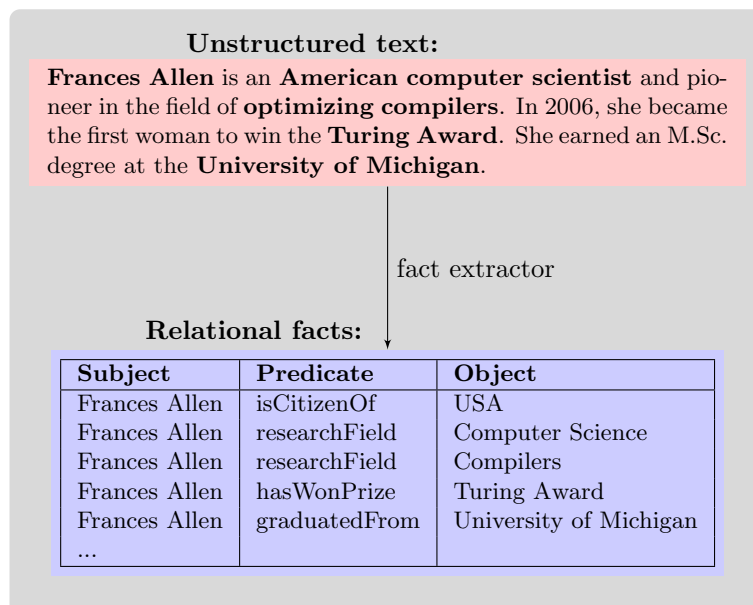


Figure 3.1: The fact extraction problem.

The process of fact extraction, as illustrated in Figure 3.1, takes as input a corpus of unstructured, natural language text documents, and produces structured data in the form of *subject-predicate-object* triples.

The prevalent approach to fact extraction is to use *pattern-based extractors*. Some methods additionally employ *consistency constraint reasoning* (or some form of constraint-aware statistical learning).

Pattern-based Extractors

In a pattern-based approach (e.g., [14, 3, 10, 19]), the system starts with a few *seed facts* to bootstrap the extraction process. For example the relation *teamWonTrophy*, that holds between soccer teams and trophies might have a seed fact:

teamWonTrophy(Germany, FIFA_World_Cup)

This seed fact was true in 1974 and 1990 and by finding it in text we can automatically detect textual *patterns* like:

X won the final and became the Y champion

Such a pattern can, in turn, discover new facts such as:

teamWonTrophy(Spain, FIFA_World_Cup)

This was indeed true in 2010. While this approach is good for high recall, it however may lead to noisy patterns and degrade precision. For example, the same seed may lead, after a few iterations, to frequent but spurious patterns such as:

X lost the final of Y.

Such a pattern can erroneously pick up the false positive:

teamWonTrophy(Netherlands, FIFA_World_Cup),

which was never true as of 2012.

Consistency Constraint Reasoning

Reasoning-enhanced systems (e.g., [109, 134, 20]) check the plausibility of the extracted *fact candidates* by their mutual consistency based on specified logical constraints. For example, suppose the system picked up a spurious pattern for the *hasAcademicAdvisor*:

X benefitted from the deep insight of Y

The pattern may lead to the fact candidate:

hasAcademicAdvisor(Tim_Berners-Lee, Max_Planck).

A constraint-aware system can invalidate this hypothesis by using prior knowledge about the birth year of Berners-Lee (1955), the death year of Max Planck (1947) and a rule that a student must be born (or at least, say, 15 years old)

before graduating. Similarly, type constraints can be used to eliminate false positives such as:

graduatedFrom(Tim_Berners-Lee, Real_Madrid).

This is not a plausible fact because *Real_Madrid* is a sports team not a university.

Consistency reasoning is expensive and faces *scalability problems*. The *ReadTheWeb* project [20, 21] recently showed how to parallelize a constraint-aware approach, using an ensemble-learning method over coupled pattern learners. However, this method yields relational pairs between *names* of entities rather than referring to *entities* themselves. For example, the result may contain instances such as:

playsInLeague(Real_Madrid_CF, Champions_League),
playsInLeague(Real_Madrid, UEFA),
playsInLeague(Real, UEFA_Champions_League).

The above three instances refer to the very same fact.

This is much lower-quality knowledge than a canonical representation that would state the fact in terms of uniquely named entities:

playsInLeague(Real_Madrid_CF, UEFA_Champions_League).

There would then be a separate relation for capturing the synonyms of entities, for example:

means(Real, Real_Madrid_CF).

Note that many names are ambiguous, so they appear in *means* facts for different entities. For example, Wikipedia knows more than 20 soccer clubs called *Real*, there are several currencies named *Real*, and so on.

PROSPERA builds on the SOFIE approach [109], which combines a *pattern-gathering phase* with a *consistency-reasoning phase*. Experiments in [109] dealt with 1,000's of input documents and 10,000's of hypotheses in about 10 hours on a single computer. Clearly, this is not good enough for Web scale. Moreover, while precision was very high, the constraint-based reasoning severely limited recall. To increase recall, one could relax the pattern gathering and allow more patterns and co-occurring fact candidates to enter the reasoning phase. However, this entails two problems:

1. *Efficiency*: it presents the reasoner with a much larger hypotheses space (more clauses for a Weighted-MaxSat solver [109]);
2. *Quality*: it introduces noisy patterns such as "X played in the qualification round of Y" for *teamWonTrophy*, and sparsely occurring patterns such as "P scored her third goal in the triumph of T" for the *athletePlaysForTeam* relation, which is hardly generalizable for learning given that female players ("her goal") have less coverage in Web sources and "three" (goals in the same match) is a rare feature.

In summary, Web-scale knowledge harvesting is not yet as robust and scalable as we wish it to be. Key limitations are:

- *non-canonical output*: shallow and scalable methods yield pairs of (ambiguous) names rather than truly entity-based facts;

- *scalability*: deep methods with constraint-based reasoning do not scale and face efficiency challenges;
- *limited recall*: reconciling high precision with high recall is challenging because of the need for and difficulty in dealing with noisy and sparse patterns.

3.2 CONTRIBUTION

PROSPERA overcomes, to a large extent, the above problems and shows how to reconcile high precision, high recall, and scalability. Although PROSPERA is based on the SOFIE framework [109], it includes major extensions along the following lines.

For pattern-based gathering of fact candidates, we introduce a new notion of *n-gram-itemset patterns*. In prior work, patterns are either consecutive substrings from the surface text that connects two named entities in a document or words on the path of the linkage graph that results from dependency-parsing the text. The former is noisy, the latter entails an expensive deep-parsing step. To gather more interesting patterns in an inexpensive way but avoid becoming even noisier, we define a pattern to be a sequence of variable-length n-grams in the context of two entities, and we employ algorithms for frequent-itemset mining to efficiently compute pattern statistics. Moreover, we allow patterns to lift individual parts into their corresponding part-of-speech tags (word categories); the resulting generalized patterns have higher support. For example, consider the lifted n-gram-itemset:

{ *scored PRP ADJ goal; match for* }

The above pattern covers all sentences with the two phrases regardless of which pronouns (PRP) or adjectives (ADJ) are used and regardless of any preceding, following, or intermediate text such as relative clauses or appositions.

For constraint-based reasoning, PROSPERA leverages SOFIE’s Weighted-MaxSat solver, but uses confidence measures from pattern gathering for setting clause weights. PROSPERA also considers negative patterns that often co-occur with entity pairs that are not in the relation of interest. For example, we can specify upfront a negative seed fact:

–teamWonTrophy(Netherlands, FIFA_World_Cup).

This tells PROSPERA that the given pair is not an element of the *teamWonTrophy* relation, then the system can automatically detect the negative pattern:

lost the final.

The system will in turn downgrade the weights of clauses that relate this pattern with fact candidates. As the reasoning still prioritizes precision over recall; the entire two-phase procedure is iterated, with re-adjusted weights, yielding much higher recall than the original SOFIE.

Finally, to gear up for Web-scale harvesting, we have developed a new architecture where tasks of all phases can be distributed across many computers and run

in parallel. For the pattern gathering and analysis, we partition by input sources, but ensure that the n-gram-itemset mining benefits from global statistics. For reasoning, we employ graph-partitioning algorithms on the clauses graph and construct parallel reasoning tasks. Tasks are mapped to distributed machines using the Hadoop implementation of the MapReduce framework.

In summary, the chapter’s novel contributions are:

- *rich and accurate patterns*: a new notion of n-gram-itemsets to generalize narrow patterns and more precisely capture noisy patterns;
- *clauses weighting*: using confidence and support statistics from the pattern-based phase for carefully weighting the reasoner’s input clauses and re-weighting them in iterations of both phases;
- *distributed architecture*: organizing both phases in a way that data and load can be partitioned across parallel machines for better scalability;
- *Web-scale experiments*: demonstrating the run-time efficiency, high precision, and high recall of our methods by performing knowledge harvesting on the ClueWeb09 corpus [26]—with 500 million Web pages. We compared PROSPERA to the ReadTheWeb experiments of [20] and outperformed their results.

All data on the experiments reported here are made available on the Web site www.mpi-inf.mpg.de/yago-naga/prospERA/.

3.3 RELATED WORK

Pattern-based Fact Gathering

Using various forms of pattern matching for fact extraction from natural-language documents has a long history in the NLP, AI and DB communities, dating back to the work by Hearst [56]. Hearst patterns [56] were the first part-of-speech-enriched regular expressions (so-called *lexico-syntactic* patterns) which aim to identify instances of predefined relationship types from free text. For example, for the *instanceOf* relation we can automatically determine instances from noun phrases around a syntactic pattern like:

$$\langle \text{class such as } \{NP_1, NP_2 \dots (\text{and|or})\} NP_n \rangle,$$

where *class* is the plural of a semantic type such as *companies* or *football players* and NP is the POS tag for proper noun such as *Microsoft* or *David Beckham*. Hearst patterns are hand-crafted; for arbitrary target relations (such as *hasCollaborator* or *hasAcademicAdvisor*) it would be difficult to come up with an exhaustive set of expressive yet accurate patterns.

Seminal work by Brin [14] was centered around the *duality of facts and patterns* which refers to the iterative mutual enrichment between patterns and facts, whereby seeds can be used to find new patterns and new patterns can be used to find more seeds. Systems based on the fact-pattern duality (e.g. [14, 3, 23,

38, 10, 16, 129, 19]) are bootstrapped with seed facts for given relations and automatically iterate, in an almost unsupervised manner, between collecting text patterns that contain facts and finding new fact candidates that co-occur with patterns. Statistical measures such as point-wise mutual information (PMI) are used to assess the goodness of newly found facts and of characteristic patterns. This is a powerful machinery for high recall, but it often leads to noisy patterns and may easily drift away from the target relations.

KnowItAll [39], and Text2Onto [23] improved the statistical assessment of fact candidates and patterns in a variety of ways, regarding robustness (lower false-positive rate while still retaining high recall), expressiveness (e.g., by adding part-of-speech tagging and other NLP-based features), and efficiency (lower-cost estimates of statistical measures). The LEILA approach [107] uses dependency-parsing-based features to boost precision, and also extends Brin’s bootstrapping technique by incorporating both positive and negative seeds. TextRunner [130, 10] extended the pattern-fact bootstrapping paradigm to *Open IE*, where the harvesting is not focused on a particular relation but considers all relationships expressed in verbal phrases. However, this relaxation makes it very difficult to apply any form of consistency reasoning on the extracted data.

Statistical-learning Methods

Statistical-learning methods for relational graphs (e.g. [99, 37, 24, 134, 20, 100]) aim to overcome semantic drift limitations of pattern-based fact gathering methods. To this end, they incorporate probabilistically weighted rules. Such rules couple random variables that denote whether fact candidates are true or false. Joint inference is then applied over all fact candidates together. While such methods yield high precision, they tend to have low recall. Such methods also tend to have high computational costs as inference is usually based on Markov-chain Monte-Carlo methods such as Gibbs sampling. This is particularly concerning because large-scale knowledge harvesting can easily lead to situations with hundred thousands of highly connected random variables. StatSnowball [134] is a system for fact harvesting that makes extensive use of Markov Logic Networks (MLNs) and Conditional Random Fields (CRFs). The Kylin/KOG framework [118, 122] applies MLNs to infer “missing infobox values” in Wikipedia.

Rule-based & Declarative Approaches

There is a line of work that explores declarative approaches to information extraction. For example, System T [98], Cimple [33, 126], or SQOUT [66] organize pattern-matching steps into execution plans which allow database-style optimizations. However, they are limited to deterministic patterns such as regular expressions. Therefore, they are not suited to dealing with the inherent uncertainty of natural language. The *DBLife* community portal (dblife.cs.wisc.edu) is based on the *Cimple* tool suite [33, 126]. *DBLife* features automatically compiled *super-homepages* of researchers with bibliographic data as well as facts about community services (PC work, etc.), lectures and much more. For gathering and reconciling these facts, Cimple provides a collection of DB-style extractors based on

pattern matching and dictionary lookups. These extractors are combined into execution plans, and periodically applied to a selected set of relevant Web sources. Rule-based fact extraction has also been customized to Wikipedia as a knowledge source, primarily to exploit the semi-structured infoboxes. DBpedia [8] pioneered large-scale extraction of infobox facts. It uses simple, recall-oriented techniques and places all attribute-value pairs into its knowledge base as they are. YAGO [108], on the other hand, uses a suite of cleanly designed rules for frequently used infobox attributes to extract and normalize the corresponding values.

Ensemble-learning

The *ReadTheWeb* project [20] combines an ensemble-learning approach with coupled pattern learners. In the NELL experiment [21], it produced about 230,000 facts for 123 different categories (unary relations) and 12,000 facts for 55 different types of binary relations from a large Web corpus with 500 million pages (a variant of the ClueWeb09 corpus).

Although this was, at the time of this study, the largest-scale experiment of this kind in the literature, it has notable limitations. First, its recall on binary relations - our primary focus - was rather low: only 12,000 facts after 66 days, and quite a few of the 55 relations acquired less than 100 facts.

Note that one could retrieve the instances of unary relations (categories) much more easily from existing knowledge bases such as DBpedia or YAGO; the real difficulty lies in binary relations. Second, the experiment involved some human interaction on a daily basis, after each iteration of rule learning, to manually remove bad extraction rules. Third, the output refers to non-canonical names rather than uniquely identified entities. The entire experiment ran more than 60 days on a large cluster. We use the *ReadTheWeb* results (which – thanks to the authors – are available on the Web) as a yardstick against which we measure our approach.

Reasoning-based Methods with Prior Knowledge

Reasoning-based methods with prior knowledge such as SOFIE [109] leverage knowledge about entities and their types. SOFIE decomposes the entire fact harvesting into two phases:

1. Pattern-based gathering of candidates for high recall
2. Reasoning about candidates and constraints for high precision

In contrast to statistical-learning methods, there is no probabilistic interpretation over the fact candidates; instead all hypotheses are organized into a set of propositional-logic clauses. Then a customized approximation algorithm for Weighted-Maximum-Satisfiability (MaxSat) is employed to identify a subset of fact candidates and characteristic patterns that together constitute “the truth”.

Constraints include functional dependencies, inclusion dependencies, type constraints, and more. Type information is extremely beneficial in early pruning of the search space of the MaxSat solver; for example, the first argument of the *teamWonTrophy* relation must be of type *sportsTeam*, as opposed to say *businessEnterprise* (which immediately rules out the hypotheses that Google has won the FIFA World Cup).

SOFIE uses the YAGO ontology [108] to map names to entities. YAGO includes entity name synonyms gathered from the redirection pages and hyperlink anchor texts in Wikipedia. For every name encountered, SOFIE looks it up in YAGO and computes a disambiguation prior for all the potential entities that the name may refer to. The disambiguation prior is assigned to each possible mapping to indicate the likelihood that the name refers to each of the candidate entities. SOFIE computes the disambiguation prior using the *bag of words model*. This model computes the similarity between the text context the name was extracted from, and the information that YAGO knows about a given candidate entity. SOFIE generates hypotheses over all the candidate entities and their disambiguation prior weights. The entity disambiguation hypotheses are resolved in the same MaxSat solver as the hypotheses about fact candidates.

3.4 OVERVIEW OF PROSPERA

We assume that we have an existing knowledge base with typed entities, which more or less includes all individual entities and their types (e.g., *footballPlayer*). In our experiments, we use YAGO [108] for this purpose, which provides us with more than 2 million typed entities and a dictionary of the possible meanings of a surface string. The YAGO *means* relation explicitly maps surface strings onto individual entities. This does not yet resolve any name-entity ambiguity, though (see below).

PROSPERA takes as input a set of binary relations and their seed examples.

Definition 3.4.1 (Binary Relations, Seeds, and Counter-seeds) *We are given a set of binary relations R_1, \dots, R_m of interest, each with a type signature and a small set of seed facts and optionally, a set of counter-seeds. The latter are entity pairs that are asserted to be definitely not among the instances of a given relation, for example:*

$\neg \text{teamWonTrophy}(\text{USA}, \text{FIFA_World_Cup})$.

We now consider a textual corpus, e.g., a set of Wikipedia articles, Web pages, or news articles, as input for harvesting facts about the relations of interest. PROSPERA processes this data in three phases:

1. *Pattern gathering*: This phase identifies triples of the form (e_1, p, e_2) where e_1 and e_2 are any two entities, each occurring as a different noun phrase, and p is a surface string that appears between e_1 and e_2 . As sentences contain merely names rather than entities, we determine all possible mappings using the *means* relation of YAGO and then use a disambiguation heuristics based

on context similarities. We form a text-window-based bag of words around the name, and a bag-of-words context around each entity that the name could possibly be mapped to. For the latter, we use the type names of the entities in YAGO (which include names of Wikipedia categories to which the entity belongs). The entity whose context has the highest word-level overlap with the name’s bag of words is chosen for the name-to-entity mapping.

2. *Pattern analysis*: This phase generalizes the basic patterns from the previous step by transforming them from long, overly specific phrases into sets of n -grams succinctly reflecting the important sub-patterns. We also compute statistics and similarity measures about patterns and use them to generate fact candidates. The pattern analysis is further detailed in Section 3.5.
3. *Reasoning*: This step complements the statistical evidence of the previous phase with logical plausibility for high precision. The fact candidates are passed to a MaxSat-based reasoner which considers pre-specified constraints to ensure mutual consistency of accepted facts and their compatibility with the consistency constraints. The reasoning phase is further detailed in Section 3.6.

Our experiments demonstrate that the disambiguation heuristic is very powerful and fairly accurate. Employing this name-entity mapping upfront is a departure from the SOFIE architecture, where entity disambiguation was part of the constraint-based reasoning [109] (and similar techniques were also used in other approaches, e.g., based on Markov logic or factor graphs [37, 120]). However, integrating name-entity mapping into consistency reasoning has high computational cost as it leads to a much larger space of clauses and Weighted MaxSat is an NP-hard problem with inherent combinatorial complexity. PROSPERA’s streamlined approach is efficient and scales very well to Web proportions.

As we will explain later, each of the three phases can be parallelized on a distributed platform. Moreover, we can iterate the three phases by feeding the output of the reasoner back into the pattern gathering, treating newly found facts as additional seeds for the next iteration. While such feedback loops are well studied for knowledge-harvesting methods that are exclusively pattern-based, such as [14, 3, 10], our approach distinguishes itself from that previous work by including the reasoning phase in each iteration. This strengthens the choice of next-iteration seeds and ensures that precision is kept high. The overall architecture of PROSPERA is illustrated in Figure 3.2.

3.5 PATTERN ANALYSIS

Seed Patterns

The basic pattern phrases from the pattern gathering phase are fed into a frequent n -gram-itemset mining algorithm for identifying strong patterns. For example, for the *hasAcademicAdvisor* relation, the pattern gathering may yield a sentence like:

“Barbara Liskov was the first woman in the US who was honored with the title of a doctor of philosophy (Ph.D.) from a technical department at Stanford University”.

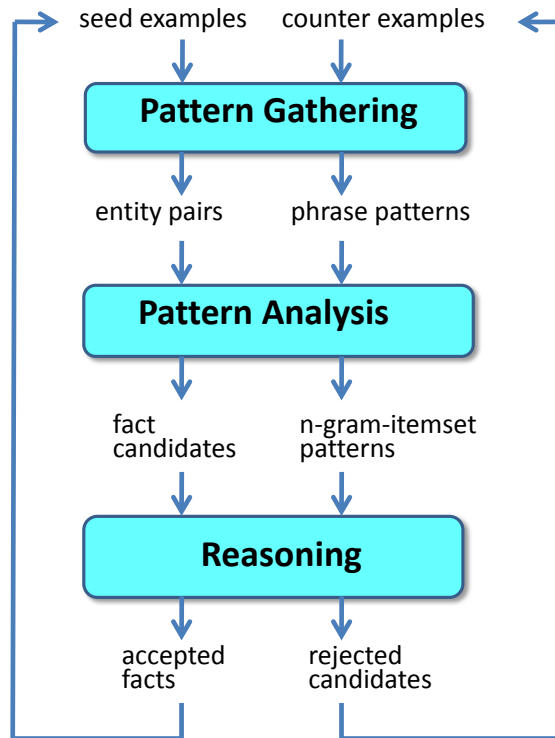


Figure 3.2: Architecture of the PROSPERA System.

Such a long and extremely specific phrase does not generalize for fact extraction, as there are hardly any other entity pairs with exactly the same pattern. A standard technique would be to consider a subsequence as a better pattern, for example, the substring:

“was honored with the title of a doctor of philosophy”.

But even this is overly specific and would occur very sparsely in the corpus.

To overcome this problem, we break down the phrases into variable-length n -grams of successive words, i.e., multiple n -grams per phrase. Then a much better pattern would be the n -gram-itemset consisting of three n -grams:

{ honored with; doctor of philosophy; Ph.D. }.

Definition 3.5.1 (Basic Pattern) Given a set $SX(R_i)$ of seed examples for a relation R_i and an input set S of sentences (or, more generally, token sequences), a basic pattern p is a sequence such that $e_1 p e_2$ occurs in S for at least one pair $(e_1, e_2) \in SX(R_i)$.

Definition 3.5.2 (N-gram-itemset Pattern) An n -gram itemset pattern is a set Q for which there is at least one sequence $s \in S$ that can be written as $s = h e_1 p e_2 t$ with a seed example $(e_1, e_2) \in SX(R_i)$ such that for all $q \in Q$ the length of q is at most n words (tokens) and q is a subsequence of p . Observe that we only consider infix patterns for p . However, even when the basic pattern is not an infix occurring between e_1 and e_2 , the n -gram-itemset pattern definition holds.

A new sentence is a candidate for containing a fact if it contains all three n-grams or at least a large subset of this n-gram-itemset. For efficiently generating the n-gram-itemset patterns, we apply the technique of frequent itemset mining [1, 95] which has been widely used to discover interesting relations among items in databases. More specifically, we use the Generalized Sequential Pattern (GSP) algorithm [95] for generalized sequence mining .

In frequent itemset mining, the task is to find sequences of words that occur in a specified minimum number of patterns. That is, let $I = a_1, a_2, \dots, a_n$ be the vocabulary of words used in patterns and $D = p_1, p_2, \dots, p_n$ be the database of patterns. An itemset X , where $X \subset I$, is considered frequent if $support(X) \geq minsup$. The support of X , $support(X)$, is the proportion of patterns in the database which contain the itemset X and $minsup$ is the minimum support required for an itemset to be considered frequent.

The GSP algorithm generates ordered frequent itemsets. The itemsets are sequences in this case because order of words matters. GSP works iteratively, generating k -grams in a bottom-up manner. In the first round, $k = 1$ and GSP retrieves all frequent individual words (1-gram), in the second round 2-grams are generated by joining pairs of 1-grams and the same procedure is followed for subsequent values of k . This way we end up with itemset patterns, where each pattern is a set of frequent n-grams.

Often, there is a variety of different wordings regarding pronouns or injected adjectives that render n-grams sparse. For example, because of the variations:

“received his” and “received her”,

we may dismiss good n-grams as too infrequent (e.g., “received her”). To overcome this issue, we run part-of-speech (POS) tagging on each of the originally gathered sentences, assigning word categories like nouns, verbs, pronouns, etc. to each of the words. We tentatively replace words with their POS tags, to obtain a more general form of n-grams that we refer to as *lifted patterns*.

Definition 3.5.3 (Lifted Pattern) A lifted pattern is a pattern p where words with certain POS tags are replaced by their tags.

We focus on POS tags for pronouns, prepositions, articles, and adjectives. This way we can generalize the above pattern into the lifted form:

“received PRP”

where PRP denotes an arbitrary pronoun.

To assess the goodness of an n-gram-itemset pattern, we compute the following statistics about co-occurrence of patterns with seed examples and counterexamples. *Support* captures the frequency of a pattern in conjunction with a seed fact, whereas *confidence* reflects the ratio of the pattern co-occurring with seed facts versus counter-seeds.

Definition 3.5.4 (Pattern Support and Confidence) For sets $SX(R_i)$ and $CX(R_i)$ of seed examples and counterexamples and an input set S of sentences, a basic (or

Pattern	Relation	Computed confidence
{PhD dissertation at}	graduatedFrom	1.0
{doctorate at the, in, with}	graduatedFrom	0.57
{doctorate at the, in, with}	hasAcademicAdvisor	0.43
{attended the}	graduatedFrom	0.96
{dissertation supervised by}	hasAcademicAdvisor	1.0
{academic career at the }	facultyAt	1.0
{was awarded the, along with}	hasCollaborator	0.44
{and associate}	hasCollaborator	1.0
{is a fellow of}	hasProfessionalAffiliation	1.0
{is a member of the }	hasProfessionalAffiliation	0.33

Table 3.1: Example seed patterns with computed confidence values

n -gram-itemset) pattern q has $support(q) =$

$$\frac{|\{s \in S | \exists (e1, e2) \in SX(R_i) : q, e1, e2 \text{ occur in } s\}|}{|S|}$$

and $confidence(q) =$

$$\frac{|\{s \in S | \exists (e1, e2) \in SX(R_i) : q, e1, e2 \text{ occur in } s\}|}{|\{s \in S | \exists (e1, e2) \in SX(R_i) \cup CX(R_i) : q, e1, e2 \text{ occur in } s\}|}$$

Definition 3.5.5 (Seed Pattern Weight) An n -gram-itemset pattern q , for given $SX(R_i)$, $CX(R_i)$, and input set S of sentences, is called a seed pattern if both $support(q)$ and $confidence(q)$ are above specified thresholds. Pattern q is associated with a seed-pattern weight, set to

$$weight(q) = \alpha \times support(q) + (1 - \alpha) \times confidence(q)$$

In our experiments, we used only confidence values and disregarded support for the weighting ($\alpha = 0$). Table 3.1 depicts some example patterns we have learned along with their computed confidence values, for a few selected relation types.

Fact Candidates

The seed patterns are used to discover new fact candidates. We consider all sentences $s \in S$ that contain two entities (x, y) of appropriate types for R_i (e.g., a person and a university for the *hasAcademicAdvisor* relation) and whose subsequence p in $s = h x p y t$ in between the two entities x, y partially matches one of the seed patterns.

Definition 3.5.6 (Subsequence-Pattern Match) *The goodness of the match is quantified by the Jaccard similarity*

$$\text{sim}(p, q) = \frac{|\{n\text{-grams} \in p\} \cap \{n\text{-grams} \in q\}|}{|\{n\text{-grams} \in p\} \cup \{n\text{-grams} \in q\}|}$$

This approximate matching of p against all seed patterns q is efficiently implemented by lookups in an n -gram index constructed from the seed patterns. Optionally, the n -grams can be weighted such that we compute *weighted Jaccard similarity*.

We process all input sentences $s = h x p y t$ this way, and again perform frequent-itemset mining to concentrate on the set of patterns to those with support above a specified threshold. The output of this step is a multi-set of weighted triples $(x, y, p)[w]$ where (x, y) is a fact candidate, p is an n -gram-itemset pattern, and w is the highest pattern-matching similarity of p with any seed pattern q . Note that it is a multi-set rather than a set because the same candidate can be encountered several times.

Definition 3.5.7 (Fact-pattern Candidate Multi-set) *For given input set S and seed-pattern set Q , the fact-pattern candidate multi-set $C(S, Q)$ is:*

$$C(S, Q) = \{(x, y, p)[w] \mid \exists s \in S : s \text{ contains } x, y, p \wedge w = \max\{\text{sim}(p, q) \times \text{weight}(q) \mid q \in Q\}\}$$

Finally, we can aggregate the fact-pattern candidates in C , grouping them either by fact candidates, to compute a strength measure of the potentially new fact, or by patterns, to quantify the goodness of a pattern.

Definition 3.5.8 (Fact Candidate Weight) *For a fact candidate (x, y) , the aggregated weight is given by: $\text{weight}(x, y) = \sum\{w \mid (x, y, p)[w] \in C\}$.*

Definition 3.5.9 (N-gram-itemset Pattern Weight) *For an n -gram-itemset pattern p , the aggregated weight is: $\text{weight}(p) = \sum\{w \mid (x, y, p)[w] \in C\}$.*

We can interpret these weights as the (statistical) evidence that (x, y) is a valid fact and p is a good pattern for further extraction steps for R_i . Note that the two weights are quite different, as the aggregations are computed over different sets.

3.6 REASONING

Following the work on SOFIE [109], the PROSPERA system also uses constraints on hypotheses to prune false positives and improve precision. In contrast to [20, 21], we include constraints on the duality of patterns and facts, and we harness the rich knowledge about entity types provided by YAGO. In addition, we specify functional dependencies, inclusion dependencies, relation properties such as symmetry, antisymmetry, or inverse relations, as well as domain-specific constraints whenever possible. The constraints are manually specified upfront, but in all our experiments this was merely a matter of a few minutes. Typical constraints look as follows, with variables $p, e1, e2, e3$ for patterns and entities and given relations R, S, T :

$$\begin{aligned} & \text{occurs}(p, e1, e2) \wedge \text{type}(e1, \text{dom}(R)) \\ & \wedge \text{type}(e2, \text{range}(R)) \wedge \text{expresses}(p, R) \Rightarrow R(e1, e2) \quad // \text{pattern-fact duality} \end{aligned}$$

$$\begin{aligned} & \text{occurs}(p, e1, e2) \wedge \text{type}(e1, \text{dom}(R)) \\ & \wedge \text{type}(e2, \text{range}(R)) \wedge R(e1, e2) \Rightarrow \text{expresses}(p, R) \quad // \text{pattern-fact duality} \end{aligned}$$

$$\begin{aligned} & R(e1, e2) \wedge \text{type}(R, \text{function}) \wedge \text{different}(e2, e3) \\ & \Rightarrow \neg R(e1, e3) \quad // \text{functional dependency} \end{aligned}$$

$$R(e1, e2) \wedge \text{sub}(R, S) \Rightarrow S(e1, e2) \quad // \text{inclusion dependency}$$

$$R(e1, e2) \wedge \text{inv}(R, T) \Rightarrow T(e2, e1) \quad // \text{inverse relations}$$

$$T(e1, e2) \wedge \text{inv}(R, T) \Rightarrow R(e2, e1) \quad // \text{inverse relations}$$

For the actual reasoning procedure, the constraints are *grounded* to produce *clauses*.

Definition 3.6.1 (Grounded Constraints and Clauses) *A constraint is a first-order logic formula of the form $\forall x_1 \forall x_2 \dots \forall x_k f(x_1, \dots, x_k)$. Constraints are grounded by substituting all meaningful constants – concrete patterns and entities – into the constraint formulas, thus providing us with a set of propositional-logic clauses. Converting the resulting propositional logical formulas into conjunctive normal form provides us with clauses of the form: $P_1 \vee P_2 \dots P_k$, where each P_i is a positive or negative literal consisting of a predicate with constants as arguments.*

We can handle clauses with several negative literals, whereas rule-induction methods (e.g., the one used in [20, 21]), are typically restricted to Horn clauses. For example, for the *graduatedFrom* relation (assuming that it were a function: it refers only to Ph.D. degrees and one can obtain a Ph.D. only from one university), the grounding procedure generates clauses such as:

$$\begin{aligned} & \text{occurs}(\text{ and PRP alma mater, Barbara_Liskov, Stanford_University }) \\ & \wedge \text{expresses}(\text{ and PRP alma mater, graduatedFrom }) \\ & \Rightarrow \text{graduatedFrom}(\text{ Barbara_Liskov, Stanford_University }) \end{aligned}$$

$$\begin{aligned} & \text{graduatedFrom}(\text{ Barbara_Liskov, Stanford_University }) \\ & \Rightarrow \neg \text{graduatedFrom}(\text{ Barbara_Liskov, UC_Berkeley }) \end{aligned}$$

$$\begin{aligned} & \text{graduatedFrom}(\text{ Barbara_Liskov, UC_Berkeley }) \\ & \Rightarrow \neg \text{graduatedFrom}(\text{ Barbara_Liskov, Stanford_University }) \end{aligned}$$

Note that the grounding already evaluates predicates whose truth value can be decided directly. For example, the *different* predicate between two entities is directly set to true or false, thus simplifying the resulting clauses. Most importantly, the type predicates are evaluated at this stage, too. For entities that do not obey the type signature of the relation at hand, the antecedent of the clause evaluates to false so that the entire clause can be eliminated. This massive pruning of clauses from the hypotheses space greatly reduces the reasoner’s load. The efficiency gain is possible because of the rich type information about entities that YAGO provides. Such optimizations were not possible in earlier work on reasoning-based information extraction such as [37]. The recent work of [20, 21] considered type information, too, by coupling the pattern-based learners for binary relations with those of unary ones. Note, however, that this was at the level of non-canonical (often ambiguous) names rather than uniquely identified entities; so it is not as clean and powerful as our rigorous typing at the entity level.

The grounded clauses are weighted, and then we finally run the Weighted-MaxSat reasoner of [109]. This computes truth values for all hypotheses on the *expresses* and *R* predicates for all relations *R* and all instantiated constants (patterns and entities), such that the total weight of the clauses that are satisfied by this truth-value-assignment becomes as large as possible. The algorithm can only approximate the maximum of this objective function, given that MaxSat is NP-hard and our algorithm runs on hundred thousands of clauses with ten thousands of variables.

The weights of clauses are derived from the pattern-confidence measures computed in the pattern analysis phase. This is a major departure from earlier work on reasoning-based information extraction: SOFIE used uniform weights except for entity disambiguation [109] and the work with Markov logic networks advocated setting weights by frequency analysis of the fact candidates (the “uncertain database”) [37].

Definition 3.6.2 (Clause Weight) *In PROSPERA, we associate the antecedent of a clause with a confidence weight about its constituent literals (elementary logical atoms). Specifically, for clauses of the form:*

$$\text{occurs}(p, e1, e2) \wedge \text{expresses}(p, R) \Rightarrow R(e1, e2),$$

we use the confidence weight of the pattern p as the weight of the entire clause. For clauses of the form

$$\text{occurs}(p, e1, e2) \wedge R(e1, e2) \Rightarrow \text{expresses}(p, e1, w2),$$

we analogously use the confidence weight of the fact candidate $R(e1, e2)$. The frequency of observing $p, e1, e2$ together (the *occurs* predicate) is irrelevant, as this would unduly boost frequent observations regardless of their quality.

Our experiments show that the reasoner becomes much more robust by using the above weights, which are essentially dependent on seed facts (and the derived seed patterns). Clauses derived from functional dependencies, relational properties, or domain-specific consistency rules are given uniform weights.

3.7 DISTRIBUTED IMPLEMENTATION

To scale out our knowledge-harvesting system, we adopted the MapReduce programming model [31, 121] based on the abstractions of *mappers* and *reducers*. Mappers specify the computation to be performed on each input record. Reducers specify how the output of the mappers is aggregated to generate the final results. MapReduce computations are based on key-value pairs. Mappers work on input key-value pairs and generate intermediate key-value pairs. Reducers consume intermediate pairs, with the same intermediate keys being passed to the same reducer. Reducers aggregate intermediate keys to emit output key-value pairs.

We have developed MapReduce algorithms for the three main phases of our architecture. We used the Hadoop open source implementation [121] and the HDFS distributed filesystem [128].

Pattern Gathering

Parsing documents for pattern gathering is trivially parallelizable as each document is scanned independently. No coordination is required between concurrent worker tasks. The input to the mappers are document identifiers (keys) and the corresponding document contents (values).

The mapper performs checks on the sentences of the document, emitting triples of the form (e_1, p, e_2) for any pair of interesting entities e_1 and e_2 . Additional processing of sentences, such as generating part-of-speech tags, is also performed in the mapper. Here the reducer merely serves the purpose of sorting and aggregating the emitted triples.

Pattern Analysis

The pattern analysis phase computes statistical measures for seed patterns and uses these to generate fact candidates. The results of the pattern analysis phase are accomplished by a sequence of MapReduce algorithms; here we focus on the major tasks and how to distribute/parallelize them.

Generate N-gram-itemset Patterns. The n-gram-itemset patterns are the primary representation on which pattern-similarity computation is based. Thus, the

```

1. FUNCTION map( $i, P_i$ )
2.   List  $N \leftarrow$  generateNgrams( $P_i$ )
3.   FOR  $n_i \in N$  DO
4.     emit( $n_i, 1$ )

1. FUNCTION reduce( $n_i, [v_1, v_2, v_3, \dots]$ )
2.   support  $\leftarrow 0$ 
3.   FOR  $v_i \in [v_1, v_2, v_3, \dots]$  DO
4.     support  $\leftarrow$  support +  $v_i$ 
5.   IF support  $\geq$  MINSUPPORT
6.     emit( $n_i, support$ )

```

Figure 3.3: MapReduce pseudo-code for frequent n-gram-itemset mining

first task is to convert the previously collected raw patterns into this format. This entails identifying frequently co-occurring n-grams within the basic patterns via frequent-itemset mining [1]. To reduce the size of the input data to subsequent algorithms, we introduce a preprocessing step to perform dictionary encoding by replacing words and patterns with integer identifiers.

The pseudo-code for computing frequent itemsets is shown in Figure 3.3. The input to the mapper consists of the key-value pair of the pattern identifier (key) and the pattern itself (value). For each input pattern, mappers generate constituent n-grams and emit, for each n-gram, an intermediate key-value pair consisting of the (dictionary-compressed) n-gram as the key and a support of 1 as the value. The reducers gather support counts for any given n-gram and sum them up to obtain the final support counts. Only those n-grams whose support is above the specified values are emitted. Note that sequential algorithms for frequent-itemset mining are typically optimized to eliminate non-frequent itemsets as early as possible. When generating frequent itemsets of cardinality (or length in our case) k , the algorithm first prunes all infrequent $(k-1)$ -grams. In contrast, our MapReduce algorithm greedily generates all itemsets and does batch pruning in the reducers. This is advantageous because 1) we are only interested in relatively short n-grams, typically 3-grams, and 2) the MapReduce paradigm is designed for batch processing and works best if coordination and communication across worker tasks is kept to a minimum.

Once we have the frequent n-gram itemsets, a second MapReduce algorithm (not shown here), is used to rewrite patterns into a form with frequent n-grams only, disregarding infrequent ones. This way we end up with n-gram-itemset patterns.

Compute Seed Pattern Confidence Values. Once all the patterns are in n-gram-itemset representation, we need to identify the seed patterns and compute their confidence values. To this end, we need to determine how often the pattern co-occurs with seed facts and how often it co-occurs with counter-seeds. We have developed two MapReduce algorithms for this purpose. The first one, shown in Figure 3.4, identifies seed patterns and tracks pattern-occurrence information. In each mapper, the (e_1, p, e_2) triples from the pattern-gathering phase are processed to test if p is a seed pattern. The mappers emit intermediate key-value pairs with the pattern identifier as the key and the seed occurrence as the value. The reducers combine all seed occurrences belonging to one pattern and emit all seed

```

1. FUNCTION map( $i, [e_1, p, e_2]$ )
2.   IF isSeedPattern( $p$ )
3.     FOR  $r \in R$  DO
4.       SeedOccurrence  $O \leftarrow [r, e_1, e_2]$ 
5.       emit( $p.id, O$ )

1. FUNCTION reduce( $p.id, [O_1, O_2, O_3, \dots]$ )
2.   List  $L \leftarrow \{ \}$ 
3.   FOR  $O \in [O_1, O_2, O_3, \dots]$  DO
4.      $L.append(O)$ 
5.   emit( $p.id, L$ )

```

Figure 3.4: MapReduce pseudo-code for seed pattern confidence

occurrences of every seed pattern. A second MapReduce algorithm (not shown here) uses this data to compute pattern confidence values.

Generate Fact Candidates. The large majority of the $[e_1, p, e_2]$ triples from the pattern gathering have patterns p that do not precisely match any seed pattern. To identify new fact candidates and quantify their statistical evidence, we conceptually compute the similarity of p with all partially matching seed patterns q based on the Jaccard coefficient of the corresponding n-gram sets (see Section 3.5)

The easiest implementation would be an exhaustive algorithm where a mapper computes the similarity of a given pattern with *all* seed patterns and then emits the best (partial) match. However, this would have high computational costs because of many unnecessary comparisons. To accelerate the computation, we first build an inverted index on the n-grams of the seed patterns and use it to compute similarity scores more efficiently. For building the index, we follow standard MapReduce practice [31]. The optimized algorithm is shown in Figure 3.5.

The mappers consume non-seed patterns, with the pattern identifier as the key and the n-gram-itemset as the value. Each mapper first loads its relevant partition of the seed n-gram index and pattern confidence values into memory. Hadoop allows mappers to preserve state across different input pairs, therefore this information is loaded only once during the Hadoop job initialization. The mapper uses the index to compute matches between seed patterns and non-seed patterns. For each such match, the similarity score between the seed pattern and the non-seed pattern p_i is computed and added to a priority queue. The mapper then emits an output key-value pair consisting of the pattern identifier and the best matching seed pattern. This information is then passed down to the reasoner which makes the final decision on the goodness of the patterns during a given iteration.

Reasoning

To parallelize the MaxSat-based reasoning, the hypotheses about fact candidates and pattern goodness are formulated as a graph.

```

1. FUNCTION map( $i, p_i$ )
2.    $I \leftarrow \text{loadSeedNgramIndex}()$ 
3.    $C \leftarrow \text{loadSeedPatternConfidenceValues}()$ 
4.   PriorityQueue  $Q \leftarrow \{ \}$ 
5.   List  $H \leftarrow \text{computeHits}(p, I, C)$ 
6.   FOR  $h \in H$  DO
7.      $h.\text{similarity} = \text{computeSimilarity}(p_i, h.\text{seedPattern})$ 
8.      $Q.\text{insert}(h, h.\text{similarity})$ 
9.   emit( $Q.\text{removeMin}()$ )

```

Figure 3.5: Mapper pseudo-code for pattern similarity and fact-candidate extraction.

Definition 3.7.1 (Hypotheses Graph) A hypotheses graph $G = (V, E)$, for a set of propositional logic clauses, is an undirected graph consisting of a node set V representing fact candidates and patterns and an edge set E representing edges between two nodes if they appear in a joint clause.

Definition 3.7.2 (K-cut Graph Partitioning) A k -cut partitioning of a hypotheses graph is a partitioning into k sub-graphs (of approximately equal size), such that the number of edges that connect vertices in different partitions is minimized.

Note that partitioning the graph may now disregard some constraints, but as MaxSat is an NP-hard problem our solution is approximate anyway. The fewer cross-partition edges are cut, the more constraints are preserved by the parallelized reasoning. Generating the graph is specified as a MapReduce job, the graph is then partitioned into k partitions, and the partitions are processed in parallel by reasoners on different compute nodes of the distributed platform.

The minimum-cut graph partitioning problem is also NP-complete and there is a plethora of approximate algorithms for it. We employed a randomized, two-phase graph partitioning algorithm, based on methods by [62] and [63].

Graph Partitioning Phase 1. Phase one coarsens the graph into a smaller graph that is a good representation of the original graph. The coarsening reduces the size of graph by edge contraction until the graph is small enough. The basic technique is to pick an edge connecting vertices u and v and collapse the two vertices: a new vertex w replaces u and v . Edges previously linking u and v to other vertices are updated to point to the new vertex w . If both u and v have edges to another vertex z , then the weight of the edge from w to z is the sum of the weights of the two edges. This helps to ensure that a balanced partitioning of the smaller graph is also an approximately balanced partitioning in the original graph. In picking the edges to contract, we heuristically favor edges that contribute more to the overall min-cut objective function. These are the heavy edges in the coarsened graph. Initially, all edges have the same uniform weight, but as vertices are collapsed, some edges obtain higher weights. In each step, we randomly select a vertex and then choose its incident edge with the highest weight for contraction. This guards the edge from being cut in the second phase of the overall algorithm.

Graph Partitioning Phase 2. Phase two partitions the coarser graph and projects the resulting partitions back into the original graph. Once a coarsened graph with a specified maximum number of vertices is obtained, it is then directly partitioned into k partitions. We use graph-growing heuristics [63] to derive k partitions from the coarsened graph. For each k , we randomly select a vertex and grow a region around it until $|V|/k$ vertices are included, picking vertices that lead to smaller increase in the weight of the edges that are cut.

In our experiments, partitioning the graph this way did not notably affect the output quality of the Weighted-MaxSat solver, which is only an approximation algorithm anyway. The time for graph partitioning was short, usually 5 minutes at most for graphs with several 100,000's of vertices.

3.8 EVALUATION

We carried out experiments to evaluate both the single machine version of PROSPERA and the distributed version.

Small-Scale Experiments

In the small scale experiments, we aimed to evaluate PROSPERA's performance with respect to the improvements it makes on the SOFIE framework on which it is based. We carried out experiments to extract academia-related information. The corpus was generated by crawling the homepages of the most prolific authors from DBLP, then augmenting these with articles of scientists from Wikipedia. Additionally, names of scientists were used to query Google for further documents. The resulting corpus consists of 87,470 documents. The knowledge base used in the experiments is the YAGO [108] ontology.

To quantify how the various aspects of PROSPERA affect performance, we evaluated the *hasAcademicAdvisor* relation. Table 3.2 shows the results.

PROSPERA has the highest recall at high precision. SOFIE produced many extractions but with low precision. The *hasAcademicAdvisor* relation is not straightforward to extract because it can be expressed by patterns that may be misleading. For example, the pattern, "x worked with y" may or may not indicate that y was the doctoral advisor of x. These misled SOFIE but PROSPERA withstood them as it identifies these cases through pattern occurrence statistics. The two systems extracted more or less the same facts; however each system also extracted some tuples the other did not. For example both systems extracted the pair (*Jeffrey Shallit, Manuel Blum*), but only PROSPERA extracted the pair (*Serge Lang, Emil Artin*), whereas only SOFIE extracted the pair (*Ravi Sethi, Jeffrey Ullman*).

Consistency checking plays a significant role in ensuring high precision as reflected in the results of the PROSPERA-NoReasoner method. The reasoner thus acts as a well-placed filter, performing type checking as well as ensuring that the logical rules are upheld. The PROSPERA-NoCounterExamples method shows

Method	# extractions	Precision
SOFIE	1,845	22%
PROSPERA	372	83%
PROSPERA-NoReasoner	22,340	1.9%
PROSPERA-NoCounterExamples	404	79%
PROSPERA-Unweighted	338	83%
Method	# fact candidates	Runtime (min)
SOFIE	105,016	122
PROSPERA	22,340	35
PROSPERA-NoReasoner	n/a	22
PROSPERA-NoCounterExamples	24,328	35
PROSPERA-Unweighted	24,328	35

Table 3.2: Performance for the *hasAcademicAdvisor* relation

the impact of counter-examples. Without the counter-examples, even weak patterns may lead to extractions; this degrades precision slightly. The PROSPERA-Unweighted method shows the impact of the weights. Disregarding pattern weights altogether results in slightly reduced recall. This is attributed to the fact that the weights guide the reasoner to the correct answer. Without the weights there may be misleading cases, causing the reasoner to reject facts that might be true. The number of fact candidates indicates the number of candidates passed on to the reasoner. It can be seen in Table 3.2 that all variations of PROSPERA provide considerable pruning of fact candidates and this results in shorter execution times compared to SOFIE.

Comparisons were also carried out using various other relations, the results are shown in Table 3.3. PROSPERA has high precision across all the relations whereas SOFIE’s precision varies widely across relations. Furthermore, for the *hasCollaborator* and *hasProfessionalAffiliation* relations, PROSPERA has much higher recall. This is because these two relations had the fewest number of seeds and the generalization capability of patterns in PROSPERA enabled further instances to be discovered without requiring exact matches between patterns. For the *graduatedFrom* and *facultyAt* relations, SOFIE’s recall suffers because these two relations have overlapping instances, since people can be faculty members at the institutions where they graduated from. Here again PROSPERA is robust to this scenario. PROSPERA has the same precision as SOFIE for the *hasWonPrize* relation. This is because this relation is typically expressed with the same patterns, hence the exact pattern matching in SOFIE works well. PROSPERA has a slightly lower recall than SOFIE for this relation, primarily because weak seed patterns that do not meet the requirements in PROSPERA were dropped.

To quantify the impact of the number of seeds used, we evaluated PROSPERA and SOFIE’s performance for the *hasAcademicAdvisor* for varying numbers of seeds. Table 3.4 shows the results.

PROSPERA has high precision and decent recall even when only a small number of seeds are used. SOFIE, on the other hand has high precision when only a

	Relation	# extractions	Precision
PROSPERA	hasAcademicAdvisor (person, person)	372	83%
	hasCollaborator (person, person)	122	91%
	facultyAt (person, university)	1,274	94%
	graduatedFrom (person, university)	1,310	89%
	hasProfessionalAffiliation (person, organization)	107	90%
	hasWonPrize (person, award)	1,309	99%
SOFIE	hasAcademicAdvisor (person, person)	1,845	22%
	hasCollaborator (person, person)	8	100%
	facultyAt (person, university)	3,147	49%
	graduatedFrom (person, university)	5,088	56%
	hasProfessionalAffiliation (person, organization)	7	100%
	hasWonPrize (person, award)	1,553	99%

Table 3.3: Performance for all relations

few seeds are used but has very low recall in these cases. When many seeds are used, there is a high chance of noisy patterns occurring with a few instances. For example, for the *hasAcademicAdvisor* relation, SOFIE's precision degrades, PROSPERA still yields high precision.

We also compared PROSPERA to the SNOWBALL[3] system, using one of their experiments (for which a non-copyright-protected part of the data was available[108]). In this test run, with the goal of extracting the headquarters of companies, PROSPERA reached 85% for a recall of 42 newly extracted, correct facts, SOFIE also

	# seeds	# extractions	Precision
PROSPERA	15	48	89%
	50	115	81%
	212	372	83%
SOFIE	15	4	100%
	50	131	31%
	212	1,845	22%

Table 3.4: Precision and recall for the *hasAcademicAdvisor* relation for varying numbers of seeds

extracted 42 correct facts with 91% precision, whereas the original SNOWBALL reached 57% precision with 37 correct facts.

In general, not many information-extraction systems are publicly available for comparative experiments. Moreover, many results in the literature cannot be reproduced in a full experiment because the publications do not disclose sufficient details about their experiments (e.g., the datasets and chosen seeds). Therefore, we cannot present a broader set of comparisons with other systems. Full details of our experiments were published at www.mpi-inf.mpg.de/yago-naga/prospere/.

Large-Scale Experiments

Setup

Large-scale experiments were carried out on the English part of the ClueWeb09 corpus [26], which consists of 500 million English-language Web pages. Evaluations reported here were restricted to binary relations between entity pairs. We focused on two domains: 9 relations from the sports domain and 5 relations from the domain of academic relationships.

The sports domain was chosen in order to compare PROSPERA to the results of the *NELL* (Never Ending Language Learning) experiment reported in [20], which had strong coverage of the sports relations and – very laudably – made all relevant data available on the ReadTheWeb site. To our knowledge, *NELL* is so far the largest and most ambitious experiment along these lines, with online data against which we could meaningfully compare our approach. We used the very same input as *NELL*: 10–15 seed facts and 5 counter-examples for each relation. In contrast to *NELL*, we did not have any human intervention during our runs. Also, *NELL* allowed 5 manually specified seed patterns as a-priori input, whereas PROSPERA used only seed facts and determined patterns autonomously.

The sports domain has no constraints other than type constraints; there are not even any functional dependencies. The academic domain, on the other hand, is an interesting choice as it has sophisticated constraints posing a stress-test to the reasoning phase in our PROSPERA system. For the academic relations, we used seeds obtained from the YAGO ontology. All instances for a given relation already in the ontology were treated as seeds. Counter-seeds were derived from instances of other relations in YAGO.

Relation	# Extractions		
	PROSPERA-6	NELL-6	NELL-66
AthletePlaysForTeam	14,685	29	456
CoachCoachesTeam	1,013	57	329
TeamPlaysAgainstTeam	15,170	83	1,068
TeamWonTrophy	98	29	397
AthletePlaysInLeague	3,920	2	641
TeamPlaysInLeague	1,920	62	288
AthleteWonTrophy	10	n/a	n/a
CoachCoachesInLeague	676	n/a	n/a
TeamMate	19,666	n/a	n/a

Table 3.5: Performance comparison between PROSPERA and NELL on sports relations: number of extractions

All experiments were performed on a Hadoop (0.20.1) cluster with 10 server-class machines. Each machine has a Dual-Xenon E5530 2.4 GHz processor (16 physical cores), 48 GB RAM (DDR3), 1.5 TB iSCSI storage, and 1 Gbit Ethernet interconnect. The NELL experiment ran on the Yahoo! M45 supercomputing cluster, but no statements were given about the number of nodes used and their utilization.

Performance metrics of interest are *recall*, *precision*, and *run-times*. Here recall refers to the number of extracted facts which are returned by each of the knowledge-harvesting systems, as there is no way of estimating the total number of truly correct facts that appear (in latent form with natural language) in the entire corpus. Precision was estimated by sampling the harvested facts and having a human judge assess their correctness. As our runs yielded much higher recall than NELL, we also ranked the resulting facts by their confidence and additionally determined the *precision@1000* for the 1000 highest-ranked facts of the largest relations. All precision assessments are based on 50 randomly sampled facts for each relation.

All data on the experiments reported here are made available as supplementary material on the Web site www.mpi-inf.mpg.de/yago-naga/prospERA/.

Scalability Experiment (Sports Relations)

The scalability experiment aimed to evaluate the performance of our approach on large and noisy data for all three performance metrics. Tables 3.5 though 3.7 show the results of the experiment on sports relations. PROSPERA ran 6 iterations, and NELL ran 66 in total. We compare PROSPERA-6 (6 iterations) against NELL-6 (first 6 iterations) and NELL-66 (all 66 iterations). For the first four relations, both precision and recall numbers for NELL are given in [20] and its supplementary material. NELL was run on the other relations as well, but no recall/precision numbers were given.

Relation	Precision		
	PROSPERA-6	NELL-6	NELL-66
AthletePlaysForTeam	82%	100%	100%
CoachCoachesTeam	88%	100%	100%
TeamPlaysAgainstTeam	89%	96%	99%
TeamWonTrophy	94%	88%	68%
AthletePlaysInLeague	94%	n/a	n/a
TeamPlaysInLeague	89%	n/a	n/a
AthleteWonTrophy	90%	n/a	n/a
CoachCoachesInLeague	99%	n/a	n/a
TeamMate	86%	n/a	n/a

Table 3.6: Performance comparison between PROSPERA and NELL on sports relations : precision

Relation	Precision@1000
	PROSPERA-6
AthletePlaysForTeam	100%
CoachCoachesTeam	n/a
TeamPlaysAgainstTeam	100%
TeamWonTrophy	n/a
AthletePlaysInLeague	n/a
TeamPlaysInLeague	n/a
AthleteWonTrophy	n/a
CoachCoachesInLeague	n/a
TeamMate	100%

Table 3.7: Performance of PROSPERA: precions@1000

PROSPERA has orders-of-magnitude higher numbers of extractions compared to NELL-6, as shown in Table 3.5. Even NELL-66 still had substantially fewer results than PROSPERA. On the other hand, NELL had precision close to 100% for most of these relations, except for the *TeamWonTrophy*. For this relation NELL-6 has precision of 88% and further degradation can be seen in NELL-66 at 68%. Generally, [20] reported that NELL’s precision would gradually go down with larger numbers of iterations. In contrast, PROSPERA’s precision was consistently good and hardly varied across iterations. For the *TeamWonTrophy* relation, PROSPERA outperformed NELL on precision (see Table 3.6), for the other relations is was somewhat worse than NELL but still well above 80%. Note, however, that the precision across all extractions is misleading here, as PROSPERA returned a much higher number of facts. We also evaluated the *precision@1000* for the largest relations, as shown in Table 3.7. Here, PROSPERA achieved 100%. So overall, our approach essentially achieved the same precision as NELL while giving much higher recall.

Relation	PROSPERA
AthletePlaysForTeam	(Ben_Gordon, Chicago_Bulls)
TeamPlaysInLeague	(Chicago_Bulls, National_Basketball_Association)
AthletePlaysForTeam	(Jason_Giambi, New_York_Yankees)
TeamPlaysAgaintsTeam	(San_Francisco_Giants, New_York_Yankees)
AthletePlaysForTeam	(Ben_Graham_(footballer), Arizona_Cardinals)
AthletePlaysForTeam	(Edgar_Gonzalez_(infielder), St._Louis_Cardinals)
TeamMate	(Metro_Prystai, Jim_Henry_(ice_hockey))

Table 3.8: Sample output from PROSPERA

Relation	NELL
AthletePlaysForTeam	(Ben Gordon, Bulls)
TeamPlaysInLeague	(Chicago Bulls, NBA)
AthletePlaysForTeam	(Jason Giambi, Yankees)
TeamPlaysAgaintsTeam	(Giants, New York Yankees)
AthletePlaysForTeam	n/a
AthletePlaysForTeam	n/a
TeamMate	n/a

Table 3.9: Sample output from NELL

PROSPERA produces high-quality extractions in canonical form: relational facts between disambiguated entities. For example, across all extractions, all relational facts involving the team *Chicago Bulls*, always use the same consistent identifier for this team. This is different from the NELL output which has facts referring to names, returning, for example, both the *Bulls* and the *Chicago Bulls*. Our approach includes entity disambiguation, thus recognizing, based on the context of the name occurrence, if the string “Bulls” refers to the entity *Chicago_Bulls* or some other team with the ending “Bulls”. Table 3.8 shows disambiguated output from PROSPERA in comparison to NELL extractions (Table 3.9).

The total run-time of PROSPERA was about 2.5 days for the 6 iterations on the sports domain. This is in contrast to NELL’s 6 or 66 days for the first 6 or all iterations, respectively. Note that NELL ran on a much larger cluster but also extract other relations and categories that we did not consider in our experiments. So the run-time numbers are not comparable. Nevertheless, especially in view of our much higher recall, the PROSPERA run-times look favorable.

In this experiment, the MaxSat-based reasoning constitutes only a small percentage of the total run-time. This is illustrated in Figure 3.6(a) where in every iteration, pattern analysis (the bottom part) took between 4 to 6 hours while reasoning took less than an hour. Not shown in Figure 3.6(a) is the run-time of the preprocessing for pattern gathering, which took approximately 20 hours. This time is included in the total figure of 2.5 days for gathering and 6 iterations of analysis and reasoning. Figure 3.6(b) shows that the number of extractions consistently increases over iterations; running more iterations would probably lead to further extractions.

To determine how individual components of our approach contribute to performance, we ran additional measurements with different variants of PROSPERA,

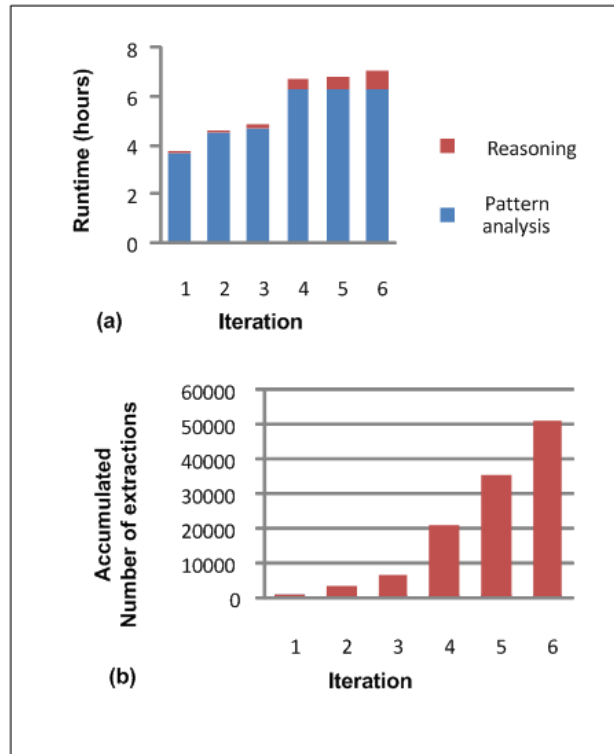


Figure 3.6: PROSPERA runtimes on the sports domain (a) and number of extractions across iterations (b).

selectively disabling one component at a time. For this study, we ran only 2 iterations. As the results in Table 3.10 show, reasoning plays a significant role in ensuring high precision. Without the reasoner, precision would be only 31%. Reasoning about the quality of the patterns ensures that patterns whose quality is uncertain do not lead to extractions. This initially leads to much lower recall, but for a good pattern, subsequent iterations gather enough support for the pattern. Thus, such a pattern is eventually accepted by later iterations of the reasoner, leading to increased recall over successive iterations. For example, in the first iteration, starting with the initial seed facts, the pattern “*would beat the*” only had a confidence weight of 0.5 for expressing the *TeamPlaysAgainstTeam* relation. By the fourth iteration, evidence in support of this pattern grew and its confidence weight increased to 0.96. Note that all this is fully automated in PROSPERA, whereas NELL had some manually designed seed patterns and used a small amount of human supervision after each iteration.

The confidence weights also play a vital role in the quality of our output, both for pruning bad patterns and for providing better statistical evidence to the reasoner about which candidates are more likely to be true. Without the use of weights, precision drops to 73%. On noisy Web data like the ClueWeb09 corpus, the new notion of n-gram-itemset patterns with confidence weights pays off well.

To determine speedup obtained by parallelizing the reasoner, we reduced the number of reasoners by half and measured run-times for reasoning in the first iteration. Using half the reasoners, 5 in total (on 5 nodes of the cluster), took

Method	# extractions	Precision	Runtime (hours)
Full-PROSPERA	3,174	90%	8.37
NoReasoner	157,610	31%	8.22
Unweighted	8,429	73%	8.32

Table 3.10: PROSPERA variants on sports relations (2 iterations).

7.6 minutes compared to 3.5 minutes when using 10 reasoners. This suggests a slightly super-linear speedup of 2.2, which is attributed to the fact that partitioning the candidate graph reduces the search space of the reasoner, resulting in faster execution times.

Constraints Experiment (Academic Relations)

As the sports-domain experiment did not have any advanced constraints, we carried out a second experiment that aimed to stress-test the constraint reasoning aspect of PROSPERA on five academic relations. We specified the following constraints in first-order logic:

- a student can have only one alma mater that she/he graduated from (with a doctoral degree);
- a student can have only one doctoral advisor (who had this role officially);
- the advisor of a student must have had a position at the university from which the student graduated;
- the advisor of a student must be older than her/his student.

We ran PROSPERA for two iterations only, using seed facts from the YAGO ontology; the results are shown in Table 3.12. Both the number of extractions and precision are high with the exception of the *hasAcademicAdvisor* relation which returned only few extractions with mediocre precision. Here, our approach of consistently referring to canonical entities became unfavorable, as we could accept only facts where both student and advisor are known to YAGO (or Wikipedia, on which YAGO is based). The number of pairs that fulfilled this strict requirement and also appeared in the ClueWeb09 corpus was too low.

In general, having rich constraints significantly improved precision as shown in Table 3.13. Without the reasoner, we obtained several birth dates for one person and in some cases more than five advisors for one person, a situation highly unlikely in reality. Without the reasoner precision dropped to 25%. Table 3.11 shows a few sample results for facts and their supporting patterns.

This experiment had longer run-times (ca. 18 hours for 2 iterations) due to the use of domain-specific constraints for the reasoner, and also because of the larger number of seed facts obtained from YAGO for the first iteration. The experiment again showed considerable speedup obtained from parallelizing the reasoner, with 5 reasoners (on 5 nodes of our cluster) taking 7.1 hours whereas 10 reasoners only took 2.7 hours.

Relation	PROSPERA
facultyAt	(Richard_Axel, Columbia_University) <i>'s group at</i>
facultyAt	(Richard_Fateman, University_of_California,_Berkeley) <i>and colleagues at the</i>
graduatedFrom	(Albert_Einstein, University_of_Zurich) <i>earned a doctorate from the</i>
hasAcademicAdvisor	(Miguel_Rolando_Covian, Bernardo_Houssay) <i>student of</i>
hasAcademicAdvisor	(Frank_Wilczek, David_Gross) <i>shared the, in, with his thesis advisor</i>

Table 3.11: Sample output from PROSPERA

Relation	# extractions	Precision	Precision @1000
bornOnDate	40,962	92%	97%
facultyAt	4,394	96%	98%
graduatedFrom	1,371	81%	n/a
hasAcademicAdvisor	46	75%	n/a
hasWonPrize	4,800	91%	100%

Table 3.12: Extracted facts and estimated precision for academic relations obtained from two iterations of PROSPERA.

Method	# extractions	Precision	Runtime (hours)
Full-PROSPERA	51,573	92%	18.3
NoReasoner	773,721	25%	13.1

Table 3.13: PROSPERA variants on academic relations

Discussion

The presented experiments are a proof of concept for the scalability of our approach. Each iteration of the analysis and reasoning phases takes only a few hours, with the Hadoop-based, parallelized PROSPERA system. In particular, even in the more demanding setting of the constraints-rich academic relations, the graph-partitioning-based distributed reasoner is fast enough to avoid bottlenecks.

In all experimental results, precision is very high and matches up against the high quality of the NELL results. In terms of recall, we achieved a much larger number of extracted facts, for some relations even orders-of-magnitude higher. Here the combination of richer patterns, statistically informative weights for clauses, and the resulting better input for the reasoner proved to be vital. As our studies with selectively disabling specific components of PROSPERA show, all building blocks

are essential and their careful integration is key to the overall performance.

Regarding the quality of the output of the knowledge-harvesting systems, let us again emphasize that the facts by PROSPERA refer to canonical entities, whereas NELL's output refers to potentially ambiguous non-canonical names. When considering these approaches for further extending near-human-quality knowledge bases such as DBpedia, Freebase, or YAGO, this clean entity-level output is an important asset. In our experiments, the name-to-entity mapping heuristics worked very well. When sampling the accepted facts, we came across very few disambiguation errors, they had negligible influence on the overall precision.

3.9 SUMMARY

This chapter has addressed the goal of large-scale fact extraction from Web sources. It extended and improved prior work by projects like *KnowItAll*, *StatSnowball*, *ReadTheWeb*, and *YAGO-NAGA* in several ways. First, we introduced a new notion of n-gram-itemset patterns and associated confidence statistics. Second, we showed how to utilize pattern statistics for MaxSat-based reasoning with informative clause weights, and we developed techniques for making the previously expensive reasoning much more efficient and parallelizable. Third, we integrated all building blocks into a MapReduce-based distributed system architecture for scalable knowledge harvesting that can achieve both high precision and much higher recall than prior methods. In large-scale experiments, we compared ourselves against the latest state-of-the-art competitor and demonstrated significant gains. Our experimental data is accessible on the Web site: www.mpi-inf.mpg.de/yago-naga/prospERA/.

We note that PROSPERA deals with a pre-specified set of relations along with the relations seeds. The next chapter addresses this limitation.

Relation Extraction with PATTY

The previous chapter introduced PROSPERA, a system for Web-scale fact extraction. One of the limitations of PROSPERA is that it extracts only facts pertaining to a small set of pre-specified relations. This chapter presents PATTY, a system for extracting a comprehensive set of relations from a text corpus. PATTY learns textual patterns that denote binary relations. The patterns are semantically typed and organized into pattern synonym sets (synsets) and into a subsumption taxonomy. From Wikipedia, PATTY learned 350,569 pattern synsets. A sample of randomly selected patterns showed a pattern accuracy of 84.7%.

4.1 MOTIVATION

WordNet [43] is one of the most widely used lexical resources in computer science. It groups nouns, verbs, and adjectives into sets of synonyms, and arranges these synonyms in a taxonomy of hypernyms. WordNet is limited to single words. It does not contain entire *phrases* or *patterns*. For example, WordNet does not contain the pattern:

X is romantically involved with Y.

Just like words, patterns can be synonymous, and they can subsume one another. The following patterns are synonymous:

X is romantically involved with Y

X is dating Y.

Both are subsumed by:

X knows Y.

Patterns for relations are a vital ingredient for many applications, including information extraction and question answering. If a large-scale resource of relational patterns were available, this could boost progress in NLP and AI tasks.

Yet, existing large-scale knowledge bases are mostly limited to abstract binary relationships between entities, such as “bornIn” [8, 13, 86, 108]. These do not

correspond to real text phrases (e.g., “a native of”, “her birthplace”, etc). Only the ReVerb system [41] yields a larger number of relational textual patterns. However, no attempt is made to organize these patterns into synonymous patterns, let alone into a taxonomy. Thus, the patterns themselves do not exhibit semantics.

4.2 CONTRIBUTION

PATTY’s goal is to systematically compile relational patterns from a corpus, and to impose a semantically typed structure on them. The result we aim at is a WordNet-style taxonomy of binary relations. In particular, we aim at patterns that contain *semantic types*, such as $\langle \text{singer} \rangle \text{ sings } \langle \text{song} \rangle$. We also want to automatically generalize syntactic variations such as:

sings her $\langle \text{song} \rangle$
and *sings his* $\langle \text{song} \rangle$,

into a more general pattern:

sings [prp] $\langle \text{song} \rangle$

with POS tag [prp]. Analogously but more demandingly, we want to automatically infer that the above patterns are semantically subsumed by the pattern:

$\langle \text{musician} \rangle \text{ performs on } \langle \text{musical composition} \rangle$,

with more general types for the entity arguments in the pattern.

Compiling and organizing such patterns is challenging for the following reasons:

1. The number of possible patterns increases exponentially with the length of the patterns. For example, the string:

“Amy sings ‘Rehab’”

can give rise to the patterns:

$\langle \text{singer} \rangle \text{ sings } \langle \text{song} \rangle$,
 $\langle \text{person} \rangle \text{ sings } \langle \text{artifact} \rangle$,
 $\langle \text{person} \rangle \text{ [vbz] } \langle \text{entity} \rangle$, etc.

If wildcards for multiple words are allowed, such as in:

$\langle \text{person} \rangle \text{ sings } * \langle \text{song} \rangle$,

the number of possible patterns explodes.

2. A pattern can be semantically more general than another pattern (when one relation is implied by the other relation), and it can also be syntactically more general than another pattern (by the use of placeholders such as [vbz]). These two subsumption orders have a non-obvious interplay, and none can be analyzed without the other.
3. We have to handle pattern sparseness and coincidental matches. If the corpus is small, the patterns:

$\langle \text{singer} \rangle \text{ later disliked her song } \langle \text{song} \rangle$
and
 $\langle \text{singer} \rangle \text{ sang } \langle \text{song} \rangle$,

may, for example, apply to the same set of entity pairs in the corpus. Still, the patterns are not synonymous.

4. Computing mutual subsumptions on a large set of patterns may be prohibitively slow. Moreover, due to noise and vague semantics, patterns may even not form a crisp taxonomy, but require a hierarchy in which subsumption relations have to be weighted by statistical confidence measures.

Toward meeting PATTY's goal, the chapter makes the following contributions:

1. *SOL patterns*: We define an expressive family of relational patterns, which combines syntactic features (S), ontological type signatures (O), and lexical features (L). The crucial novelty is the addition of the ontological (semantic) dimension to patterns. When compared to a state-of-the-art pattern language, we found that SOL patterns yield higher recall while achieving similar precision.
2. *Mining algorithms*: We present efficient and scalable algorithms that can infer SOL patterns and subsumptions at scale, based on instance-level overlaps and an ontological type hierarchy.
3. A large *Lexical resource*:. On the Wikipedia corpus, we obtained 350,569 pattern synsets with 84.7% precision. We make our pattern taxonomy available for further research at www.mpi-inf.mpg.de/yago-naga/patty/.

4.3 RELATED WORK

Binary Relations in Knowledge Bases

A wealth of taxonomic knowledge bases (KBs) about entities and their semantic classes have become available. These are very rich in terms of unary predicates (semantic classes) and their entity instances. However, the number of *binary relations* (i.e., relation types, not instances) in these KBs is usually small: Freebase [13] has a few thousand hand-crafted relations. WikiNet [86] has automatically extracted ca. 500 relations from Wikipedia category names. DBpedia [8] has automatically compiled ca. 8000 names of properties from Wikipedia infoboxes, but these include many involuntary semantic duplicates such as *surname* and *lastname*. In all of these projects, the resource contains the *relation names*, but not the *natural language patterns* for them. The same is true for other projects along these lines [85, 93, 94, 108].

Surface Patterns in Knowledge bases

Knowledge base projects that automatically populate relations from Web pages also learn *surface patterns* for the relations: examples are TextRunner/ReVerb [10, 41], NELL [20, 77], Probase [123], the dynamic lexicon approach by [59, 122], the LDA-style clustering approach by [131], which learns relation-specific lexica to train relation-specific CRF extractors and projects on Web tables [74,

112]. Of these, only TextRunner/ReVerb and NELL have made large pattern collections publicly available.

ReVerb [41] constrains patterns to verbs or verb phrases that end with prepositions, while PATTY can learn arbitrary patterns. More importantly, all methods in the TextRunner/ReVerb family are blind to the ontological dimension of the entities in the patterns. Therefore, there is no notion of semantic typing for relation phrases as in PATTY.

NELL and OntExt

NELL [20] is based on a fixed set of pre-specified relations with type signatures, (e.g., *personHasCitizenship*: $\langle person \rangle \times \langle country \rangle$), and learns to extract suitable noun-phrase pairs from a large Web corpus. In contrast, PATTY discovers patterns for relations that are a priori unknown.

In *OntExt* [77], the NELL architecture was extended to automatically compute new relation types (beyond the pre-specified ones) for a given type signature of arguments, based on a clustering technique. For example, the relation *musicianPlaysInstrument* is found by clustering pattern co-occurrences for the noun-phrase pairs that fall into the specific type signature $\langle musician \rangle \times \langle musicinstrument \rangle$. This technique works for one type signature at a time, and does not scale up to mining a large corpus. Also, the technique is not suitable for inferring semantic subsumptions. In contrast, PATTY efficiently acquires patterns from large-scale corpora and organizes them into a subsumption hierarchy.

Class-based Attribute Discovery

Class-based *attribute discovery* is a special case of mining relational patterns (e.g., [4, 91, 92, 97]). Given a semantic class, such as *movies* or *musicians*, the task is to determine relevant attributes, such as *cast* and *budget* for movies, or *albums* and *biography* for musicians, along with their instances. Unlike PATTY's patterns, the attributes are not typed. They come with a pre-specified type for the domain, but without any type for the range of the underlying relation.

Relation-centric NLP tasks

There are further *relation-centric tasks in NLP and text mining* that have commonalities with our endeavor, but differ in fundamental ways. The SemEval-2010 task on classification of semantic relations between noun-phrase pairs [55] aimed at predicting the relation for a given sentence and pair of nominals, but used a fixed set of pre-specified relations. Another task in this research avenue is to characterize and predict the argument types for a given relation or pattern [65, 84]. This is closer to KB population and less related to our task of discovering relational patterns and systematically organizing them.

Unary Predicates and Lexical Resources

From a *linguistic perspective*, there is ample work on patterns for unary predicates of the form *class(entity)*. This includes work on entailment of classes, i.e., on *is-a* and *subclassOf* relationships. Entailment among binary predicates of the form *relation(entity1, entity2)* has received less attention [71, 52, 11]. These works focus solely on verbs, while PATTY learns arbitrary phrases for patterns.

Several lexical resources capture verb categories and entailment: WordNet 3.0 [43] contains about 13,000 verb senses, with troponymy and entailment relations; VerbNet [61] is a hierarchical lexicon with more than 5,000 verb senses in ca. 300 classes, including selectional preferences. Again, all of these resources focus solely on verbs.

ConceptNet 5.0 [53] is a thesaurus of commonsense knowledge built as a crowd-sourcing endeavor. PATTY, in contrast, is constructed fully automatically from large corpora.

Paraphrasing

Automatic learning of paraphrases and textual entailment has received much attention (see the survey of [6]), but does not consider fine-grained typing for binary relations, as PATTY does.

4.4 OVERVIEW OF PATTY

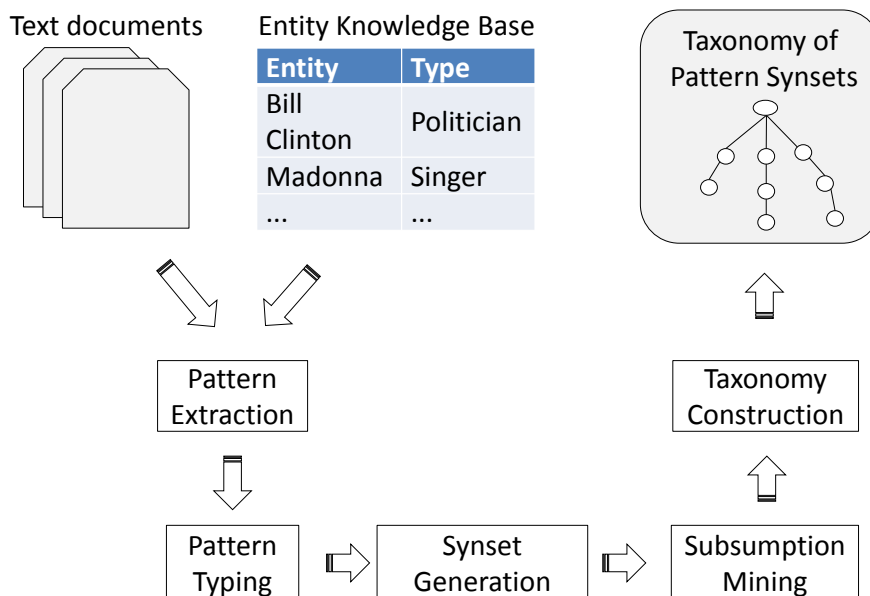


Figure 4.1: Architecture of the PATTY System.

PATTY processes large text corpora (e.g., the full text of Wikipedia, news archives, or Web crawls) to build a taxonomy of textual patterns. Figure 4.1 shows PATTY's architecture. PATTY is backed by a knowledge base of semantically typed entities. For the latter, we use either YAGO [108] or Freebase [13]: YAGO has classes derived from Wikipedia categories and integrated with WordNet classes to form a hierarchy of types (Figure 4.2 shows a simplified hierarchy of types); Freebase has a handcrafted type system with upper level topical domains as top tier and about entity classes as a second tier. PATTY works in four stages:

1. **Pattern extraction.** A pattern is a surface string that occurs between a pair of entities in a sentence, thus the first step is to obtain basic textual patterns from the input corpus (Section 4.5).
2. **SOL Pattern Transformation.** The second step is to transform plain patterns into SOL patterns thereby enhancing them with ontological types (Section 4.6).
3. **Pattern Generalization.** The third step is to generalize the patterns, both syntactically and semantically (Sections 4.7 and 4.8).
4. **Subsumption mining.** The last step is to arrange the patterns into a hierarchy based on hypernymy/hyponymy relations between patterns (Section 4.9).

Each of the stages is explained in detail in the next few sections.

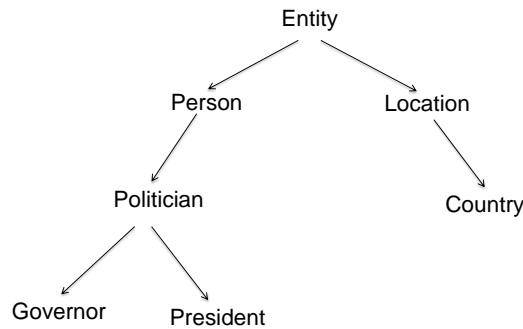


Figure 4.2: An example type system.

4.5 PATTERN EXTRACTION

This section explains how we obtain basic textual patterns from the input corpus. We first apply the Stanford Parser [75] to the individual sentences of the corpus to obtain dependency paths. The dependency paths form a directed graph, with words being nodes and dependencies being edges. For example, the sentence:

"Winehouse effortlessly performed her song Rehab."

yields the following dependency paths:

nsubj(performed-3, Winehouse-1)
advmod(performed-3, effortlessly-2)
poss(Rehab-6, her-4)
nn(Rehab-6, song-5)
dobj(performed-3, Rehab-6)

While our method also works with patterns obtained from shallow features such as POS tags, we found that dependency paths improve pattern extraction precision especially on long sentences.

We then detect mentions of named entities in the parsed corpus. For this purpose, we use a dictionary of entities. This can be any resource that contains named entities with their surface names and semantic types [8, 108, 58, 13]. In our experiments, we used the YAGO2 knowledge base [58]. We match noun phrases that contain at least one proper noun against the dictionary. For disambiguation, we use a simple context-similarity prior, as described in [109]. We empirically found that this technique has accuracy well above 80% (and higher for prominent and thus frequently occurring entities). In our example, the entity detection yields the entities:

Amy Winehouse
 and
Rehab (song).

Whenever two named entities appear in the same sentence, we extract a textual pattern. For this purpose, we traverse the dependency graph to get the shortest path that connects the two entities. In the example, the shortest path between "Winehouse" and "Rehab" is:

Winehouse *nsubj* performed *dobj* Rehab.

In order to capture only relations that refer to subject-relation-object triples, we only consider shortest paths that start with subject-like dependencies, such as *nsubj*, *rcmod* and *partmod*. To reflect the full meaning of the patterns, we expand the shortest path with adverbial and adjectival modifiers, for example the *advmod* dependency. The sequence of words on the expanded shortest path becomes our final textual pattern. In the example, the textual pattern is:

Amy Winehouse effortlessly performed Rehab (song).

4.6 SOL PATTERN MODEL

Textual patterns are tied to the particular surface form of the text. Therefore, we transform the textual patterns into a new type of patterns, called syntactic-ontological-lexical patterns (SOL patterns). SOL patterns extend lexico-syntactic patterns by ontological type signatures for entities. The SOL pattern language is expressive enough to capture fine-grained relational patterns, yet simple enough to be dealt with by efficient mining algorithms at Web scale.

Definition 4.6.1 (SOL Pattern) A SOL pattern is an abstraction of a textual pattern that connects two entities of interest. It is a sequence of words, POS-tags, wildcards, and ontological types.

A POS-tag stands for a word of the part-of-speech class such as a *noun*, *verb*, *possessive pronoun*, etc. We introduce the special POS-tag [word], which stands for any word of any POS class. A wildcard, denoted *, stands for any (possibly empty) sequence of words. Wildcards are essential to avoid overfitting of patterns to the corpus. An ontological type is a semantic class name (such as $\langle \text{singer} \rangle$) that stands for an instance of that class. Every pattern contains at least two types, and these are designated as *entity placeholders*.

Definition 4.6.2 (Entity Placeholder) An entity placeholder in a SOL pattern is a semantic class name that represents the types of entities that are allowed to appear in that position.

In the following we consider only SOL patterns with exactly two entity placeholders.

Definition 4.6.3 (String Pattern Match) A string and a pattern match, if there is an order-preserving bijection from sequences of words in the string to items in the pattern, so that each item can stand for the respective sequence of words. For example, the pattern:

$\langle \text{person} \rangle$'s [adj] voice * $\langle \text{song} \rangle$

matches the strings:

"Amy Winehouse's soft voice in 'Rehab'"

and

"Elvis Presley's solid voice in his song 'All shook up'".

Definition 4.6.4 (Type Signature) The type signature of a pattern is the pair of the entity placeholders. In the example, the type signature is $\text{person} \times \text{song}$.

Definition 4.6.5 (Pattern Support Set) The support set of a pattern is the set of pairs of entities that appear in the place of the entity placeholders in all strings in the corpus that match the pattern. In the example, the support set of the pattern could be:

$\{(Amy, Rehab), (Elvis, AllShookUp)\}$.

Each pair is called a support pair of the pattern.

Definition 4.6.6 (Lexico-Syntactic Generalization) Pattern B is syntactically more general than pattern A if every string that matches A also matches B.

Definition 4.6.7 (Semantic Generalization) Pattern B is semantically more general than A if the support set of B is a superset of the support set of A.

Definition 4.6.8 (Pattern Synset) If A is semantically more general than B and B is semantically more general than A, the patterns are called synonymous. A set of synonymous patterns is called a pattern synset. Two patterns, of which neither is semantically more general than the other, are called semantically different.

To generate SOL patterns from the textual patterns, we decompose the textual patterns into n-grams (n consecutive words). A SOL pattern contains only the n-grams that appear frequently in the corpus and the remaining word sequences are replaced by wildcards. For example, the sentence:

“was the first female to run for the governor of”

might give rise to the pattern:

* *the first female* * *governor of*,

if “the first female” and “governor of” are frequent in the corpus.

To find the frequent n-grams efficiently, we apply the technique of frequent item-set mining [1, 95]: each sentence is viewed as a “shopping transaction” with a “purchase” of several n-grams, and the mining algorithm computes the n-gram combinations with large co-occurrence support¹. These n-grams allow us to break down a sentence into wildcard-separated subsequences, which yields an SOL pattern. We generate multiple patterns with different types, one for each combination of types that the detected entities have in the underlying ontology.

We quantify the *statistical strength* of a pattern by means of its support set. We compute support and confidence as follows:

Definition 4.6.9 (Pattern Support) For a given pattern p with type signature $t1 \times t2$, the support of p is the size of its support set.

Definition 4.6.10 (Pattern Confidence) For a given pattern p with type signature $t1 \times t2$, its confidence is computed by comparing the support-set sizes of p and an untyped variant p^u of p , in which the types $\langle t1 \rangle$ and $\langle t2 \rangle$ are replaced by the generic type $\langle \text{entity} \rangle$. We define the confidence of p as the ratio of the support-set sizes of p and p^u .

4.7 LEXICO-SYNTACTIC PATTERN GENERALIZATION

Almost every pattern can be generalized into a syntactically more general pattern in several ways: by replacing words by POS-tags, by introducing wildcards (combining more n-grams), or by generalizing the types in the pattern. It is not obvious which generalizations will be reasonable and useful. We observe, however, that generalizing a pattern may create a pattern that subsumes two semantically different patterns. For example, the generalization:

$\langle \text{person} \rangle$ [vb] $\langle \text{person} \rangle$,

subsumes the two semantically different patterns :

$\langle \text{person} \rangle$ loves $\langle \text{person} \rangle$

and

$\langle \text{person} \rangle$ hates $\langle \text{person} \rangle$.

¹ Our implementation restricts n-grams to length 3 and uses up to 4 n-grams per sentence

This means that the pattern is semantically meaningless.

Therefore, we proceed as follows. For every pattern, we generate all possible generalizations. If a generalization subsumes multiple patterns with disjoint support sets, we abandon the generalized pattern. Otherwise, we add it to our set of patterns.

4.8 SEMANTIC PATTERN GENERALIZATION

The main difficulty in generating semantic subsumptions is that the support sets may contain spurious pairs or be incomplete, thus destroying crisp set inclusions. To overcome this problem, we designed a notion of a *soft set inclusion*, in which one set S can be a subset of another set B to a certain degree. One possible measure for this degree is the confidence, i.e., the ratio of elements in S that are in B ,

$$\text{deg}(S \subseteq B) = |S \cap B|/|S|.$$

However, if a support set S has only few elements due to sparsity, it may become a subset of another support set B , even if the two patterns are semantically different. Therefore, one has to take into account also the *support*, i.e., the size of the set S . Traditionally, this is done through a weighted trade-off between confidence and support.

To avoid the weight tuning, we instead devised a probabilistic model. We interpret S as a *random sample* from the “true” support set S' that the pattern would have on an infinitely large corpus. We want to estimate the ratio of elements of S' that are in B . This ratio is a Bernoulli parameter that can be estimated from the ratio of elements of the sample S that are in B . We compute the Wilson score interval $[c - d, c + d]$ [15] for the sample. This interval guarantees that with a given probability (set a priori, usually to $\alpha = 95\%$), the true ratio falls into the interval $[c - d, c + d]$. If the sample is small, d is large and c is close to 0.5. If the sample is large, d decreases and c approaches the naive estimation $|S \cap B|/|S|$. Thereby, the Wilson interval center naturally balances the trade-off between confidence and the support. Hence we define:

$$\text{deg}(S \subset B) = c.$$

This estimator may degrade when the sample size is too small. We can alternatively use a conservative estimator :

$$\text{deg}(S \subset B) = c - d,$$

i.e., the lower bound of the Wilson score interval. This gives a low score to the case where $S \subset B$ if we have few samples (S is small).

4.9 TAXONOMY CONSTRUCTION

We now have to arrange the patterns in a semantic taxonomy. A baseline solution would compare every pattern support set to every other pattern support set in

ID	Pattern Synset & Support Sets
P_1	$\langle \text{Politician} \rangle$ was governor of $\langle \text{State} \rangle$ A,80 B,75 C,70
P_2	$\langle \text{Politician} \rangle$ politician from $\langle \text{State} \rangle$ A,80 B,75 C,70 D,66 E,64
P_3	$\langle \text{Person} \rangle$ daughter of $\langle \text{Person} \rangle$ F,78 G,75 H,66
P_4	$\langle \text{Person} \rangle$ child of $\langle \text{Person} \rangle$ I,88 J,87 F,78 G,75 K,64

Table 4.1: Pattern Synsets and their Support Sets

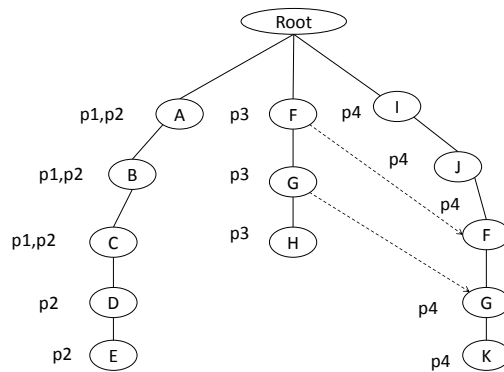


Figure 4.3: Prefix-tree for the Synsets in Table 4.1.

order to determine inclusion, mutual inclusion, or independence. This would be prohibitively slow. For this reason, we make use of a prefix-tree for frequent patterns [50]. The prefix-tree stores support sets of patterns. We then developed an algorithm for obtaining set intersections from the prefix-tree.

Prefix-Tree Construction

Suppose we have pattern synsets and their support sets as shown in Table 4.1. An entity pair in a support set is denoted by a letter. For example, in the support set for the pattern:

$\langle \text{Politician} \rangle$ was governor of $\langle \text{State} \rangle$,

the entry $\langle A,80 \rangle$ may denote the entity pair:

Arnold Schwarzenegger, California,

with an occurrence frequency 80. The contents of the support sets are used to construct a prefix-tree.

Definition 4.9.1 (Prefix-tree of Support Sets) Consider a set T of SOL patterns represented by their support sets. A prefix tree of support sets T is a tree consisting of a node set V representing entity pairs in the support sets of patterns. If pattern

synsets have entity pairs in common, they share a common prefix; thus the shared parts can be represented by one prefix-path in the tree.

Nodes are augmented with synset information stored at the nodes. Each node (entity pair) stores the identifiers of the patterns whose support sets contain that entity pair. In addition, each node stores a link to the next node with the same entity pair.

This is a compact representation of the support sets. To increase the chance of shared prefixes, entity pairs are inserted into the tree in decreasing order of occurrence frequency.

Figure 4.3 shows the tree for the pattern synsets in Table 4.1. The left-most path contains synsets P_1 and P_2 . The two patterns have a prefix in common, thus they share the same path. This is reflected by the synsets stored in the nodes in the path. Synsets P_2 and P_3 belong to two different paths due to dissimilar prefixes although they have common nodes. Instead, their common nodes are connected by the same-entity-pair links shown as dotted lines in Figure 4.3.

Definition 4.9.2 (Same-entity-pair Link) *A same-entity-pair link is a link between two nodes with the same entity pairs. It is created whenever the entity pair already exists in the tree but with a prefix different from the prefix of the synset being added to the tree.*

The size of the tree is at most the total number of entity pairs making up the supports sets of the synsets. The height of the tree is at most the size of the largest support set.

Mining Subsumptions from the Prefix-Tree

To efficiently mine subsumptions from the prefix-tree, we have to avoid comparing every path to every other path as this introduces the same inefficiencies that the baseline approach suffers from.

From the construction of the tree it follows that for any node N_i in the tree, all paths containing N_i can be found by following node N_i 's links including the same-entity-pair links (see dotted lines in Figure 4.3). By traversing the entire path of a synset P_i , we can reach all the pattern synsets sharing common nodes with P_i . This leads to our main insight: if we start traversing the tree bottom up, starting at the last node in P_i 's support set, we can determine exactly which paths are subsumed by P_i . Traversing the tree this way for all patterns gives us the sizes of the support set intersection. The determined intersection sizes can then be used in the Wilson estimator to determine the degree of semantic subsumption and semantic equivalence of patterns.

For example, in Figure 4.3, suppose we start at node E . From the synset list, we identify that the only synset terminating in node E is P_2 , thus we start searching for all synsets that are subsumed by P_2 . At node D , we do not encounter any new synsets, at node C we pick up synset P_1 , we thus mine the possible subsumption that $P_1 \Rightarrow P_2$, with the size of their intersection, $P_2 \cap P_1$ initialized to 1. We continue traversing the path of P_2 until it terminates at node A , incrementing

the counter for $P_2 \cap P_1$, and picking up any new possible subsumptions along the way for all the new synsets encountered on the path, in this case there are none. At the end of P_2 's path, we end up with a possible subsumption is $P_1 \Rightarrow P_2$, with $P_2 \cap P_1 = 3$. Having processed node E , the next node to process is node D , but we see that no synsets terminates in D , thus we proceed to the next node C which is terminating node for synset P_1 , traversing P_1 's path we do not encounter any other synsets thus P_1 does not subsume any other synsets.

For the case when same-entity-pair links are present, we examine the paths of P_3 and P_4 . Starting bottom-up, we begin with node K , which we start to process as it is a synset terminating node, P_4 stops there. Traversing P_4 's path, we first encounter node G , which has same-entity-pair links. We thus find G 's head node, which is the the first G node added to the tree, from that node we follow the same-entity-pair links, picking up any synset we encounter as being possibly subsumed by P_4 . In this case, G was first added in the tree for path P_3 , we thus introduce a potential subsumption $P_3 \Rightarrow P_4$ and initialize $P_3 \cap P_4 = 1$, the next link to G is in the path P_4 , we thus continue to the next node, node F , from the same-entity-pair links of F we update $P_3 \cap P_4 = 2$.

The next synset terminating node in Figure 4.3 is H when synset P_3 ends, traversing its path, we encounter nodes G and F , resulting in a possible $P_4 \Rightarrow P_3$ with $P_4 \cap P_3 = 2$. It is possible that both $P_i \Rightarrow P_j$ and $P_j \Rightarrow P_i$ hold simultaneously. This is an indication of a synonyms, that $P_i = P_j$.

Notice that in this example, for clarity of exposition, we first processed the left-most sub-tree, but based on the bottom-up processing order the algorithm in fact starts with the right-most branch.

The algorithm is given in *Algorithm 4.1*. Given a prefix tree T , and the minimum subsumption threshold α , we process every synset terminating node in the tree. For every such node, we traverse the path from that node to the root, and keep track of all possible subsumptions and their corresponding set intersection sizes. Finally, we add all the subsumptions of the node to the set of all subsumptions, granted that they meet the minimum subsumption threshold. Since the number of synset terminating nodes is proportional to the number of synsets in the data, the complexity of the algorithm is $O(|Synsets|)$, thus it is linear in the number of synsets.

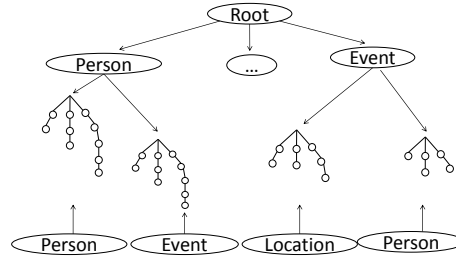


Figure 4.4: An example type-segmented tree.

Algorithm 4.1 Prefix-tree Subsumption MiningInput: Support set prefix-tree T and a subsumption threshold α Output: Complete set of subsumption relations, S

```

1: procedure MineSubmptions( $T, \alpha$ )
2:    $S \leftarrow \emptyset$ ; // initialize subsumptions set
3:   for node  $n_i \in T$  do // process tree bottom up
4:     if not  $n_i$  is synset terminating
5:       continue;
6:      $C_i \leftarrow$  synsets terminating in  $n_i$ ;
7:      $S_i \leftarrow \emptyset$ ; // subsumptions of current node
8:     for node  $n_j \in$  path  $n_i \rightarrow$  root do
9:        $CX_j \leftarrow$  synsets in  $n_j$ ;
10:      for synset  $c_i \in C_i$  do
11:        for synset  $cx_j \in CX_j$  do
12:          if  $cx_j \Rightarrow c_i \notin S_i$ 
13:             $S_i.add(cx_j \Rightarrow c_i, 0)$ ; // initialize  $cx_j \cap c_i$ 
14:             $S_i.increment(cx_j \cap c_i)$ ;
15:          endfor
16:        endfor
17:       $S \cup \{ cx \Rightarrow c \in S_i : (|cx - c| \leq \alpha) \}$ 
18:    endfor
19:  endfor
20:  return  $S$ 

```

Sub-tree segmentation. Algorithm 4.1 requires that the tree be stored in memory, which may not always fit. For this reason, we propose an optimization step which reduces the memory requirements for storing the tree. Recall that a synset can only subsume another if their type signatures are compatible. Therefore, at any given time, we only need to keep in memory those parts of tree that are compatible. In this case, we can alter the tree construction algorithm so that the root is constrained by types, so that nodes belonging to a pair of compatible types go to one sub-tree. An example type-segmented tree is shown in Figure 4.4.

Alternative MapReduce Algorithm. If sub-trees still do not fit in memory, we

Entity-Pair	Synsets
A	P_1, P_2
B	P_1, P_2
C	P_1, P_2
D	P_2
E	P_2
F	P_3P_4
G	P_3P_4
H	P_3
I	P_4
J	P_4
K	P_4

Table 4.2: Inverted data for Map-Reduce algorithm

developed a highly scalable algorithm using the Map-Reduce framework. First, we invert the synset support data. Instead of providing for a synset, all entity-pairs that occur with it, we provide for an entity-pair, all the synsets that it occurs with. Table 4.2 shows the inverted support sets of the data in Figure 4.1. We then determine the co-occurrence of synsets by emitting pairs of synsets that co-occur for every synset they co-occur together. This is done in the mappers. For example for A, B and C, the mappers emit p_1, p_2 , whereas for E, F, I, J, K nothing is emitted and so on. The reducers aggregate co-occurrence information, to effectively output the sizes of the set intersection of the possible subsumptions. Once these set intersections are known, another Map-Reduce job is launched to determine which of these surpass the subsumption thresholds.

Removing Cycles

Once we have generated subsumptions between relational patterns, there might be cycles in the graph we generate. We ideally want to remove the minimal total number of subsumptions whose removal results in an a directed acyclic graph (DAG). This task is related to the minimum feedback-arc-set problem: given a directed graph, we want to remove the smallest set of edges whose removal makes the remaining graph acyclic. This is a well known NP-hard problem [60]. We use a greedy algorithm for removing cycles and eliminating redundancy in the subsumptions, thus effectively constructing a DAG. Starting with a list of subsumption edges ordered by decreasing weights, we construct the DAG bottom-up by adding the highest-weight subsumption edge. This step is repeated for all subsumptions, where we add a subsumption to the DAG only if it does not introduce cycles or redundancy. Redundancy occurs when there already exists a path, by transitivity of subsumptions, between pattern synsets linked by the subsumption. This process finally yields a DAG of pattern synsets – the PATTY taxonomy.

4.10 EVALUATION

Setup

The PATTY extraction and mining algorithms were run on two different input corpora:

- The New York Times archive (*NYT*) which includes about 1.8 Million newspaper articles from the years 1987 to 2007
- The English edition of Wikipedia (*WKP*), which contains about 3.8 Million articles (as of June 21, 2011).

Experiments were carried out, for each corpus, with two different type systems:

- The type system of YAGO2, which consists of about 350,000 semantic classes from WordNet and the Wikipedia category system
- The two-level domain/type hierarchy of Freebase which consists of 85 domains and a total of about 2000 types within these domains.

All relational patterns and their respective entity pairs are stored in a MongoDB database. We evaluated PATTY along four dimensions: quality of patterns, quality of subsumptions, coverage, and design alternatives. These dimensions are discussed in the following four subsections. We also performed an extrinsic study to demonstrate the usefulness of PATTY for paraphrasing the relations of DBpedia and YAGO2. In terms of runtimes, the most expensive part is the pattern extraction, where we identify pattern candidates through dependency parsing and perform entity recognition on the entire corpus. This phase runs about a day for Wikipedia on a cluster. All other phases of the PATTY system take less than an hour. All experimental data is available on our Web site at www.mpi-inf.mpg.de/yago-naga/patty/.

Precision of Relational Patterns

To assess the precision of the automatically mined patterns (patterns in this section always mean pattern synsets), we sampled the PATTY taxonomy for each combination of input corpus and type system. We ranked the patterns by their statistical strength (Section 4), and evaluated the precision of the *top 100* pattern synsets. Several human judges were shown a sampled pattern synset, its type signature, and a few example instances, and then stated whether the pattern synset indicates a valid relation or not. Evaluators checked the correctness of the type signature, whether the majority of patterns in the synset is reasonable, and whether the instances seem plausible. If so, the synset was flagged as meaningful. The results of this evaluation are shown in column four of Table 4.3, with a 0.9-confidence Wilson score interval [15]. In addition, the same assessment procedure was applied to *randomly sampled* synsets, to evaluate the quality in the long tail of patterns. The results are shown in column five of Table 4.3. For the top 100 patterns, we achieve above 90% precision for Wikipedia, and above

80% for 100 random samples. Sample relational patterns are shown in *Appendix A*.

Corpus	Types	Patterns	Top 100	Random
NYT	YAGO2	86,982	0.89±0.06	0.72±0.09
	Freebase	809,091	0.87 ±0.06	0.71±0.09
WKP	YAGO2	350,569	0.95±0.04	0.85±0.07
	Freebase	1,631,531	0.93±0.05	0.80±0.08

Table 4.3: Precision of Relational Patterns

From the results we make two observations. First, Wikipedia patterns have higher precision than those from the New York Times corpus. This is because some the language in the news corpus does not express relational information; especially the news on stock markets produced noisy patterns picked up by PATTY. However, we still manage to have a precision of close to 90% for the top 100 patterns and around 72% for random sample on the NYT corpus. The second observation is that the YAGO2 type system generally led to higher precision than the Freebase type system. This is because YAGO2 has finer grained, ontologically clean types, whereas Freebase has broader categories with a more liberal assignment of entities to categories.

Precision of Subsumptions

We evaluated the quality of the subsumptions by assessing 100 top-ranked as well as 100 randomly selected subsumptions. As shown in Table 4.4, a large number of the subsumptions are correct. The Wikipedia-based PATTY taxonomy has a random-sampling-based precision of 75%.

Corpus	Types	# Edges	Top 100	Random
NYT	YAGO2	12,601	0.86±0.07	0.68±0.09
	Freebase	80,296	0.89±0.06	0.41±0.09
WKP	YAGO2	8,162	0.83±0.07	0.75±0.07
	Freebase	20,339	0.85±0.07	0.62±0.09

Table 4.4: Quality of Subsumptions

Example subsumptions from Wikipedia are:

- $\langle person \rangle$ nominated for $\langle award \rangle$ \square
- $\langle person \rangle$ winner of $\langle award \rangle$
- $\langle person \rangle$'s wife $\langle person \rangle$ \square
- $\langle person \rangle$'s widow $\langle person \rangle$

Coverage

To evaluate the coverage of PATTY, we would need a complete ground-truth resource that contains all possible binary relations between entities. Unfortunately, there is no such resource². We tried to approximate such a resource by manually compiling all binary relations between entities that appear in Wikipedia articles of a certain domain. We chose the domain of popular music, because it offers a plethora of non-trivial relations (such as *addictedTo(person,drug)*, *coveredBy(musician,musician)*, *dedicatedSongTo(musician,entity)*). We considered the Wikipedia articles of five musicians (Amy Winehouse, Bob Dylan, Neil Young, John Coltrane, Nina Simone). For each page, two annotators hand-extracted all relationship types that they would spot in the respective articles. The annotators limited themselves to relations where at least one argument type is *musician*. Then we formed the intersection of the two annotators' outputs (i.e., their agreement) as a reasonable gold standard for relations identifiable by skilled humans. In total, the gold-standard set contains 163 relations.

We then compared our relational patterns to the relations included in four major knowledge bases, namely, YAGO2, DBpedia (DBP), Freebase (FB), and NELL, limited to the specific domain of music. Table 4.5 shows the absolute number of relations covered by each resource. For PATTY, the patterns were derived from the Wikipedia corpus with the YAGO2 type system.

gold standard	PATTY	YAGO2	DBP	FB	NELL
163	126	31	39	69	13

Table 4.5: Coverage of Music Relations

PATTY covered 126 of the 163 gold-standard relations. This is more than what can be found in large semi-curated knowledge bases such as Freebase, and twice as much as Wikipedia-infobox-based resources such as DBpedia or YAGO offer. Some PATTY examples that do not appear in the other resources at all are:

- *musician* PRP *idol* *musician* for the relation *hasMusicalIdol*
- *person* criticized by *organization* for *criticizedByMedia*
- *person* headliner *artifact* for *headlinerAt*
- *person* successfully sued *person* for *suedBy*
- *musician* wrote hits for *musician* for *wroteHitsFor*,
- *performer* be dedicate to *person* for *dedicatedSongTo*,
- *person* in [det] week's *press* for *appearedInMedia*.

This shows (albeit anecdotically) that PATTY's patterns contribute added value beyond today's knowledge bases. The full comparison results of PATTY vs. other resources is shown in *Appendix A*.

²Lexical resources such as WordNet contain only verbs, but not binary relations such as *is the president of*. Other resources are likely incomplete.

	Reverb-style patterns	PATTY without types	PATTY full
# Patterns	5,996	184,629	350,569
Patterns Precision	0.96±0.03	0.74±0.08	0.95±0.04
# Subsumptions	74	15,347	8,162
Subsumptions Precision	0.79 ±0.09	0.58±0.09	0.83±0.07
# Facts	192,144	6,384,684	3,890,075
Facts Precision.	0.86 ±0.07	0.64±0.09	0.88 ±0.06

Table 4.6: Results for Different Pattern Language Alternatives

Relation	Paraphrases	Precision
DBPedia/artist	83	0.96±0.03
DBPedia/associatedBand	386	0.74±0.11
DBPedia/doctoralAdvisor	36	0.558±0.15
DBPedia/recordLabel	113	0.86±0.09
DBPedia/riverMouth	31	0.83±0.12
DBPedia/team	1,108	0.91±0.07
YAGO/actedIn	330	0.88±0.08
YAGO/created	466	0.79±0.10
YAGO/isLeaderOf	40	0.53±0.14
YAGO/holdsPoliticalPosition	72	0.73±0.10

Table 4.7: Sample Results for Relation Paraphrasing Precision

Pattern Language Alternatives

We also investigated various design alternatives to the PATTY pattern language. We looked at three main alternatives: the first is verb-phrase-centric patterns advocated by ReVerb [41], the second is the PATTY language without type signatures (just using sets of n-grams with syntactic generalizations), and the third one is the full PATTY language. The results for the Wikipedia corpus and the YAGO2 type system are shown in Table 4.6; precision figures are based on the respective top 100 patterns or subsumption edges. We observe from these results that the type signatures are crucial for precision. Moreover, the number of patterns, subsumptions and facts found by verb-phrase-centric patterns (ReVerb [41]), are limited in recall. General pattern synsets with type signatures, as newly pursued in PATTY, substantially outperform the verb-phrase-centric alternative in terms of pattern and subsumption recall while yielding high precision.

Extrinsic Study: Relation Paraphrasing

To further evaluate the usefulness of PATTY, we performed a study on relation paraphrasing: given a relation from a knowledge base, identify patterns that

Relation	Sample Paraphrases
DBPedia/artist	[adj] studio album of, [det] song by ...
DBPedia/associatedBand	joined band along, plays in ...
DBPedia/doctoralAdvisor	[det] student of, under * supervision ...
DBPedia/recordLabel	[adj] artist signed to, [adj] record label ...
DBPedia/riverMouth	drains into, [adj] tributary of ...
DBPedia/team	be * traded to, [prp] debut for ...
YAGO/actedIn	starred in * film, [adj] role for ...
YAGO/created	founded, 's book ...
YAGO/isLeaderOf	elected by, governor of ...
YAGO/holdsPoliticalPosition	[prp] tenure as, oath as ...

Table 4.8: Sample Results for Relation Paraphrasing

can be used to express that relation. Paraphrasing relations with high-quality patterns is important for populating knowledge bases and counters the problem of semantic drifting caused by ambiguous and noisy patterns.

We considered relations from two knowledge bases, DBpedia and YAGO2, focusing on relations that hold between entities and do not include literals. PATTY paraphrased 225 DBpedia relations with a total of 127,811 patterns, and 25 YAGO2 relations with a total of 43,124 patterns. Among these we evaluated a random sample of 1,000 relation paraphrases. Table 4.7 shows precision figures for some selected relations, along anecdotic example patterns.

Some relations are hard to capture precisely. For *DBPedia/doctoralAdvisor*, e.g., PATTY picked up patterns like “worked with” as paraphrases. These are not entirely wrong, but we evaluated them as false because they are too general to indicate the more specific doctoral advisor relation.

Overall, however, the paraphrasing precision is high. Our evaluation showed an average precision of 0.76 ± 0.03 across all relations.

4.11 APPLICATIONS

PATTY presents a valuable resource for a variety of applications: First, it can boost IE and knowledge-base population tasks by its rich and clean repository of paraphrases for the relations. Second, it can improve Open IE by associating type signatures with patterns. Third, it can help to discover “Web witnesses” when assessing the truthfulness of search results or statements in social media [42]. Last, it provides paraphrases for detecting relationships in keyword queries, thus lifting keyword search to the entity-relationship level. This can help to understand questions and text snippets in natural-language QA.

We developed a front-end to the PATTY data for exploring these possibilities in three ways:

1. Using PATTY as a thesaurus to find paraphrases for relations

- Using PATTY as a simple kind of QA system to query the database without having to know the schema
- Exploring the relationships between entities, as expressed in the textual sources.

The Web-based front-end is running AJAX for asynchronous communication with the server.

The screenshot shows the PATTY web interface. On the left, a sidebar lists various relations under 'YAGO Relations', with 'actedIn' selected. The main content area is titled 'Relation: yago:actedIn' and displays a table of patterns, domains, ranges, confidences, and support counts. A modal window is open, showing a 'Synset' for the pattern 's character from;' with several alternative patterns and example sentences.

Pattern	Domain	Range	Confidence	Support
also starred in;	actor	event	0.695	425
starred in [[det]] film;	person	artifact	0.937	97
appeared in [[adj]] film;	person	movie	0.865	80
twice won [[det]];				
also costarred as;				
also acted [[con]];				
[[det]] star in;				
[[prp]] movie;				
[[prp]] film debut with;				
s character from;				
played [[det]] [[adj]] film;				
portrayed in;				
appeared;				
received only;				

The modal window shows a 'Synset' for the pattern 's character from;' with the following patterns:

- s character from;
- [[adj]] character [[det]];
- character in;
- [[adj]] character to;
- [[det]] character as;

Example sentences shown in the modal window:

- Arnold Schwarzenegger , Terminator (franchise)
- Jodie Foster , The Accused (1988 film)
- Gold Digger , The Mummy (1999 film)

Figure 4.5: PATTY paraphrases for the YAGO relation *actedIn*

Using PATTY as a Thesaurus

PATTY connects the world of textual surface patterns to the world of predefined RDF relationships. Users who are aware of RDF-based knowledge bases can explore how RDF relations map to their textual representations. For example, as shown in Figure 4.5, PATTY knows over 300 ways in which the YAGO relation *actedIn* can be expressed textually. We hope that this wealth of data can inspire new applications in information extraction, QA, and text understanding.

Users do not need to be familiar with RDF in order to use PATTY. For example, users can find different ways to express the *hasAcademicAdvisor* relation, simply by typing “worked under” into the search box. PATTY also provides the text snippets where the mention was found as a proof of provenance. These text snippets can be explored to understand the context in which a pattern can have a certain meaning. In addition, users can browse the different meanings of patterns, as they occur with different types of entities.

PATTY provides not only relations and patterns, but also a subsumption hierarchy of patterns, where more general patterns subsume more specific patterns. The PATTY subsumptions can be explored by clicking nodes that are linked to the root of a pattern. When a node is clicked, the server retrieves all patterns that imply the activated pattern. Figure 4.6 is a screenshot of a small part of the subsumptions.

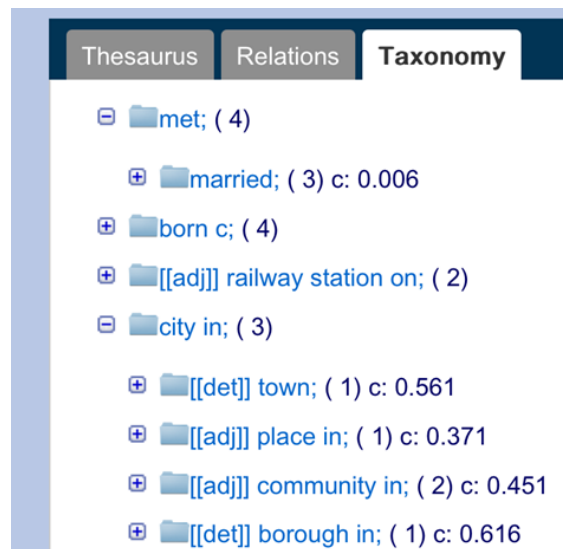


Figure 4.6: A part of the PATTY taxonomy

Schema-Agnostic Search

Internally, PATTY stores all extracted patterns with their support sets. This allows users to search for facts in the database. For this purpose, the PATTY front-end provides a search interface where the user can enter Subject-Predicate-Object triples. Different from existing systems, the user does not have to know the schema of the database (i.e., the relations of the fact triples). It is fully sufficient to enter natural language keywords. For example, to find the costars of Brad Pitt, the user can type

“costarred with”

in place of the relation. PATTY will then search not only for the exact words “costarred with” but also automatically use the paraphrases

“appeared with”,
 “cast opposite”, and
 “starred alongside”,
 see Figure 4.7 .

This way the query only needs to be issued once and the user does not need to do multiple paraphrases as is the case for keyword search engines. For each result, PATTY can show the textual sources from which it was derived.

The type signatures of the patterns can be used to narrow down the search results according to different semantic types. For example, when searching for a pop-

The screenshot shows the PATTY interface with the following components:

- Navigation tabs: Thesaurus, Relations, Taxonomy.
- Search fields: Arg 1 (Brad Pitt), Relation (costarred with), Arg 2 (empty).
- Buttons: Join, Syntax, Semantic, Preferences, Corpora.
- Results table:

Arg1	Arg2	Pattern
Brad Pitt	Juliette Lewis	be reunited with;
Brad Pitt	Anthony Hopkins	also cast as;
Brad Pitt	Heinrich Harrer	led [[adj]];
Brad Pitt	Anthony Hopkins	also cast as;
Brad Pitt	Cate Blanchett	then starred [[ad
Brad Pitt	Antonio Banderas	then starred [[ad
Brad Pitt	Rick Schroder	that played;
Brad Pitt	Cate Blanchett	then starred [[ad
Brad Pitt	Antonio Banderas	then starred [[ad
Brad Pitt	Gwyneth Paltrow	then starred alor
Brad Pitt	Morgan Freeman	then starred alor
- Detail view for 'Brad Pitt also cast as; Anthony Hopkins':
 - Synset: also cast as; later cast; then cast in;
 - Sample sentence: He was cast opposite Anthony Hopkins in the 1994 drama Legends of the Fall , which earned him his first Golden Globe nomination .

Figure 4.7: Schema-free search for costars of Brad Pitt

ular subject like Barack Obama or Albert Einstein, the result may span multiple pages. If the user is interested in only one particular aspect of the entity, then the domain of the subject can be semantically restricted. For example, to see what PATTY knows about Albert Einstein in his role as a scientist, the user can restrict the domain of the relation to *scientist*. Such a query returns Einstein’s teaching positions, his co-authors, information about his theories, etc.; but it does not return information about his wives or political activities.

These schema-agnostic queries can be extended to simple join queries. This works by filling out multiple triples and linking them with variables, similar to the way SPARQL operates. Different from SPARQL, our system does not require the user to know the relation name or the entity names. For example, to find visionaries affiliated with MIT, it is sufficient to type

?x vision ?y, ?x ?z MIT.

This will search for people *?x* who have a vision *?y* and who stand in some relationship *?z* with an entity with name *MIT*. The results are shown in Figure 4.8.

Explaining Relatedness

PATTY can also be used to discover relationships between entities [42]. For example, if the user wishes to know how Tom Cruise and Nicole Kidman are related, it is sufficient to type “Nicole Kidman” into the subject box and “Tom Cruise” into the object box. PATTY will then retrieve all semantic relationships between the two, together with the patterns in which this relationship is expressed. For each result, users can click on the source button discover provenance. The results are shown in Figure 4.9.

The screenshot shows the PATTY interface with three tabs: Thesaurus, Relations, and Taxonomy. The 'Relations' tab is active. The query configuration is as follows:

- Arg 1: ?x
- Relation: ?r
- Arg 2: Massachusetts I
- Buttons: Syntax, Semantic

Below the main query, there is a 'Join' section:

- Arg 3: ?x
- Relation: s vision
- Arg 4: ?y
- Button: Join

Below the join section, there is a section for 'VLDB12 Demo Queries'. At the bottom, there is a table with 3 columns: Arg1, Arg2, and Pattern. The table contains 7 rows of results.

Arg1	Arg2	Pattern
Vannevar Bush	Massachusetts Institute of Technology	has received [[det]] [[adj]] doctorate from;
Tim Berners-Lee	Massachusetts Institute of Technology	also founded at;
Vannevar Bush	Massachusetts Institute of Technology	received [[det]] [[adj]] doctorate from;
Vannevar Bush	Massachusetts Institute of Technology	[[con]] left [[det]];
Tim Berners-Lee	Massachusetts Institute of Technology	founded of;
Vannevar Bush	Massachusetts Institute of Technology	joined then;
Tim Berners-Lee	Massachusetts Institute of Technology	was professor [[num]];

Figure 4.8: Simple join query to find visionaries affiliated with MIT

This principle can be extended to full conjunctive queries. For example, to find the entity that links Natalie Portman and Mila Kunis, the user can type

Natalie Portman ?r ?x, Mila Kunis ?s ?x.

This will find all entities $?x$ that link the two actresses, as well as an explanation of how this entity establishes the link. In the example, PATTY finds a ballet movie for $?x$, and says that both actresses appeared in this movie. The results are shown in Figure 4.10. As this example shows, PATTY has created an internal, semantic representation of the input text documents, which allow her to answer semi-structured queries. In addition to generating semantic patterns, in a sense, PATTY has summarized the input text documents. Users can exploit and query these summaries.

4.12 SUMMARY

This chapter presented PATTY, a large resource of text patterns and the methods for constructing it from Web data. Different from existing resources, PATTY organizes patterns into synsets and a taxonomy, similar in spirit to WordNet. Our evaluation shows that PATTY's patterns are semantically meaningful, and that they cover large parts of the relations of other knowledge bases. The Wikipedia-based version of PATTY contains 350,569 pattern synsets at a precision of 84.7%, with 8,162 subsumptions, at a precision of 75%. The PATTY resource is freely available for interactive access and download at www.mpi-inf.mpg.de/yago-naga/patty/.

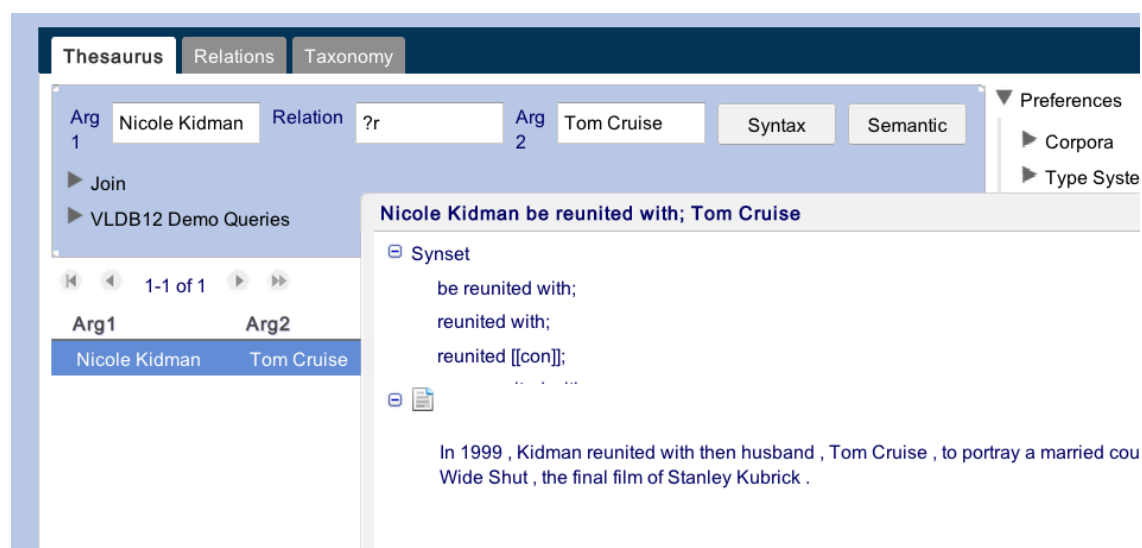


Figure 4.9: Explaining relatedness through a direct triple

Our approach harnesses existing knowledge bases for entity-type information. However, PATTY is not tied to a particular choice for this purpose. In fact, it would be straightforward to adjust PATTY to use surface-form noun phrases rather than disambiguated entities, as long as we have means to infer at least coarse-grained types (e.g., person, organization, location). In the next chapter we show how a system can extract facts pertaining to relations extracted by PATTY but also capable of dealing with out-of-knowledge-base entities.

The screenshot shows the PATTY interface with the 'Relations' tab selected. A query configuration is displayed with the following fields:

- Arg 1: Natalie Portman
- Relation: ?r
- Arg 2: ?x
- Syntax: [button]
- Semantic: [button]

A 'Join' section is expanded, showing a second query configuration:

- Arg 3: Mila Kunis
- Relation: ?s
- Arg 4: ?x
- Join: [button]

Below the configuration, there is a section for 'VLDB12 Demo Queries' and a pagination indicator '1-4 of 4'. A table displays the results of the join query:

Arg1	Arg2	Pattern
Natalie Portman	Black Swan (film)	twice won [[det]];
Mila Kunis	Black Swan (film)	s character from;
Mila Kunis	Black Swan (film)	s character from;
Mila Kunis	Black Swan (film)	twice won [[det]];

Figure 4.10: Explaining relatedness through a join-query

Entity Extraction with PEARL

Both PROSPERA and PATTY rely on a common assumption in ontology-based IE — that there exists a knowledge base which defines all entities and their types. However, when extracting facts from highly dynamic data such as news and social media, new entities emerge that are not in the reference knowledge base. In this chapter, we address the problem of out-of-knowledge-base entities. We present a method for typing previously unseen entities. Our method is based on an Integer Linear Program (ILP) and leverages pattern type signatures and type disjointness constraints. In addition, we demonstrate the value of fact extraction from news and social media through the application of fact-based emerging story identification. Prior work on story identification generates stories as clusters of related entities. No explicit semantic relations between the entities are given.

5.1 MOTIVATION

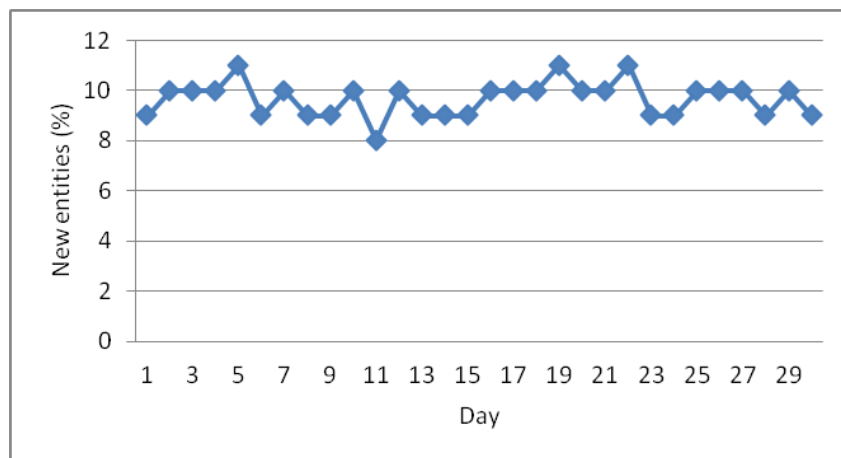


Figure 5.1: Out-of-knowledge-base entities per day

Dynamic content is an important part of the Web, it accounts for a substantial

amount of Web traffic. For example, much time is spent reading news, blogs and user comments in social media. In order to construct comprehensive knowledge bases and avoid staleness, this kind of data must be brought into scope. DBpedia Live addresses the issue of dynamic data updates to the DBpedia ontology. Updates are pulled from Wikipedia via HTTP by accessing the Wikipedia live feed. In this way, Wikipedia changes are extracted and propagated into the DBpedia ontology within minutes after they are made. DBpedia Live extractors are limited to Wikipedia, which is primarily focused on popular entities.

In news and social media, there is a significant number of new entities emerging every day beyond those prominent enough to have a Wikipedia page. To analyze the extent of new entity mentions, we considered a subset of Google news data from July 2012 (see Section 5.8). We looked up every entity mention in YAGO2 [57], and recorded all missing entities. Noun phrases unknown to YAGO2 were recorded as missing, even if they are synonyms of entities already known to YAGO2. Our analysis, shown in Figure 5.1, revealed that about 10% of entities in a given day are out-of-knowledge-base entities. Thus standard ontology-based IE would ignore facts related to *1 in 10* entities for any given day. Lin et al. [72] observed that of all the 15 million facts extracted by Reverb[41], over 5 million facts are related to entities that could not be linked to Freebase [13] entities.

We illustrate the problem of fact extraction from dynamic sources with the example in Figure 5.2. Traditional ontology-based IE only extracts one fact, whereas the desired output contains three facts. Traditional IE fails to extract the facts stated in the first two sentences, as the knowledge base does not contain the relations:

hasRunningMate(person, person) and
ThreatenToHitArea(hurricane, location).

Traditional IE also fails to recognize the name *Hurricane Isaac*— a new entity. Observe that the desired output does not and should not contain any facts from the last sentence because it does not express any relational fact in the sense of a subject-predicate-object triple.

As illustrated by the above example, fact extraction from rapidly updated data sources entails a number of technical challenges that cannot simply be solved by applying standard IE methods. These challenges, while partially addressed by some approaches, such as the recall-oriented Open IE approach, they have not been studied for high-precision ontology-based IE. Prominent among these challenges are: *dynamic entity recognition*, and *timely extraction*.

Dynamic Entity Recognition

Traditional IE extracts facts about disambiguated entities by mapping noun phrases in text to entities in a dictionary of entities. For example, when the noun phrase “Jeff Dean” is encountered in text, it is mapped to the correct entity, which can either be the Google engineer or the rock musician. However, knowledge bases are never complete in the entities they contain. This is in part due to newly emerging entities (e.g., a new hurricane) and entities in the long tail. For example, Jeff

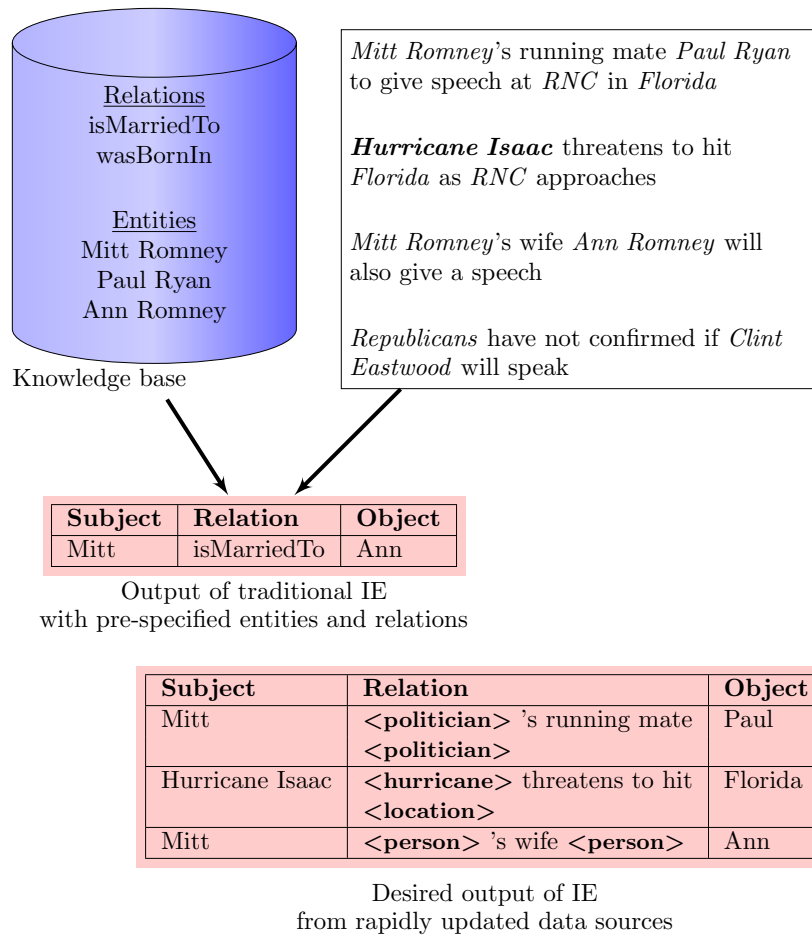


Figure 5.2: Example of output from dynamic sources

Dean the Google engineer is a long-tail entity, he does not have a Wikipedia page. He is therefore missing in Wikipedia-derived knowledge bases. A method for fact extraction on dynamic data needs to recognize and handle out-of-knowledge-base entities as they emerge. This should however go beyond simply treating each noun phrase as a separate entity. Ideally, the output must contain semantics along with noun phrases, facilitating high accuracy fact extraction.

Timely Extraction

A method for fact extraction from rapidly updated sources requires ensuring that new facts are extracted and added to the knowledge base right after they become available as opposed to a few weeks or months down the line. This requires applying methods that solve the above two challenges, more frequently than it is traditionally done in batch-oriented approaches.

5.2 CONTRIBUTION

We present PEARL, a system for extracting facts from rapidly updated data sources. PEARL introduces a method for learning types for *out-of-knowledge-base* entities. Adding *type information* to such new entities yields useful semantics that can be leveraged to boost fact accuracy.

To ensure timely fact extraction, PEARL uses a *continuous processing model* which processes a stream of in-coming documents. PEARL demonstrates the utility of fact extraction on rapidly updated data sources through the *emerging story mining* application. PEARL mines stories as they emerge by clustering related facts. Previous work on story extraction [7, 30] defines stories as clusters of co-occurring entities. The actual *relationships* between the entities are not explicitly stated.

In summary, the chapter’s novel contributions are:

- *A framework for fact extraction from dynamic data*: we introduce a novel pipeline for mining facts from streams of rapidly updated data sources (Section 5.4).
- *Entity typing*: we introduce a method for typing previously unseen entities as they emerge. Our approach learns entity types from the type signatures of the relational patterns they occur with. Additionally, to resolve type incompatibilities, we leverage disjointness constraints by specifying and solving an Integer Linear Program (ILP). (Section 5.6);
- *Emerging story extraction*: we demonstrate that fact extraction from rapidly updated data sources enables novel applications. We develop a graph-based method for mining emerging stories as sets of semantically related facts (Section 5.7).
- *Evaluation*: we evaluate PEARL on real world news data and demonstrate the effectiveness of our methods (Section 5.8).

5.3 RELATED WORK

Our problem is related to literature spanning three areas: named entity recognition, semantic type inference, and event identification.

Named Entity Recognition

Recognizing named entities in text has been studied by a large body of work. Named Entity Recognition (NER) systems aim to detect and assign types to named entities in text. Traditional NER tools such as that by Finkel et. al [45] are coarse-grained and only deal with a few types such as *person*, *organization*, and *company*. Even fine-grained NER [68] methods have only dealt with up to 200 types. In contrast, PEARL deals with thousands of types. Entity disambiguation and entity linking systems map noun phrases in text to entities in an existing knowledge base [17, 51, 57, 76, 88, 114]. The knowledge base is expected to

contain entities, their types, and their key attributes. The main difference between these approaches and PEARL is that the former are primarily concerned with entities that are already known as opposed to discovering and typing out-of-knowledge-base entities.

Inferring Semantic Types

More closely related to our work is the approach of Lin et al. [72] for predicting types for out-of-knowledge-base entities. Two issues are addressed: determining if a noun phrase is an entity or not; and for every noun phrases that is a true entity, determining which Freebase types apply. To determine if a noun phrase is an entity, a timestamped corpus spanning hundreds of years is used. In particular the Google books n-gram corpus from the year 1500 to 2007 is used. The main insight is that true entities have different usage patterns over time than non-entities. Using this insight and features learned from the corpus, their approach can determine that, for example, "Sex and the City" is an entity, while "John and David", "The method", and "12 cats" are not entities. However, this approach is not applicable to emerging entities in news and social media because there is no historical data for such entities. There are methods that employ effective heuristics based on capitalization, singular words, etc., such as those used in [108] Lin et al.'s entity typing method employs a large collection of factual assertions in the form of $\langle \textit{nounphrase1} \rangle \textit{pattern} \langle \textit{nounphrase2} \rangle$. Among those assertions, a large number of them are related to noun phrases that are linkable to Freebase entities, call this set L . The remaining set, U , contains assertions related to unlinkable entities. The unlinkable entities are typed in a three step process:

- **Find Patterns.** For an entity $e \in U$, find a set R of all the patterns in U that co-occur with e in their domain, as opposed to their range. For example, if the entity to be typed is *Sun Microsystems*, and it occurs in the assertion $\langle \textit{Sun Microsystems} \rangle \textit{has released a minor update to} \langle \textit{Java 1.4.2} \rangle$, then the pattern "has released a minor update to" is added to R .
- **Find Similar Entities.** Find the linkable entities in L that occur in the domain, as opposed to the range, of most of the patterns in R . In this case, entities such as *Microsoft* and *Apple* may show up. These entities are added to a set S .
- **Predict Types.** Assign to e the most frequent Freebase types of the entities in S . In this case it may be *business operation*.

While this approach is similar to PEARL, there are key differences. First, Lin et al.'s method does not attempt to resolve inconsistencies among the predicted types, it is left as future work. In contrast, PEARL uses an ILP with type disjointness constraints to solve such inconsistencies. Second, Lin et al.'s method uses untyped patterns. PEARL uses typed patterns which are harnessed for computing type-pattern co-occurrence likelihood estimations. For example, PEARL is able to estimate conditional type-pattern co-occurrence likelihoods as discussed in Section 5.6. Third, Lin et al.'s method only makes use of pattern domains, not

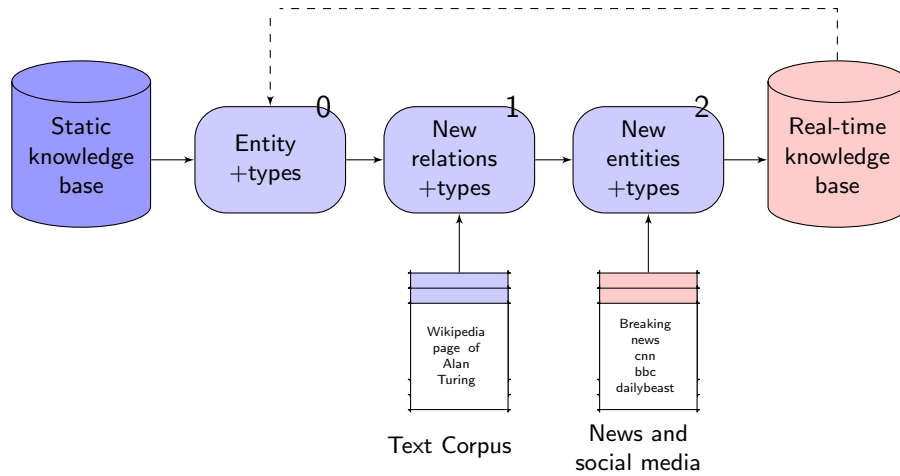


Figure 5.3: Architectural overview of the PEARL approach

ranges. This implies that entities that appear in ranges may not be typed. In contrast, PEARL uses both domains and ranges to infer types.

Event Identification.

Research on story and event identification has largely focused on detecting entities that co-occur at the same time in a bursty manner [7, 30]. The intuition is that if entities receive “buzz” around the same time, then they participate in the same event. An important difference between this line of work and our work is that prior work does not state the relationships that hold between entities participating in an event; thus their events are merely clusters of entities, lacking informative semantics.

Another line of work on event extraction uses patterns to extract special aspects of events. For example, the location and time of earthquakes [103]. In contrast, we are not restricted to extracting specific aspects of an event.

Event extraction has also been applied to micro-blogs. Ritter et al. [101] studied the problem of extracting a calendar of events from *twitter.com*. However, due to their brief nature, such micro-blog posts are not suitable for extracting relational facts. Hence the resulting event descriptions are brief.

5.4 OVERVIEW OF PEARL

PATTY showed that types are a key asset for discovering a large set of relations beyond common ones. Our main insight is that, *semantic types* can also be used to recognize and type out-of-knowledge-base entities.

Figure 5.3 illustrates the overall PEARL approach. As input, our approach takes a knowledge base which contains entities along with their types. In the first step

(marked *step 1* in Figure 5.3), we use PATTY to generate relations in the form of patterns with type signatures. This generates relations like: $\langle actor \rangle$'s *character in* $\langle movie \rangle$. In the second step (marked *2* in Figure 5.3), we use the typed phrases, extracted in *step 1*, to extract facts. Different from standard ontology-based IE, we also extract facts that refer to out-of-knowledge-base entities. For such new entities, we learn their types from the phrases they occur with.

The output of our system are relational facts of the form:

$\langle Entity1 \rangle$ $\langle domain \rangle$ *phrase* $\langle range \rangle$ $\langle Entity2 \rangle$.

An example fact in this format is:

$\langle Mitt Romney \rangle$ $\langle politician \rangle$'s *running mate* $\langle politician \rangle$ $\langle Paul Ryan \rangle$.

5.5 DETECTION OF NEW ENTITIES

To detect noun phrases that potentially refer to entities, we apply a part-of-speech tagger to the input text.

For a given noun phrase, there are four possibilities:

- a) The noun phrase refers to a general concept (a class or abstract concept), not an individual entity.
- b) The noun phrase is a known entity that can be directly mapped to the knowledge base.
- c) The noun phrase is a new name for a known entity.
- d) The noun phrase is a new entity not known to the knowledge base at all.

Case a) refers to non-entities, we would like to discard such noun phrases. To decide if a noun phrase is a true entity (i.e., an individual entity that is a member of one or more lexical classes) or a non-entity (i.e., a common noun phrase that denotes a class or a general concept), we base the decision on the following hypothesis (inspired by and generalizing [17]):

Hypothesis 5.5.1 (Entity Detection Hypothesis) *A*

given noun phrase, not known to the knowledge base, is a true entity if its headword is singular and is consistently capitalized (i.e., always spelled with the first letter in upper case).

Here the notion of headword refers to noun-group parsing which decomposes a noun phrase into its semantic anchor (the headword), one or more pre-modifiers and one or more post-modifiers. For example, in “the first black president of the United States” the headword is “president”.

Case b) refers to entities already in the knowledge base. Therefore, this case can be handled by a named entity disambiguation tool such as AIDA [57]. We treat case c) equivalently to case d), because if the name does not appear in the knowledge base there is no way of knowing which entity it refers to. Note

that we already leverage KBs like YAGO2 that contain many millions of entities with a huge number of surface names (synonyms) derived from anchor texts and redirects of Wikipedia links. Our task thus is to compute semantics types for noun phrases falling under cases c) and d). The noun phrases can then be added to the knowledge base with their inferred types.

5.6 TYPING OUT-OF-KNOWLEDGE-BASE ENTITIES

To deduce types for new entities we propose to align new entities along the type signatures of patterns they occur with. In this manner we use the patterns to suggest types for the entities they occur with. In particular, we infer entity types from pattern type signatures. Our approach builds on the following hypothesis:

Hypothesis 5.6.1 (Type Alignment Hypothesis) *For a given pattern such as $\langle actor \rangle$'s character in $\langle movie \rangle$, we assume that an entity pair (x, y) frequently occurring with the pattern in text implies that x and y are of the types $\langle actor \rangle$ and $\langle movie \rangle$, respectively.*

Challenges and Objective

While the type alignment hypothesis works as a starting point, it introduces false positives. Such false positives stem from the challenges of polysemy, fuzzy pattern matches, and incorrect paths between entities.

Polysemy

The same lexico-syntactic pattern can have different type signatures. For example, the following are four different patterns:

$\langle singer \rangle$ released $\langle album \rangle$,
 $\langle music_band \rangle$ released $\langle album \rangle$,
 $\langle company \rangle$ released $\langle product \rangle$,
 $\langle country \rangle$ released $\langle prisoner \rangle$.

For an entity pair (x, y) occurring with the pattern "released", x can be one of four different types ($singer$, $music_band$, $company$, $country$) and y can be one of three different types ($album$, $product$, $prisoner$).

Fuzzy Pattern Matches

We cannot expect that the phrases we extract in text will be exact matches of the typed relational patterns we have learned with PATTY. Therefore, for better recall, we must accept fuzzy matches. Quite often however, the extracted phrases matches multiple relational patterns to various degrees. Each of the matched relational patterns has its own type signature. The type signatures of the various matched patterns can be incompatible with one another.

Incorrect Paths between Entities

This problem emerges when a pair of entities occurring in the same sentence do not stand in a subject-object relation. For example, consider the sentence: *Liskov graduated from Stanford and obtained her PhD degree from MIT*. In this sentence, there is no relationship between Stanford and MIT, however we may erroneously extract: *[Stanford] obtained her PhD degree from [MIT]*. If Stanford were an unknown entity and we had a semantically-typed pattern which says people obtain PhDs from institutions, then we would wrongly infer that Stanford is a person.

Deep linguistic processing such as dependency parsing facilitates correct path finding between entity pairs. However, dependency parsing does not adequately solve the issue. Web sources contain a plethora of sentences that are not well-formed. Such sentences mislead the dependency parser to extract wrong dependencies.

Our solution takes into account polysemy, fuzzy matches, as well as issues stemming from potential incorrect-path limitations. We define and address the following optimization problem:

Definition 5.6.2 (Type Inference Optimization) *Given all the candidate types for x , find the best types or “strongly supported” types for x . The final solution must satisfy type disjointness constraints. Type disjointness constraints are constraints that indicate that, semantically, a pair of types cannot apply to the same entity at the same time. For example, a $\langle \text{university} \rangle$ cannot be a $\langle \text{person} \rangle$. We also study a relaxation of type disjointness constraints through the use of type correlation constraints.*

Our task is therefore twofold: first, generate candidate types for new entities; second, find the best types for each new entity among its candidate types.

Candidate Types for Entities

The candidate types for a new entity are the possible types that can be assigned to it based on its occurrences with typed relational patterns.

Definition 5.6.3 (Candidate Type) *A new entity x appears in a number of triples $(x, p_1, y_1), (y_2, p_2, x), \dots, (y_n, p_n, x)$. We say that x is supported by patterns p_1, p_2, \dots, p_n . Each pattern p_i has a type signature with a domain and a range. If x occurs as a left-hand argument, we consider the domain of p_i as a candidate type for x . If x occurs as a right-hand argument, we consider the range of p_i as a candidate type for x . Hence the triples (x, p_i, y) and (y, p_i, x) are treated differently.*

We compute confidence weights for candidate types. Ideally, if an entity occurs with a pattern which is highly specific to a given type then the candidate type should have high confidence. For example “is married to” is more specific to people than “expelled from”. A *person* can be expelled from an *organization* but a *country* can also be expelled from an *organization* such as NATO.

There are various ways to compute weights for *candidate types*. We first introduce a uniform weight approach and then present a method for computing more informative weights.

Uniform Weights

Suppose a new entity x occurs in text with phrases $(x \text{ phrase}_1 y_1)$, $(x \text{ phrase}_2 y_2)$, ..., $(x \text{ phrase}_n y_n)$. Suppose these occurrences lead to the facts $(x, p_1 y_1)$, $(x, p_2 y_2)$, ..., $(x, p_n y_n)$. The p_i s are the *typed relational patterns* extracted by PATTY. The facts are generated by matching *phrases* to relational patterns with type signatures. The type signature of a pattern is denoted as:

$$\text{sig}(p_i) = (\text{domain}(p_i), \text{range}(p_i))$$

We allow fuzzy matches, hence each fact comes with a match score. This is the similarity degree between the phrase observed in text and the typed relational pattern.

Definition 5.6.4 (Fuzzy Match Score) Suppose we observe the surface string: $(x \text{ phrase } y)$ which leads to the fact: x, p_i, y . The fuzzy match similarity score is: $\text{sim}(\text{phrase}, p_i)$, where similarity is the n -gram Jaccard similarity between the phrase and the typed pattern.

The confidence that x is of type *domain* is:

Definition 5.6.5 (Candidate Type Confidence) For a given observation $(x \text{ phrase } y)$, where *phrase* matches patterns p_1, \dots, p_n , with domains d_1, \dots, d_b which are possibly the same:

$$\text{typeConf}(x, \text{phrase}, d) = \sum_{\{p_i: \text{domain}(p_i)=d\}} (\text{sim}(\text{phrase}, p_i))$$

Observe that this sums up over all patterns that match the phrase.

To compute the final confidence for $\text{typeConf}(x, \text{domain})$, we aggregate the confidences over all *phrases* occurring with x .

Definition 5.6.6 (Aggregate Confidence) For a set of observations $(x, \text{phrase}_1, y_1)$, $(x, \text{phrase}_2, y_2)$, ..., $(x, \text{phrase}_n, y_n)$, the aggregate candidate type confidence is given by:

$$\begin{aligned} \text{aggTypeConf}(x, d) &= \sum_{\text{phrase}_i} \text{typeConf}(x, \text{phrase}_i, d) \\ &= \sum_{\text{phrase}_i} \sum_{\{p_i: \text{domain}(p_i)=d\}} (\text{sim}(\text{phrase}_i, p_i)) \end{aligned}$$

The confidence for the range $\text{typeConf}(x, \text{range})$ is computed analogously. All confidence weights are normalized to values between 0 and 1.

The limitation of this approach is that each pattern is considered equally good for suggesting *candidate types*. Thus this approach does not take into account the intuition that an entity occurring with a pattern which is highly specific to a given type is a stronger signal that the entity is of the type suggested. Our next approach addresses this limitation.

Co-occurrence Likelihood Weight Computation

We devise a likelihood model for computing weights for entity *candidate types*. Central to this model is the estimation of the likelihood of a given type occurring with a given pattern.

Suppose using PATTY methods we mined a typed relational pattern $\langle t_1 \rangle p \langle t_2 \rangle$. Suppose that we now encounter a new entity pair (x, y) occurring with a *phrase* that matches p . We can compute the likelihood of x and y being of types t_1 and t_2 , respectively, from the likelihood of p co-occurring with entities of types t_1, t_2 . Therefore we are interested in:

Definition 5.6.7 (Type-Pattern Likelihood) *The likelihood of p co-occurring with an entity pair (x, y) of the types (t_1, t_2) is given by:*

$$P[t_1, t_2 | p] \quad (5.1)$$

where t_1 and t_2 are the types of the arguments observed with p from a corpus such as Wikipedia. $P[t_1, t_2 | p]$ is expanded as:

$$P[t_1, t_2 | p] = \frac{P[t_1, t_2, p]}{P[p]}. \quad (5.2)$$

The expressions on the right-hand side of Equation 5.2 can be directly estimated from a corpus as follows:

Definition 5.6.8 (Corpus-based Estimation) $P[t_1, t_2, p]$ is the relative occurrence frequency of the typed pattern among all entity-pattern-entity triples in a corpus (e.g., the fraction of $\langle \text{musician} \rangle$ plays $\langle \text{song} \rangle$ among all triples).

$P[p]$ is the relative occurrence frequency of the untyped pattern (e.g., plays) regardless of the argument types. For example, this sums up over both $\langle \text{musician} \rangle$ plays $\langle \text{song} \rangle$ occurrences and $\langle \text{actor} \rangle$ plays $\langle \text{fictional character} \rangle$.

We use the Wikipedia English version to for corpus-based estimations. If we observe a fact where one argument name can be easily disambiguated to a knowledge-base entity so that its type is known, and the other argument is considered to be an out-of-knowledge-base entity, we condition the joint probability of t_1, p , and t_2 in a different way:

Definition 5.6.9 (Conditional Type-Pattern Likelihood) *The likelihood of an entity of type t_1 occurring with a pattern p and an entity of type t_2 is given by:*

$$P[t_1 | t_2, p] = \frac{P[t_1, t_2, p]}{P[p, t_2]} \quad (5.3)$$

where the $P[p, t_2]$ is the relative occurrence frequency of a partial triple, for example, $\langle * \rangle$ plays $\langle \text{song} \rangle$.

Observe that all numbers refer to occurrence frequencies. For example, $P[t_1, p, t_2]$ is a fraction of the total number of triples in a corpus.

Multiple patterns can suggest the same type for an entity. Therefore, the weight of the assertion that y is of type t , is the total support strength from all phrases that suggest type t for y .

Definition 5.6.10 (Aggregate Likelihood) *The aggregate likelihood candidate type confidence is given by:*

$$\text{typeConf}(x, \text{domain}) = \sum_{\text{phrases}} \sum_{p_i} (\text{sim}(\text{phrase}_i, p_i) * L)$$

Where $L = P[t_1, t_2 | p]$ OR $P[t_1 | t_2, p]$ OR $P[t_2 | t_1, p]$

The confidence weights are normalized to values in $[0, 1]$.

So far we have presented a way of generating a number of *weighted candidate types* for x . In the next step we pick the best types for an entity among all its candidate types.

Integer Linear Program Formulation

Given a set of *weighted candidate types*, our goal is to pick a compatible subset of types for x . The additional asset that we leverage here is the compatibility of types: how likely is it that an entity belongs to both type t_i and type t_j . Some types are mutually exclusive, for example, the type *em location* rules out *em person* and, at finer levels, *city* rules out *river* and *building*, and so on. Our approach harnesses these kinds of constraints.

Our solution is formalized as an Integer Linear Program (ILP). We have candidate types for x : t_1, \dots, t_n . First, we define a decision variable T_i for each candidate type $i = 1, \dots, n$. These are binary variables: $T_i = 1$ means type t_i is selected to be included in the set of types for x , $T_i = 0$ means we discard type t_i for x .

In the following we develop two variants of this approach: a “hard” ILP with rigorous disjointness constraints, and a “soft” ILP which considers type correlations.

“Hard” ILP with Type Disjointness Constraints

We infer type disjointness constraints from the YAGO2 knowledge base using occurrence statistics. Types with no overlap in entities or insignificant overlap below a specified threshold are considered disjoint. Notice that this introduces *hard constraints* whereby selecting one type of a disjoint pair rules out the second type. We define type disjointness constraints $T_i + T_j \leq 1$ for all disjoint pairs t_i, t_j (e.g. *person-artifact*, *movie-book*, *city-country*, etc.). The ILP is defined

as follows, where the weights w_i are the aggregated likelihoods derived in the previous subsection:

objective

$$\max \sum_i T_i \times w_i$$

binary variables

$$\forall_i T_i \in \{0, 1\}$$

type disjointness constraint

$$\forall (t_i, t_j)_{\text{disjoint}} T_i + T_j \leq 1$$

“Soft” ILP with Type Correlations

In many cases, two types are not really mutually exclusive in the strict sense, but the likelihood that an entity belongs to both types is very low. For example, few drummers are also singers. Conversely, certain type combinations are boosted if they are strongly correlated. An example is guitar players and electric guitar players.

Our second ILP considers such soft constraints. To this end, we pre-compute *Pearson correlation* coefficients for all type pairs (t_i, t_j) based on co-occurrences of types for the same entities. These values $v_{ij} \in [-1, 1]$ are used as weights in the objective function of the ILP. We additionally introduce pair-wise decision variables Y_{ij} , set to 1 if the entity at hand belongs to both types t_i and t_j , and 0 otherwise. This coupling between the Y_{ij} variables and the T_i, T_j variables is enforced by specific constraints.

For the objective function, we choose a linear combination of the per-type evidence, using weights w_i as before, and the type-compatibility measure, using weights v_{ij} . The ILP with correlations is defined as follows:

objective

$$\max \alpha \sum_i T_i \times w_i \times (1 - \alpha) \sum_{ij} Y_{ij} \times v_{ij}$$

binary variables

$$\forall_i T_i \in \{0, 1\}$$

type correlation constraints

$$\forall_{i,j} Y_{ij} + 1 \geq T_i + T_j$$

$$\forall_{i,j} Y_{ij} \leq T_i$$

$$\forall_{i,j} Y_{ij} \leq T_j$$

Note that both ILP variants need to be solved per entity, not over all entities together. The “soft” ILP has a size quadratic in the number of candidate types, but this is still a tractable input for modern solvers. We use the Gurobi software package to compute the solutions for the ILP’s.

For both the “hard” and “soft” variants of the ILP, the solution is the best types for entity x satisfying the constraints. We can thus use these types to emit the final output of PEARL, which is relational facts of the form:

$$\langle \text{Entity1} \rangle \langle \text{domain} \rangle \text{phrase} \langle \text{range} \rangle \langle \text{Entity2} \rangle.$$

5.7 EMERGING STORY MINING

As an application for fact extraction on news we study emerging story mining. There have been previous studies on emerging story mining [7, 30, 101]. However, we are not aware of any methods that create stories as sets of facts with semantic relations. Instead, previous methods have relied on a general form of relatedness to generate stories. They primarily use “temporal” co-occurrences. Two entities are said to be in the same story if they co-burst; co-occurring frequently over a given time period [30]. In contrast, our method produces relationships between pairs of entities. In this regard, the stories we generate are more descriptive than stories of sets of entities. We define a story as follows:

Definition 5.7.1 (Story) *Given a set Φ of all extracted facts, where a fact is an entity-(typed)pattern-entity triple $(e1, p, e2)$, a story is a set of facts $S_i \subset \Phi$ that are closely related.*

Intuitively, two documents are likely to be discussing the same story if they have a large overlap in the facts they state. Formally we have:

Definition 5.7.2 (Fact-based Document Similarity) *The fact-based similarity of a pair of documents $d1$ and $d2$ is quantified by the Jaccard similarity of their facts.*

$$\begin{aligned} \text{sim}(d1, d2) &= \text{Jaccard}(d1, d2) \\ &= \frac{|\{\text{facts} \in d1\} \cap \{\text{facts} \in d2\}|}{|\{\text{facts} \in d1\} \cup \{\text{facts} \in d2\}|} \end{aligned}$$

Computing fact-based document similarity can be done efficiently for document pairs. This information is then used to build the *document fact graph*.

Definition 5.7.3 (Document-Fact Graph) *A document-fact graph $G = (V, E)$ is a weighted undirected graph consisting of a node set V representing documents and an edge set E representing fact-based document similarity between documents. Each edge has a weight w indicating the fact-based document similarity score.*

Intuitively, the *document-fact graph* contains dense clusters of subgraphs. Each such dense cluster expresses a story. Our task is to mine stories from the graph. To solve this task, algorithms for identifying dense subgraphs can be applied. Our approach is based on the idea of random walks [54]. We run a random walk starting at a document node v , and obtain a ranking score for each visited node $v_j \in V$. The ranking score of a visited node reflects how likely the document is part of the same story. We can apply this algorithm to every document node in the graph; however, it would not be clear what the stories are. To counter this, we introduce the concepts of *major documents* and *supporting documents*.

Definition 5.7.4 (Major & Supporting Documents) Given a document fact graph G , a document node v_i is a *major document* if it has a large number of immediate neighbors. The document is said to be a *major document* for a yet to be discovered story S . $V^* \in V$ is the set of all nodes corresponding to major documents. For a given story S_i , its *supporting documents* is given by nodes V_i^- , which are the nodes that can be reached from one of the story's major document through random walks.

The intuition is that a *major document* is vital to the story. Such a document states facts that are central to the story. It may also contain some non-central facts. We hypothesize that facts that are central to a story are repeated in most documents pertaining to that story. Such facts are usually given as background information to the reader. This means that a *major document* v has a large number of immediate neighbors (above a specified threshold). Nodes in v 's neighborhood repeat v 's set of facts to some degree. An example story is shown in Figure 5.4. The *major document* in the story is $d1$. The rest of the documents, $d2, d3, d4, d5$, contain some of the facts stated by $d1$ and are therefore *supporting documents*.

Thus the story formation algorithm begins by identifying all *major documents*. The next step is to form neighborhoods around the major documents. For given a *major document* $v \in V^*$, we want to compute a relevance score for the rest of the nodes in the graph. Relevance scores are computed by random walks that start with an initial distribution over the nodes where the entry corresponding to a major document is set to one and all other entries are set to zero. As in standard Markov chains, the probability of taking a particular edge is proportional to the edge weight over all the outgoing edges. Moreover, walks are occasionally restarted based on coin tosses with probability ϵ ; in these cases, the restart always jumps back to the original starting point. The result of the random walks is an approximation of the stationary distribution of reaching other nodes from the given major document.

All the nodes whose relevance score (with respect to the major document) is above a threshold constitute the *supporting documents*. Note that a document can be a supporting document in multiple stories. *Algorithm 5.1* describes the story formation procedure.

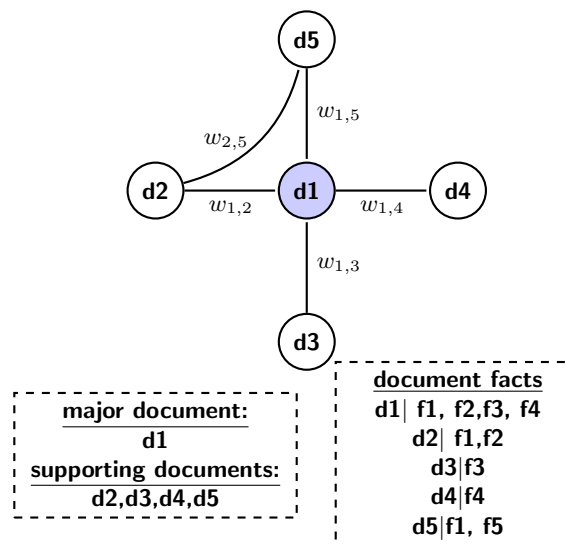


Figure 5.4: A fact graph depicting a single story

Algorithm 5.1 Story Formation

```
1: procedure ExtractStories( $G$ )
2:    $S \leftarrow \emptyset$ ; // initialize story set
3:    $V^* \leftarrow \emptyset$ ; // initialize major docs
4:   for node  $v_i \in G$  do
5:      $L(v_i) \leftarrow$  neighbors of  $v_i$ ;
6:     if  $|L(v_i)| >$  threshold  $\theta$ 
7:        $V^* \leftarrow V^* \cup \{v_i\}$ ;
8:   endfor
9:    $R \leftarrow$  computeRandomWalkScores( $G$ );
10:   $V_i^- \leftarrow \{v_j \in V : (R_j > \epsilon)\}$ 
17:   $S^i \leftarrow V_i^- \cup \{v_i\}$ ;
11:   $S \leftarrow S \cup S_i$ ;
12: endfor
13: return  $S$ 
```

5.8 EVALUATION

Setup

Data Collection

To define a suitable corpus of test data, we obtained a continuous stream of news documents by subscribing to *Google News* RSS feeds for a few topics starting in April 2012. As of October 2012, the accumulated document count is 318,434. All our experiments are carried out on this data. We subscribed to eleven news topics, namely: *Angela Merkel*, *Barack Obama*, *Business*, *Entertainment*, *Hillary Clinton*, *Joe Biden*, *Mitt Romney*, *Newt Gingrich*, *Rick Santorum*, *SciTech* and *Top News*. For the reference knowledge base of known entities, we used YAGO2 [58]. The type system used is that of the YAGO2, which is derived from WordNet. For disambiguating mentions of known entities, we used AIDA [57].

Quality Assessment

All human evaluations were carried out using Amazon Mechanical Turk (MTurk). MTurk is a platform for crowd-sourcing tasks which require human input. Tasks on MTurk are small questionnaires consisting of a description and a set of questions. The questions can be single-choice, multiple-choice or open (text). In order to minimize effects of mistakes or cheating, each task was submitted to multiple turkers.

Methods under Comparison

For the typing of out-of-KB entities, we compared PEARL against two state-of-the-art baselines:

- **NNPLB** (No Noun Phrase Left Behind), the method of [72], based on the propagation of types for known entities through salient patterns occurring with both known and unknown entities. We re-implemented the algorithm of [72] in our framework, using the relational patterns of PATTY [81] for comparability. For assessment we sample from the top-5 highest ranked types for each entity.
- **HYENA** Hierarchical tType classification for Entity NAMES), the method of [132], based on a feature-rich classifier for fine-grained, hierarchical type tagging. This is a state-of-the-art representative of similar methods such as [96, 73].

Evaluation of Entity Typing

We evaluated the quality of types PEARL assigned to new entities emerging in the news stream. The turkers were presented with sentences from the news tagged with out-of-KB entities and the types PEARL inferred for the entities. The task was

	PEARL		HYENA		NNPLB	
	total	%	total	%	total	%
num. of. entity-type pairs	105	100%	105	100%	105	100%
majority VG or SAT	82	78%	26	25%	48	46%
majority VG	61	58%	14	13%	27	26%
points > 0	78	74%	24	23%	44	42%
avg. points	2.62		-2.42		-0.79	
at least one VG or SAT	94	90%	59	56%	72	69%
at least one NS or W	40	38%	91	87%	85	81%

Table 5.1: Results for entity typing (NS: *not sure*, W: *wrong*, S: *satisfactory*, VG: *very good*).

to assess the correctness of types assigned to an entity mention. For example, the input sentence could be *Angela Merkel has arrived for the Brussels Summit, and poured cold water over hopes of a major breakthrough over the next two days*. The extracted entity and type were combined into a sentence to make understanding the task easier. For example if PEARL inferred that Brussels Summit is an political event, we generate and present the sentence: *Brussels Summit is an event*. We allowed four possible assessment values:

- a) *Very good* output corresponds to a perfect result.
- b) *Satisfactory* output exhibits minor errors. For instance, the description *G20 Summit is an organization* is wrong, because the summit is an event, but G20 is indeed an organization. The problem in this example is incorrect segmentation of a named entity.
- c) *Wrong* for incorrect types (e.g., *Brussels Summit is a politician*).
- d) *Not sure / do not know* for other cases.

Comparing PEARL to Baselines

We evaluated 105 entity-type test samples, using out-of-KB entities which were mentioned frequently in the news corpus: in at least 20 different news articles. Each test sample was given to 3 different turkers for assessment.

Since the turkers did not always agree if the type for a sample is good or not, we aggregate their answers. We count how many times an entity type was classified as *not sure* (NS), *wrong* (W), *satisfactory* (S) and *very good* (VG). We use majority voting to decide whether the type was assigned correctly to an entity. We consider the following variants of voting: i) we treat very good and satisfactory as correct, ii) only very good counts as correct. Additionally, we aggregate the answers using a point scoring system: a *very good* judgment is assigned +2 points, satisfactory +1, wrong -2 (and not sure is discounted). We report the average number of points and the number of samples with scores ≥ 0 .

Table 5.1 shows the results of how PEARL performed versus the NNPLB and HYENA methods. PEARL outperformed both opponents on all metrics. On the metric of majority votes of Very Good or Satisfactory, PEARL achieved 78% pre-

κ	<i>Fleiss</i>	<i>Cohen</i> _{VG}	<i>Cohen</i> _{VGVSat}
<i>PEARL</i>	0.31	0.40	0.44
<i>HYENA</i>	0.26	0.27	0.31
<i>NNPLB</i>	0.21	0.26	0.37
<i>PEARL-all</i>	0.52	0.59	0.68
<i>PEARL-soft</i>	0.39	0.58	0.44

Table 5.2: Inter-judge agreement kappa: Fleiss’ κ & adapted Cohen’s κ .

cision compared to HYENA’s 25% and NNPLB’s 46%. When we count samples where at least one turker assigned Very Good or Satisfactory, PEARL reaches 90% precision compared to 56% for HYENA and 69% for NNPLB. Anecdotic examples for entity typing by PEARL are shown in Table 5.6. We also compare sample results of the same entities as ranked lists of types returned by each of the methods in Table 5.7. Some of the samples provide insight into where limitations of the baselines lie.

As shown in the samples in Table 5.7, HYENA’s output contains a great deal of noise. HYENA’s relatively poor performance can be attributed to the fact that its features are mainly syntactic such as bi-grams and part-of-speech tags. Web data is challenging, it has a lot of variations in syntactic formulations. This introduces a fair amount of ambiguity which can easily mislead syntactic features. Leveraging semantic features as done by PEARL could improve HYENA’s performance. Furthermore, HYENA fails to assign types to some entities that PEARL finds types for. For example, *Melinda Liu* is correctly typed by PEARL as a *journalist* but both HYENA and NNPLB failed to assign any types to it. Furthermore, HYENA assigns negatively correlated types to the same entity. For a given entity, the decision if a given type is to be assigned to the entity is made independent of the other types, however PEARL jointly makes decisions about all candidate types in its ILP.

While the NNPLB method performs better than HYENA, in comparison to PEARL, there is room for improvement. Like HYENA, the NNPLB method also fails to assign types to some entities which PEARL correctly finds types for. This results from NNPLB’s focus on entities that appear in the subject role. This limitation could be addressed by also considering entities in the object role in a similar manner to the way PEARL does. Also like HYENA, NNPLB assigns negatively correlated types to the same entity. This limitation could be addressed by applying PEARL’s ILPs to the candidate types suggested by NNPLB.

Inter-judge agreements of the human assessors are given in Tables 5.2 and 5.4. For Table 5.2 we calculated the Fleiss’ kappa and Cohen’s kappa κ , which are standard measures for inter-judge agreement. The standard assumption for Fleiss’ κ is that labels are categorical, so that each disagreement counts the same. This is not the case in our settings, where different labels may indicate partial agreement. Therefore, Fleiss κ is a lower-bound measure of agreement in our experiments; the “true agreement” seems higher. Nevertheless, the observed Fleiss κ values show that the task was fairly clear to the turkers; values > 0.2 are generally considered as acceptable [67]. Cohen’s κ assumes that there are only two judges and two classes. Since it is not the case in our experiments, we had to

	PEARL		PEARL-all		PEARL-soft	
	total	%	total	%	total	%
num of. entity-type pairs	105	100%	105	100%	105	100%
majority VG or SAT	82	78%	69	66%	56	53%
majority VG	61	58%	57	54%	39	37%
points > 0	78	74%	68	65%	49	47%
avg. points	2.62		1.80		0.31	
at least one VG or SAT	94	90%	82	78%	82	78%
at least one NS or W	40	38%	45	43%	67	64%

Table 5.3: Results for entity typing by PEARL variants (NS: *not sure*, W: *wrong*, S: *satisfactory*, VG: *very good*).

	PEARL				PEARL-all				PEARL-soft				HYENA				NNPLB			
	Not Sure	Wrong	Sat.	V. Good	Not Sure	Wrong	Sat.	V. Good	Not Sure	Wrong	Sat.	V. Good	Not Sure	Wrong	Sat.	V. Good	Not Sure	Wrong	Sat.	V. Good
Not sure	0	2	2	4	0	6	2	6	0	7	5	8	0	8	2	4	1	17	8	9
Wrong	-	43	37	15	-	76	19	17	-	88	50	25	-	163	32	52	-	105	38	49
Sat.	-	-	12	71	-	-	9	39	-	-	18	31	-	-	8	20	-	-	7	38
V. good	-	-	-	129	-	-	-	141	-	-	-	83	-	-	-	26	-	-	-	43

Table 5.4: Inter-judge agreement in entity-typing assessments.

adapt it. First, we converted the assessments into two categories. We did it in two ways: in $Cohen_{VG \vee Sat}$ we merge *Very good* with *Satisfactory* and *Wrong* with *Not sure*, whereas in $Cohen_{VG}$ we merged *Satisfactory*, *Wrong* and *Not sure* into one category. Next, we found pairs of judges who assessed at least 10 identical entity-type pairs and calculated Cohen’s κ for each such pair. The reported results are averages over such pairs. For all evaluations, Cohen’s κ is > 0.2 .

For inter-judge agreements we additionally report in Table 5.4 how many times a particular assessment co-occurred for the same test sample, where a great deal of agreement among the assessors can be seen.

PEARL Variants

We also evaluated several variants of PEARL. The previous numbers refer to the variant with “hard ILP” evaluated on frequent entities. We now report on configurations with “soft ILP” on frequent entities, and with the default configuration of “hard ILP” for randomly sampled entities including infrequent ones. The results are shown in Table 5.3. As expected, the precision when sampling over all entities drops a bit. This can be attributed to the fact that some of the infrequently occurring noun phrases are not real entities. While our entity-detection heuristics weeds out many non-entities, some still remain. Furthermore, for infrequent entities, PEARL does not have enough evidence for reliable type assignments. The configuration with “soft ILP”, with α set to 0.5, generally performed worse than the “hard ILP” variant. Nevertheless, it remains an interesting option because of the tunable α , which allows better control over precision-recall trade-offs. So if

an application requires high recall, an appropriately tuned “soft ILP” variant of PEARL could be the method of choice.

NDCG

For assigning types to new entities, for a given entity e , an entity-typing system return a ranked list of types $\{t_1, t_2, \dots, t_n\}$. We evaluated the ranking quality, using the top-5 ranks for each method, by asking MTurk users to label each type with scores 2 (Very Good), 1 (Satisfactory), or 0 (Wrong). These assessments are aggregated into the normalized discounted cumulative gain (NDCG), a widely used measure for ranking quality. We computed NDCG for three methods: PEARL, HYENA, and NNPLB. The results are shown in Table 5.5. Here again, we see that PEARL outperforms all other methods by a large margin.

	NDCG
PEARL	0.53
HYENA	0.16
NNPLB	0.16

Table 5.5: NDCG for various entity-typing methods.

Fuzzy Score Evaluation

We varied the minimum fuzzy match score between phrases and the typed relational phrases. A low minimum fuzzy score means that we permit more uncertain matches to suggest *candidate types* for entities. For this we evaluated three methods. The first is full PEARL with hard constraints. The second, denoted *No ILP*, is a version of PEARL that does not attempt to resolve type inconsistencies but leverages *type-pattern co-occurrence likelihood* weights. The third, denoted *Uniform Weights* is a version of PEARL that does not use *type-pattern co-occurrence likelihood* weights, but resolves type inconsistencies.

To assess precision, we selected a random sample of 60 out-knowledge-base entities extracted in July 2012. For a given entity in the sample, we evaluated the types suggested for it by each of the three methods. An entity type assignment by a given method is marked correct if *all* types the methods assigns to an entity are correct.

Figures 5.5, 5.6, and 5.7 present performance results of the three methods. PEARL outperforms the other methods by significant margins. For the case of minimum fuzzy match score of 0.4 (shown in Figure 5.5, full results presented in *Appendix B*), PEARL achieves 85% precision compared to *No ILP* with 28% precision and the *Uniform Weights* method with 50% precision. This shows that both the ILP and corpus-learned weights are crucial for high precision. For the second case, seen in Figure 5.6, we set the minimum fuzzy match score to 0.8. While PEARL still outperforms the other methods, its precision drops to 77%. This is because using a high threshold means that most of the support for type suggestions is lost. These results show that fuzzy matches are important for both

high recall, as well as for boosting support for the correct types. Furthermore, a high minimum match score leads to PEARL's relative recall dropping to 52% (see Figure 5.7). Recall was computed with respect to the number of typed entities, out of the 60 in the sample set.

Table 5.8 shows sample output of the three methods. PEARL correctly types non-trivial entities. For example, PEARL correctly types the entity *Minaj* as a *musician*. The *Uniform Weights* method types *Minaj* as an *athlete*. The *No ILP* method types it as both an *athlete* and *musician*.

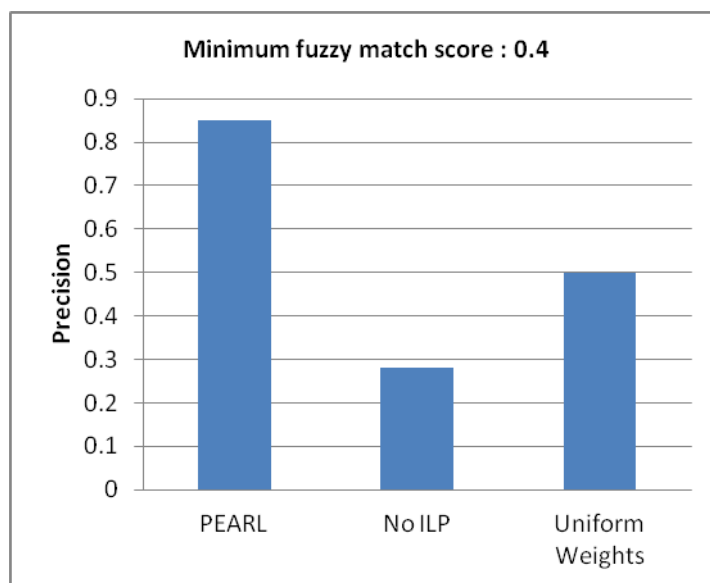


Figure 5.5: Precision of entity typing methods at 0.4 minimum fuzzy match score.

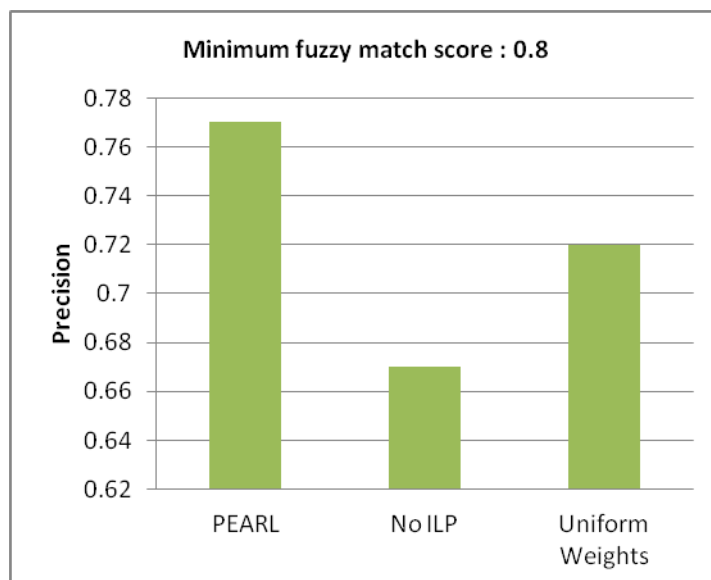


Figure 5.6: Precision of entity typing methods at 0.8 minimum fuzzy match score.

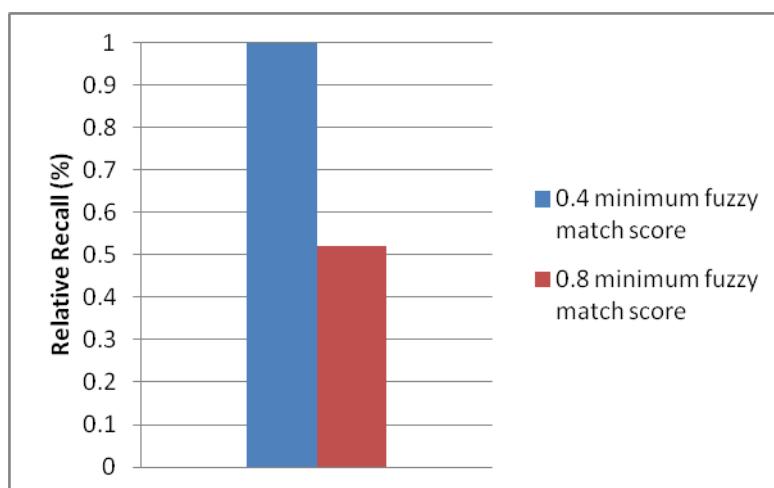


Figure 5.7: Relative recall for varying degrees of minimum fuzzy matches.

Entity	Inferred Type	Source Sentence (s)
Lochte	medalist	Lochte won America’s lone gold on the first day of swimming competition.
Malick	director	Turn the clock back 15 months, and Brad Pitt, Sean Penn and Jessica Chastain all graced the red carpet in Cannes for Malick’s 2011 movie , “ The Tree of Life”.
Bonamassa	musician	Bonamassa recorded Driving Towards the Daylight in Las Vegas with a mix of veteran studio musicians including drummer Anton Fig from the Late Show with David Letterman band and Nashville bass ace Michael Rhodes. At the age of 12, Bonamassa opened for B.B. King in Rochester , N.Y. “It was a thrill”, he said and in 2009 he fulfilled a dream by performing at the Royal Albert Hall in London, where Eric Clapton made a guest appearance.
Analog Man	album	Analog Man is Joe Walsh’s first solo album in 20 years.
Rep. Debbie Wasserman Schultz	person	Thomas Roberts speaks with Rep. Debbie Wasserman Schultz , chair of the Democratic National Committee, about a new Quinnipiac Poll that shows ...
LightSquared	organization	LightSquared paid Boeing some \$1 billion for two satellites with the largest antenna receivers ever put into space, one of which was launched and is circling the Earth now.
Melinda Liu	journalist	“My fervent hope is that it would be possible for me and my family to leave for the U.S. on Hillary Clinton’s plane,” Chen said in a telephone interview with journalist Melinda Liu of the Daily Beast.
U.S. Border Patrol Agent Brian Terry	military officer	The inspector general determined that ATF agents and federal prosecutors had enough evidence to arrest and charge Jaime Avila, a Phoenix gun smuggler, months before Border Patrol Agent Brian Terry was killed near Tucson in December 2010.
RealtyTrac	publication	Earlier this month, RealtyTrac reported that for the first time since it began compiling foreclosure statistics in 2005, Illinois had the highest foreclosure rate among all the states in August.

Table 5.6: Example source sentences for PEARL results.

Entity	PEARL	HYENA	NNPLB
Lochte	person, medalist	painting, device, exile, artifact , person	person, scholar, politician, athlete, musician
Malick	person, director	event, battle	person, manufacturer, director, actor, photographer
Bonamassa	person, musician, actor	event	person, musician, singer
Analog Man	artifact, album	noble, scholar, aristocrat, person, artifact	artifact, song, album, musical_composition
Rep. Debbie Wasserman Schultz LightSquared	person organization, company	— organization , medalist, artifact, device	— organization, scholar, person, director,
Melinda Liu	person, journalist	—	—
U.S. Border Patrol Agent Brian Terry	person, military_officer	person	person, military_officer, head_of_state, president
RealtyTrac	artifact, publication	organization, book, executive, ruler	artifact, device, publication

Table 5.7: Example results from PEARL, HYENA, NNPLB

New Entity	Proposed Types		
	PEARL	No ILP	Uniform Weights
Sen. Dianne Feinstein D-Calif	person	person, artifact, prisoner, publication	person
MSNBC show	artifact	artifact, country, city	city
Mountain Lion	event, movie	event, artifact, person, movie	event, movie
Knowledge Graph	artifact	artifact, event	artifact
ABCNews.com	person	person, artifact, event, publication,	person
Prevention magazine	person	person, artifact, event, publication	artifact, publication
Singer Taylor Swift	person	person, country	person
Cee Lo Green	person	person, organization, team	team
Justice Elena Kagan	person	person, organization, event, university	person
Lena Dunham	actor, manufacturer, person	person, actor, event, judge	actor, musician, person
Minaj	musician, person, pianist, singer	person, musician, athlete, actor	athlete, basketball_player, football_player, person

Table 5.8: Sample results for typing out-of-knowledge-base entities

Story Mining Evaluation

We also conducted a user study on the use-case of story mining. We compare two story representations: a set of entities versus a set of facts. MTurk users were presented with a description of a story in one of the two representations and a list of three news headlines. The headlines were picked from documents belonging to different sets of stories. One of the three headlines was from a document of the described story; this ground-truth was not known to the turkers. The turkers were then asked to select the headline which matches the described story. In addition to the three headlines we provided a fourth option of "Not sure". To make the task less obvious, we presented headlines which belong to stories that overlap in the associated entities (but still leaving enough entities that occur in only one of the three stories). We evaluated 50 story descriptions for each of the two representation models: set of entities vs. set of facts. Each story description was presented to three turkers, resulting in 300 assessments in total. The results are shown in Table 5.9.

	<i>Precision</i>
<i>PEARL (set-of-facts representation)</i>	45%
<i>baseline (set-of-entities representation)</i>	38%

Table 5.9: Quality assessment of story representations.

The set-of-facts representation outperformed the set-of-entities representation. However, the results also show that precision is still low at 45%. This is due to the situation that important facts of a story cannot always be extracted into explicit *subject-predicate-object* triples. While devising a method for capturing complex relationships is beyond the scope of this dissertation, it makes an interesting line for future work.

Additionally, to assess precision of the stories mined by our algorithm, we randomly selected a sample of 30 stories. A given story is marked correct by a human judge if it corresponds to a real-life story. It is marked wrong otherwise. A real-life story is a story that can be categorized as a specific type of event, for example, a court case, a death, a campaign event or an announcement of a significant new movie role for an actor. Of the 30 evaluated stories, 21 (70 %) were marked as real-life stories (see Table 5.10). Table 5.11 shows the types of stories found in the evaluated samples. Almost all of the evaluated stories were of different event types. The exception is the category of *market share competition* which has two stories - one on market share competition among browsers, and the other on Twitter and Facebook advertisement market share competition.

Total	Correct	Precision
30	21	70%

Table 5.10: Precision of PEARL stories

Event type	# of stories
murder case	1
firing/replacement	1
high profile meeting	1
campaign attack	1
death	1
sports event reschedule	1
legal trial	1
new movie role	1
market share competition	2
initial public offering	1
sports match	1
political crisis	1
stock Markets	1
event cancellation	1
product announcement	1
police investigation	1
band reunite	1
lawsuit	1
rape scandal	1
prostitution scandal	1
pregnancy	1

Table 5.11: A categorization of the evaluated stories.

5.9 SUMMARY

This chapter addressed the problem of dealing with out-of-knowledge-base entities in ontology-based IE. Considering such entities is crucial when extracting facts from highly dynamic data sources such as news and social media. Our analysis of a real-world news stream showed that about 10% of daily entity mentions were unknown to the YAGO2 knowledge base. The chapter presented PEARL, a method for typing previously unseen entities. PEARL is based on an Integer Linear Program (ILP) and leverages pattern type signatures and type disjointness constraints. Our evaluation on the same news stream showed that PEARL outperformed two baselines. We also demonstrated the value of extracting facts from news through the emerging story mining application. Our evaluation showed that the stories we extracted correspond to real-world stories.

Conclusion

KNOWLEDGE BASES, due to their potential to boost progress in building intelligent systems, are being embraced in both the commercial world and research. This trend has resulted in a variety of long-running knowledge base construction projects. Commercial search engines have recently started rolling out knowledge-backed features, enabling consumers to forgo clicking on hyperlinks, instead presenting direct answers. This and many other applications have solidified the need to pursue the challenging but rewarding endeavor of harvesting knowledge for machine understanding. This chapter recaps the contributions of the dissertation and brings forward some directions for future research.

6.1 CONTRIBUTIONS

The first contribution of the dissertation is the PROSPERA system for fact extraction. PROSPERA reconciles high precision, high recall and scalability. However, PROSPERA relies on a reference knowledge base to define binary relations for extraction and to provide a dictionary of entities. The next two contributions have addressed freeing PROSPERA, and ontology-guided fact extraction in general, from limited binary relations and incomplete dictionaries of entities. PATTY extracts semantically typed relational patterns. The PATTY resource contains significantly more binary relations than existing knowledge bases. PEARL presented a method for semantically enhancing out-of-knowledge-base entities, making them valuable for high-accuracy fact extraction. The issue of out-of-knowledge-base entities is especially relevant to the setting of fact extraction from highly dynamic sources such as news and social media. PEARL demonstrated the value of extracting facts from such sources through the application of fact-based emerging story mining.

6.2 FUTURE DIRECTIONS

While this dissertation has made significant progress toward comprehensive knowledge base construction, this is a grand-scale challenge and much work remains to be done. In what follows we discuss several directions for future work.

Fact Extraction Across Sentences

We focused on extracting facts from a single sentence. However, it is very common for related information to span across multiple sentences. In such cases, the use of pronouns such as "he" or "she" is very common. While approaches from natural language processing research have been proposed, much remains to be done toward developing general methods for resolving such pronouns. There is an opportunity for so called *big data* methods to play a role, leveraging massive corpora from the Web.

Unbounded Relations

The methods presented in this dissertation have focused on extracting relations where patterns appear between a pair of entities. For example from the sentence:

Barbara Liskov is a professor at MIT,

the relation $\langle \text{professor} \rangle \text{ faculty at } \langle \text{university} \rangle$ can easily be extracted. However, suppose the sentence is phrased differently as follows:

A professor at MIT, Barbara Liskov has been awarded the 2008 Turing Award.

The phrase indicating the relationship between MIT and Liskov is not confined between two entities and is therefore not extracted. Exploring such unbounded relations could further boost recall.

Structured Provenance

The Web model of authorship means that anyone can state anything. Sometimes this leads to conflicting views. Mutual exclusion constraints can be used to infer the most likely ground truth of facts. However, applying such constraints is limited to a few relations such as $\langle \text{person} \rangle \text{ born on } \langle \text{date} \rangle$. However we can leverage provenance information to annotate facts with their sources. Some statements are directly attributed to their sources. Hence we can exploit sentences of the form: *According to X, Y is Z.* A structured provenance model can be incorporated into knowledge bases. Applications using the data can leverage such provenance information to treat different facts differently based on the authority and trustworthiness of the sources they come from.

Relations about Relations

In PATTY we focused on two types of relatedness: synonymy and hypernymy. However many more types of relatedness between binary relations can be extracted. For example, we can also extract antonyms — where one relation is the opposite of another. Furthermore, some relations have units, we could extract the units of relations like *height*, *GDP*, etc. In addition, some relations have value constraints, for example, it is not possible for a person’s height to be 5 meters.

Typing N-nary Relations

In PATTY we focused on mining a large collection of binary relations. However, certain relations are more naturally expressed as n -nary relations, where $n > 2$. Such relations might be better suited to certain settings, such as explaining complex relatedness and causality relations.

Non-Entities

In PEARL we focused on typing out-of-knowledge-base entities. However, it is equally important to ask the question of which noun phrases are real entities and which ones are not. Methods that make use of historical usage patterns are not applicable to new entities. While heuristics can be effective, there is an opportunity to also explore methods combining linguistic processing and data-mining techniques.

PATTY vs. Other Resources

This appendix shows the details of the comparison between PATTY and existing resources as introduced in *Chapter 4*. In particular, we compare PATTY's relations to those found in YAGO, DBpedia, Freebase and NELL.

A PATTY VS. OTHER RESOURCES

Relation	Patty	YAGO	DBpedia	Freebase	NELL
eponymFor (person) × (person)	(person) named after (person)	—	namedAfter	—	—
hasNationality (person) × (country)	(person) DET citizen of (country)	isCitizenOf	(person) citizenship (*)	country of nationality	personHasCitizenship
hasAncestry (person) × (country)	(person) of descent (country)	—	—	ethnicity	—
diedFrom (person) × (event)	(person) died from (catastrophe) complications of	—	deathCause	cause of death	—
bornInCity (person) × (city)	(person) born in (city)	wasBornIn	birthPlace	place of birth	personborninlocation
diedInCity (person) × (city)	(person) died in (city)	diedIn	deathPlace	place of death	—
buriedAt (person) × (location)	(person) buried at (city)	—	placeOfBurial	place of burial	—
knownFor (person) × (entity)	(person) known for (artifact)	isKnownFor	known for	—	—
wasNominatedFor (person) × (award)	(person) nominated for (artifact)	—	—	award nominations	—
hasWonPrize (person) × (award)	(person) was awarded (artifact)	hasWonPrize	award	awards won	—
hasReligiousBackground (person) × (religion or belief)	(person) was raised as (organization)	—	—	—	—
believesInReligion (person) × (religion)	—	—	religion	religion	—
convertedToReligion (person) × (religion)	(person) converted to (organization)	—	—	—	—
influencedByPerson (person) × (person)	(person) influence by (person)	influences	influencedBy	influenced by	—
adoresGroup (person) × (organization)	(person) fan of (organization)	—	—	—	—
attendedSchool (person) × (school)	(person) attended (organization)	—	formerHighschool	—	—
attendedUniversity (person) × (university)	(person) attended at (university)	—	almaMater	—	—
obtainedDegreeFrom (person) × (university)	(person) obtained degree from (university)	graduatedFrom	—	—	—
droppedOutOf (person) × (university)	(person) had dropped out (university)	—	—	—	—
wasExpelledFrom (person) × (school)	(person) expelled from (organization)	—	—	—	—
hasBodyMark (person) × (body covering)	—	—	—	—	—
hasHairStyle (person) × (hairdo)	—	—	—	—	—
livedInCity (person) × (city)	(person) lives in (city)	—	residence	places lived	—

Relation	Patty	YAGO	DBpedia	Freebase	NELL
raisedInCity (person) × (city)	(person) raised in (city)	—	—	—	—
madeTripTo (person) × (location)	(person) trip to (country)	—	—	—	—
hasMother (person) × (person)	(person) 's mother (person)	—	—	—	—
hasFather (person) × (person)	(person) 's father (person)	—	—	—	—
hasSister (person) × (person)	(person) 's sister (person)	—	—	—	—
hasBrother (person) × (person)	(person) 's brother (person)	—	—	—	—
hasDaughter (person) × (person)	(person) 's daughter (person)	—	—	—	—
hasSon (person) × (person)	(person) 's son (person)	—	—	—	—
hasChild (person) × (person)	(person) 's child (person)	hasChild	—	—	—
adoptedChild (person) × (person)	(person) adopted (person)	—	—	—	—
adoptedSon (person) × (person)	(person) adopted (person)	—	—	—	—
adoptedDaughter (person) × (person)	(person) adopted (person)	—	—	—	—
hasGrandParent (person) × (person)	(person) 's grandmother (person)	—	—	—	—
hasGodChild (person) × (person)	—	—	—	—	—
hasGodDaughter (person) × (person)	—	—	—	—	—
hasGodSon (person) × (person)	—	—	—	—	—
hasGodFather (person) × (person)	—	—	—	—	—
hasGodMother (person) × (person)	—	—	—	—	—
hasGodMotherInLaw (person) × (person)	—	—	—	—	—
hasGodFatherInLaw (person) × (person)	—	—	—	—	—
hasDaughterInLaw (person) × (person)	—	—	—	—	—
hasSonInLaw (person) × (person)	(person) son in law of (person)	—	—	—	—
marriedTo (person) × (person)	(person) is married (person)	isMarriedTo	spouse	spouse	hasspouse
divorcedFrom (person) × (person)	(person) divorced from (person)	—	—	—	—
estrangedFrom (person) × (person)	(person) PRP estranged (person)	—	—	—	—
hasHusband (person) × (person)	(person) 's husband (person)	isMarriedTo	spouse	spouse	wifeof
hasWife (person) × (person)	(person) 's wife (person)	isMarriedTo	spouse	spouse	husbandof
hasChildhoodFriend (person) × (person)	(person) hasChildhoodFriend (person)	—	—	—	—
hasFriend (person) × (person)	(person) friend to (person)	—	—	celebrity friends	—
hasBoyfriend (person) × (person)	(person) 's boyfriend (person)	—	—	—	—
hasGirlfriend (person) × (person)	(person) 's girlfriend (person)	—	partner	—	—
knows (person) × (person)	(person) where got to know (person)	—	—	—	—
hasLoveRelationWith (person) × (person)	(person) relationship with (person)	—	—	romantically involved with	—
strainedRelationWith (person) × (person)	(person) PRP estranged (person)	—	—	—	—
reunitedWithPerson (person) × (person)	(person) reunited with (person)	—	—	—	—
reunitedWithGroup (person) × (group)	(musician) reunited with (organization) ADJ members of	—	—	—	—

A PATTY VS. OTHER RESOURCES

Relation	Patty	YAGO	DBpedia	Freebase	NELL
influencedByMusician (musician) × (musician)	(person) influenced by (person)	—	influencedBy	influenced by	—
influencedByBand (musician) × (group)	(person) influenced by groups ADJ (organization)	—	influencedBy	influenced by	—
influencedByGenre (person) × (genre)	—	—	—	—	—
influencedMusician (musician) × (musician)	(musician) influenced ADJ artists (musician)	—	influencedBy	influenced by d	—
influencedBand (person) × (group)	—	—	influencedBy	influenced	—
influencedGenre (person) × (genre)	—	—	—	—	—
hasMusicalIdol (person) × (group)	(musician) PRP idol (musician)	—	—	—	—
learnedFrom (musician) × (musician)	(person) learned from (person)	—	—	—	—
discoveredTalent (musician) × (musician)	(musician) discovered CON managed CON managed ADJ performers (artist)	—	—	—	—
wroteHitsFor (musician) × (musician)	(musician) wrote hits for (musician)	—	—	—	—
hasSingingVoice (musician) × (voice type)	—	—	voiceType	vocal range	—
playsInstrument (musician) × (musical instrument)	(musician) plays (musical_instrument)	hasMusicalRole	instrument	instruments played	musician-plays-instrument
playsGenre (musician) × (music genre)	—	hasGenre	genre	—	music-artistgenre
releasesAlbum (musician) × (album)	(musician) 's album (album)	—	album	musical album	—
releasesDebutAlbum (person) × (group)	(musician) 's debut album (album)	—	—	—	—
releasesDebutAlbum (person) × (group)	(musician) 's debut album (album)	—	—	—	—
releasesLiveAlbum (musician) × (album)	—	—	—	live album	—
releasesAcousticAlbum (musician) × (album)	—	—	—	—	—
releasesSingle (musician) × (song)	(song) 's hit song (song)	—	—	—	—
releasesDebutSingle (musician) × (song)	(song) DET song from PRP PRP debut album (album)	—	—	—	—
wroteSong (musician) × (song)	(musician) song written by (song)	—	writer	works composed	—
wroteSong (person) × (song)	(song) lyrics written by (person)	—	lyrics	lyrics written	—
dedicatedSongTo (musician) × (person)	(musician) dedicated DET song to (person)	—	—	dedications	—
performsForLabel (musician) × (organization)	(musician) signed to ADJ record label (organization)	—	—	record label	—
hasProducer (musician) × (person)	(person) produced ADJ artists including (musician)	—	producer	record producer	—
hasManager (musician) × (person)	(musician)'s manager (person)	—	manager	tour manager	—
hasRepresentative (person) × (musician)	(person) represented ADJ artists ADJ (musician)	—	—	—	—
hasSoundEngineer (musician) × (person)	(person) recording engineer [[con]] producer for ADJ client (singer)	—	—	recording engineer	—
memberOf (musician) × (group)	(musician) member of (organization)	—	currentMember	member of	—
co-founded (musician) × (group)	(musician) cofounded DET band (organization)	—	foundingPerson	founders	—
recruited (musician) × (musician)	(musician) recruited DET players (musician)	—	—	—	—
playedWithBand (musician) × (group)	(musician) has performed with DET band (organization)	—	—	—	—
playedWithPerson (musician) × (musician)	(musician) performed along (musician)	—	associatedMusicalArtist	—	—
hadDuetWith (musician) × (musician)	(musician) duet with (musician)	—	—	—	—
singsWith (musician) × (musician)	(musician) sings with (musician)	—	—	—	—

A PATTY VS. OTHER RESOURCES

Relation	Patty	YAGO	DBpedia	Freebase	NELL
recordedAtLocation (musician) × (location)	(album) sings with (city)	—	recordedIn	place recorded	—
performedAt (musician) × (event)	(musician) performed PRP ADJ concert in (artifact)	—	—	concert venue	—
performedAtLocation (musician) × (location)	(musician)'s concert (city)	—	—	concert venue	—
headlinerAt (musician) × (event)	(musician) headliner (artifact)	—	—	—	—
headlinerAt (person) × (event)	(person) involved in (event)	—	—	—	—
invitedByPerson (person) × (person)	(person) invited by (person)	—	—	—	—
rejectedPerformance (person) × (event)	—	—	—	—	—
canceledPerformance (person) × (event)	—	—	—	—	—
performedForPerson (person) × (event)	(musician) has performed for (person)	—	—	—	—
performedForOrg (musician) × (organization)	(musician) has performed for (organization)	—	—	—	—
covered (musician) × (song)	(musician) covered DET song (song)	—	—	—	—
coveredBy (musician) × (musician)	(musician) covered by (musician)	—	—	—	—
wroteSoundtrackFor (musician) × (movie)	(musician) wrote DET music of (movie)	wroteMusicFor	—	soundtrack	—
parodied (musician) × (person)	(person) was parodied (person)	—	—	—	—
appearedOnTV (person) × (broadcast)	(person) ADJ television appearances (event)	—	—	broadcast artists	—
appearedOnRadio (person) × (broadcast)	(musician) has appeared on radio stations(artifact)	—	—	broadcast artists	—
appearedInFilm (person) × (movie)	(person) appeared in films (movie)	actedIn	—	film appearances	—
starredInFilm (person) × (movie)	(person) starred in (movie)	actedIn	starring	film appearances	actor-starredin-movie
madeFilm (person) × (movie)	(person) directed(movie)	directed	director of	films directed	director-directed-movie
appearedInMedia (person) × (medium)	(person) appeared on cover of (publication)	—	—	—	personwritten-aboutin-publication
featuredIn (person) × (artifact)	(person) featured in (artifact)	—	—	—	—
interviewedBy (person) × (person)	(person) interviewed by (person)	—	—	interviews given	—
praisedByPerson (person) × (person)	(person) praised by (person)	—	—	—	—
praisedByPerson (person) × (person)	(person) praised by (person)	—	—	—	—
praisedFor (person) × (entity)	—	—	—	—	—
criticizedByPerson (person) × (person)	(person)'s critique of (person)	—	—	—	—
praisedByMedia (person) × (medium)	—	—	—	—	—
criticizedByMedia (person) × (medium)	(person)criticized by (organization)	—	—	—	—
criticizedAt (person) × (event)	—	—	—	—	—
criticizedFor (person) × (entity)	—	—	—	—	—
gotTributeFrom (person) × (person)	(person) paid tribute (person)	—	—	—	—
gaveTributeTo (person) × (person)	(person) tribute to (person)	—	—	—	—
hasSculptureAt (person) × (organization)	(person) sculpture at (organization)	—	—	—	—
hasExhibitionAt (person) × (organization)	(person) exhibition in (organization)	—	—	—	—
hasWorkAnalyzedBy (person) × (person)	(person) analysis of (person)	—	—	—	—

A PATTY VS. OTHER RESOURCES

Relation	Patty	YAGO	DBpedia	Freebase	NELL
wroteBook (person) × (book)	(person) author of (artifact)	created	author	author	writerwrote-book-
wrotePoem (person) × (artifact)	(person) wrote PRP poem(artifact)	created	—	works written	—
hasPublisher (person) × (company)	(person) published by (company)	—	publisher	publisher	—
licensedRightsTo (person) × (organization)	(organization) bought DET remain (person)	created	author	rights holder of	—
joinedCampaignFor (person) × (entity)	(person) campaigned for(organization)	—	—	—	—
advocateFor (person) × (entity)	(person) advocate for (organization)	—	—	—	—
supportedByPerson (person) × (person)	(person) supported by (person)	—	—	—	—
supportedByOrg (person) × (organization)	(person) supported by (organization)	—	—	—	—
criticOf (person) × (entity)	(person) critique of (person)	—	—	—	—
wasDrunkAt (person) × (event)	—	—	—	—	—
misbehavedAt (person) × (event)	—	—	—	—	—
tookDrug (person) × (drug)	—	—	—	—	—
addictedTo (person) × (drug)	—	—	—	substance abuse problems	—
suffersFrom (person) × (disease)	(person) suffers from (artifact)	—	—	—	—
diagnosedWith (person) × (disease)	(person) diagnosed with (artifact)	—	—	—	—
hospitalizedFor (person) × (event)	—	—	—	—	—
treatedFor (person) × (condition)	—	—	—	—	—
treatedUsing (person) × (drug)	—	—	—	—	—
hadAccidentAt (person) × (location)	(person) accident in (city)	—	—	—	—
arrestedFor (person) × (entity)	(person) arrested for (event)	—	—	—	—
accusedOf (person) × (entity)	(person) been accused of (event)	—	—	—	—
admitted (person) × (entity)	(person) admitted to (event)	—	—	—	—
sentencedFor (person) × (drug)	—	—	—	—	—
sentencedTo (person) × (drug)	—	—	—	—	—
signedContractWith (person) × (company)	(person) is signed with (company)	—	—	—	—
workedFor (person) × (organization)	(person) worked for (organization)	worksAt	employer	employer	agentbelongs-toorganization
workedWith (person) × (person)	(person) collaborated with (person)	—	—	—	—
owns (person) × (organization)	(person) owns DET (organization)	—	owner	owner	-
launched (person) × (person)	(person) launched DET (organization)	—	—	—	—
loanedTo (person) × (person)	(person) loaned to (organization)	—	—	—	—
donatedTo (person) × (person)	(person) donated to (organization)	—	—	—	—
earnedAt (person) × (event)	(person) earned for (event)	—	—	—	—
suedBy (person) × (event)	(person) successfully sued (person)	—	—	legal entanglements	—
sued (person) × (entity)	(person) sued (person)	—	—	—	—
suedFor (person) × (event)	(person) sued for (artifact)	—	—	—	—

PEARL Evaluation

In this appendix we present the output of the PEARL variants for the 60 out-of-knowledge-base entities corresponding to results reported in *Chapter 5*. We present results for PEARL, PEARL without an (ILP) , and Pearl with uniform weights.

New Entity	Proposed Types		
	PEARL	PEARL without ILP	PEARL with uniform weights
Kim Jong Eun	military_officer, person	person, organization, event, country	military_officer, person
Mr Thompson	person	person, organization, enterprise	enterprise, organization
NAACP convention	event	event, person, organization, artifact	event
London Games	event, festival	event, city, person, medalist	event, festival
Rupert Sanders	director, person	person, river, director, musician	musician, person
Bush tax	artifact	artifact, person, athlete, football_player,	athlete, football_player, person
Susan Ellis	person	person, event, artifact, company	businessman, businessperson, person
Judge Colin Birss	head_of_state, person	person, organization, ship, artifact	head_of_state, person
Adam Mann	actor, musician, person	person, artifact, organization, event	actor, musician, person
Google +	artifact	artifact, event, person	artifact

New Entity	Proposed Types		
	PEARL	PEARL without ILP	PEARL with uniform weights
GOP convention	event	event, country, city, artifact	country
New York Mayor Rudy Giuliani	businessman, businessperson, person	person, organization, musician, artifact	head_of_state, person, president, scholar
Cavill	person	person, event, scholar	person
Twitter CEO Dick Costolo	person	person, event, scholar	person
Jerri DeVard	person	person	person
Galaxy Tab	person	person, organization, artifact, criminal	criminal, person
Russian Foreign Minister Sergey Lavrov	person	person	person
Scott Lanman	person	person, country, artifact, scholar	person
Dina ElBoghdady	person	person, event, artifact, movie	person
Hilaria Thomas	person	person	person
Mark O'Mara	person	person, organization, team journalist	journalist, person
Treasury Secretary Timothy F. Geithner	artist, composer, person	person, writer, artist, musician	head_of_state, person, president, scholar

New Entity	Proposed Types		
	PEARL	PEARL without ILP	PEARL with uniform weights
Carlina White	actor, person	person, organization, artifact, event	actor, musician, person
Misty Cook-Morrissey	actor, person	person, actor, country, artifact	actor, person
Peregrine Financial Group PFG	musician, person	person, singer, musician	actor, musician, person
Hyon Yong Chol	person, priest, spiritual_leader	person, organization, artifact, company	athlete, director, person, trainer
George Zimmerman	person	person, organization, team, event	event, team,
Mr. Sedlak	athlete, director, football_player, person,	person, organization, athlete, artist	athlete, director, football_player, person
Rami Abd al-Rahman	person	person, event, scholar	person
Chicago Mayor Rahm Emanuel	artist, person	person, event, country, artist	artist, composer, guitarist, person
Trayvon Martin	person	person	person
Obama campaign official	journalist, person	person, journalist, writer	journalist, person
Jerry del Missier	person	person, politician, musician	musician, person

New Entity	Proposed Types		
	PEARL	PEARL without ILP	PEARL with uniform weights
Josh Earnest	person	person	person
Prime Minister Salam Fayyad	head_of_state, person	person, artifact, country, city	military_officer, person
Kathryn McCormick	actor, person	person, actor, event, artifact	actor, musician, person
European Union officials	artist, manufacturer, person	person, organization, event, artifact	military_officer, person, scholar
Conn.	city	city, country, organization, person	country
LeBron	athlete, person	person, country, organization, wrestler	athlete, person
Alex Okrent	person	person, event, spiritual_leader, saint	musician, person
Mrs Merkel	person	person, country, organization	person
Gabby Douglas	person	person, artifact, event	person
Galaxy S III	artifact	artifact	artifact
ECB President Mario Draghi	person	person	
Ore.	city	city, artifact	city

B PEARL VS. BASELINES

New Entity	Proposed Types		
	PEARL	PEARL without ILP	PEARL with uniform weights
AuthenTec	company, organization	person, organization, artifact, event	artist, person,
Investor Marc Andreessen	person	person	person
Ric Gillespie	person	person, organization, artifact, politician	artist, person, photographer
Gen. Manaf Tlass	head_of_state, person	person, sovereign, ruler, head_of_state	head_of_state, person,
Eric Dunham	person	person, military_officer	military_officer, person
Olympic trials	event	event organization, university	organization, university
Ariz.	city	city, artifact, person, scholar	city
Citicorp	person	person, organization	person
Wimbledon title	event	event	event
Nanette Kinkade	person, event, artist, musician	artist, person	artist, painter, person
Andreessen Horowitz	company, organization	organization, company, enterprise	enterprise, organization,
Elaine Barrish	governor, person	person, politician, governor, actor	governor, person
S.C.	country	country, organization	country
Md.	city	city, country, artifact, event	country
Bloomberg News	artifact, publication	artifact, publication	artifact, publication



Bibliography

- [1] R. Agrawal, T. Imielinski, A.N. Swami: Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 207–216, Washington, D.C., USA, 1993.
- [2] M. K. Agarwal, K. Ramamritham, M. Bhide: Real Time Discovery of Tense Clusters in Highly Dynamic Graphs: Identifying Real World Events in Highly Dynamic Environments. In *Proceedings of the VLDB Endowment*, PVLDB 5(10), pages 980–991, 2012.
- [3] E. Agichtein, L. Gravano: Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the ACM Digital Libraries Conference*, pages 85–94, San Antonio, TX, USA,
- [4] E. Alfonseca, M. Pasca, E. Robledo-Arnuncio: Acquisition of instance attributes via labeled and related instances. In *Proceeding of the 33rd International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 19–23, Geneva, Switzerland, 2010.
- [5] *First International Workshop on Automatic Knowledge Base Construction (AKBC)*, 2010.
- [6] I. Androutsopoulos, P. Malakasiotis: A Survey of Paraphrasing and Textual Entailment Methods. *Journal of Artificial Intelligence Research* 38, pages 135–187, 2010.
- [7] A. Angel, N. Koudas, N. Sarkas, D. Srivastava: Dense Subgraph Maintenance under Streaming Edge Weight Updates for Real-time Story Identification. In *Proceedings of the VLDB Endowment*, PVLDB 5(10):574–585, 2012.
- [8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z.G. Ives: DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, pages 722–735, Busan, Korea, 2007.
- [9] K. Banker: MongoDB in Action. *Manning*, 2011.
- [10] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni: Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676, Hyderabad, India, 2007.
- [11] J. Berant, I. Dagan, J. Goldberger: Global Learning of Typed Entailment Rules. In *Proceedings of the 49th Conference of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 610–619, Portland, Oregon, USA. 2011.
- [12] T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web. *Scientific American*, 284(5), pages 34–43, 2001.
- [13] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor: Freebase: a Collaboratively Cre-

- ated Graph Database for Structuring Human Knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages, 1247-1250, Vancouver, BC, Canada, 2008.
- [14] S. Brin. Extracting Patterns and Relations from the World Wide Web. In *Selected papers from the International Workshop on the WWW and Databases (WebDB)*, pages 172–183, London, UK, 1999.
- [15] L. D. Brown, T. Tony Cai, A. Dasgupta: Interval Estimation for a Binomial Proportion. *Statistical Science* 16, pages 101–133, 2001.
- [16] R. Bunescu, R. Mooney: Extracting Relations from Text: From Word Sequences to Dependency Paths. *Text Mining & Natural Language Processing*, 2007.
- [17] R. C. Bunescu, M. Pasca: Using Encyclopedic Knowledge for Named entity Disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, 2006.
- [18] M. J. Cafarella, A.Y. Halevy, D. Z. Wang, Eugene Wu, Yang Zhang: WebTables: Exploring the Power of Tables on the Web. In *Proceedings of the VLDB Endowment*, PVLDB, 1(1), pages 538–549, 2008.
- [19] M. J. Cafarella: Extracting and Querying a Comprehensive Web Database. In *Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, 2009.
- [20] A. Carlson, J. Betteridge, R.C. Wang, E.R. Hruschka, T.M. Mitchell: Coupled Semi-supervised Learning for Information Extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining (WSDM)*, pages 101–110, New York, NY, USA, 2010.
- [21] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, T.M. Mitchell: Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, Atlanta, Georgia, USA, 2010.
- [22] S. Ceri, M. Brambilla (Eds.): *Search Computing: Challenges and Directions*. Springer, 2009.
- [23] P. Cimiano and J. Völker: Text2onto - a Framework for Ontology Learning and Data-driven Change Discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages 227–238, Alicante, Spain, 2005.
- [24] M.-W. Chang, L.-A. Ratinov, N. Rizzolo, D. Roth: Learning and Inference with Constraints. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 1513–1518, Chicago, Illinois, USA, 2008.
- [25] P. Chen, R. M. Verma: A Query-Based Medical Information Summarization System Using Ontology Knowledge. In *Proceedings of the 19th IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 37–42, Salt Lake City, Utah, USA, 2006.
- [26] ClueWeb09 corpus. boston.lti.cs.cmu.edu/Data/clueweb09/
- [27] E. F. Codd: A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), pages 377–387, 1970.
- [28] J. R. Cowie: Automatic Analysis Of Descriptive Texts. In *Proceedings of the 1st Applied Natural Language Processing Conference (ANLP)*, pages, 117–123, Santa Monica, California, USA, 1983.
- [29] S. Cucerzan: Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, 2007.
- [30] A. Das Sarma, A. Jain, C. Yu: Dynamic Relationship and Event Discovery. In *Proceedings of the*

- Forth International Conference on Web Search and Web Data Mining (WSDM)*, pages 207–216, Hong Kong, China, 2011.
- [31] J. Dean, S. Ghemawat: MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)* pages 137–150, San Francisco, California, USA, 2004.
- [32] G. F. DeJong: An Overview of the FRUMP System. *Strategies for Natural Language Processing*, pages 149-176, 1982.
- [33] P. DeRose, W. Shen, F. Chen, Y. Lee, D. Burdick, A. Doan, R. Ramakrishnan: DBlife: A Community Information Management Platform for the Database Research Community. In *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, 2007.
- [34] P. DeRose, W. Shen, F. Chen, A. Doan, R. Ramakrishnan: Building Structured Web Community Portals: A Top-down, Compositional, and Incremental Approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 399 –410, Vienna, Austria, 2007.
- [35] N.N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, S. Merugu: A Web of Concepts. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Dystems (PODS)*, pages 1–12, Providence, Rhode Island, USA, 2009.
- [36] A. Doan, L. Gravano, R. Ramakrishnan, S. Vaithyanathan. (Eds.): Special Issue on Information Extraction. *SIGMOD Record*, 37(4), 2008.
- [37] P. Domingos, D. Lowd: *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool, 2009.
- [38] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates: Unsupervised Named Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165(1), pages 91–134, 2005.
- [39] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates: Web-scale Information Extraction in KnowItAll. In *Proceedings of the 13th International Conference on World Wide Web (WWW)*, pages 100-110, New York, NY, USA, 2004.
- [40] O. Etzioni, M. Banko, S. Soderland, D.S. Weld: Open Information Extraction from the Web. *Communications of the ACM* 51(12), pages 68–74, 2008.
- [41] A. Fader, S. Soderland, O. Etzioni: Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, Edinburgh, UK, 2011.
- [42] L. Fang, A. Das Sarma, C. Yu, P. Bohannon: REX: Explaining Relationships between Entity Pairs. In *Proceedings of the VLDB Endowment PVLDB* 5(3), pages 241–252, 2011.
- [43] C. Fellbaum (Editor): *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [44] D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, D. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefel, C. A. Welty: Building Watson: An Overview of the DeepQA Project. *AI Magazine* 31(3), ppages 59–79, 2010.
- [45] J.R. Finkel, T. Grenager, C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, Ann Arbor, Michigan, 2005.

- [46] R. Gaizauskas, Y Wilks: Information Extraction: Beyond Document Retrieval. *Computational Linguistics and Chinese Language Processing* 3(2) 1998.
- [47] R. Grishman: Information Extraction: Techniques and Challenges: Lecture Notes in Computer Science, Springer, Vol. 1299, p. 10–27, 1997.
- [48] T. R. Gruber: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* 5(2), pp. 199–220, 1993.
- [49] R. Gupta, S. Sarawagi: Joint Training for Open-domain Extraction on the Web: Exploiting Overlap when Supervision is Limited. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM)*, pages 217–226, Hong Kong, China, 2011.
- [50] J. Han, J. Pei , Y. Yin : Mining frequent patterns without candidate generation. In *Proceedings of 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1–12, Dallas, Texas, USA, 2000.
- [51] X. Han, J. Zhao: Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. In *Proceedings of 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 215 –224,Hong Kong, China, 2009.
- [52] C. Hashimoto, K. Torisawa, K. Kuroda, S. D. Saeger, M. Murata, J. Kazama: Large-Scale Verb Entailment Acquisition from the Web. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1172–1181, Singapore, 2009.
- [53] C. Havasi, R. Speer, J. Alonso. ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge. In *Proceedings of the Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria, 2007.
- [54] T. H. Haveliwala: Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web (WWW)*, pages 517–526, Honolulu, Hawaii, USA, 2002.
- [55] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. O. Seaghdha, S. Pado, M. Pennacchiotti, L. Romano, S. Szpakowicz: SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals, 5th ACL International Workshop on Semantic Evaluation, 2010.
- [56] M. A. Hearst: Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, Nantes, France, 1992.
- [57] J. Hoffart, M. A. Yosef, I. Bordino and H. Fuerstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, Gerhard Weikum: Robust Disambiguation of Named Entities in Text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 782–792, Edinburgh, UK, 2011.
- [58] J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, G. Weikum: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 229–232, Hyderabad, India. 2011.
- [59] R. Hoffmann, C. Zhang, D. S. Weld: Learning 5000 Relational Extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 286–295, Uppsala, Sweden, 2010.
- [60] V. Kann: On the Approximability of NP-complete Optimization Problems. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm. 1992.
- [61] K. Kipper, A. Korhonen, N. Ryant, M. Palmer, A Large-scale Classification of English Verbs. *Language Resources and Evaluation Journal* 42(1): 21-40, 2008.

- [62] D.R. Karger, C. Stein: A New Approach to the Minimum Cut Problem. *Journal of ACM* 43(4), 1996.
- [63] G. Karypis, V. Kumar: A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering. *Journal of Parallel Distributed Computing* 48(1), 1998.
- [64] Z. Kozareva, E. H. Hovy: Learning Arguments and Supertypes of Semantic Relations Using Recursive Patterns. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1482–1491, Uppsala, Sweden, 2010.
- [65] Z. Kozareva, K. Voevodski, S. Teng. Class Label Enhancement via Related Instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 118–128, Edinburgh, UK, 2011.
- [66] A. Jain, P. G. Ipeirotis, A. Doan, L. Gravano: Join Optimization of Information Extraction Output: Quality Matters. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, pages 186–197, Shanghai, China, 2009.
- [67] J. R. Landis, G. G. Koch: The measurement of observer agreement for categorical data in Biometrics. Vol. 33, pp. 159–174, 1977.
- [68] C. Lee, Y-G. u Hwang, M.-G. Jang: Fine-grained Named Entity Recognition and Relation Extraction for Question Answering. In *Proceedings of the Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 799–800, Amsterdam, The Netherlands, 2007.
- [69] D. Lenat, R. Guha: Cyc: A Midterm Report. *AI Magazine* 11(3), pp. 32–59, 1990.
- [70] T. Lin, O. Etzioni, J.Fogarty: Identifying Interesting Assertions from the Web. In *Proceedings of 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1787–1790, Hong Kong, China, 2009.
- [71] D. Lin, P. Pantel: DIRT: Discovery of Inference Rules from Text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 323–328, San Francisco, California, USA, 2001.
- [72] T. Lin, Mausam , O. Etzioni: No Noun Phrase Left Behind: Detecting and Typing Unlinkable Entities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 893–903, Jeju, South Korea, 2012.
- [73] Xiao Ling, Daniel S. Weld: Fine-Grained Entity Recognition. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2012
- [74] G. Limaye, S. Sarawagi, S. Chakrabarti: Annotating and Searching Web Tables Using Entities, Types and Relationships. In *Proceedings of the VLDB Endowment PVLDB* 3(1), 2010
- [75] M-C. de Marneffe, B. MacCartney and C. D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation(LREC)*, Genoa, Italy, 2006.
- [76] D. N. Milne, I. H. Witten: Learning to Link with Wikipedia. In *Proceedings of 17th ACM Conference on Information and Knowledge Management (CIKM)*, pages 509-518, Napa Valley, California, USA, 2008.
- [77] T. Mohamed, E.R. Hruschka, T.M. Mitchell: Discovering Relations between Noun Categories. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1447-1455, Edinburgh, UK, 2011.
- [78] Sixth Message Understanding Conference (MUC-6), Morgan Kaufmann, 1995.
- [79] N. Nakashole, M. Theobald, G. Weikum: Find your Advisor: Robust Knowledge Gathering from the Web. WebDB, 2010.

- [80] N. Nakashole, M. Theobald, G. Weikum: Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM)*, pages 227–326, Hong Kong, China, 2011.
- [81] N. Nakashole, G. Weikum, F. Suchanek: PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1135 -1145, Jeju, South Korea, 2012.
- [82] N. Nakashole, G. Weikum, F. Suchanek: Discovering and Exploring Relations on the Web. In *Proceedings of the VLDB Endowment*, PVLDB 5(10), pages 1982–1985, 2012.
- [83] N. Nakashole, G. Weikum: Real-time Population of Knowledge Bases: Opportunities and Challenges, AKBC-WEKEX, 2012.
- [84] Preslav Nakov, Marti A. Hearst: Solving Relational Similarity Problems Using the Web as a Corpus. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 452–460, Columbus, Ohio, USA, 2008.
- [85] Roberto Navigli, Simone Paolo Ponzetto: BabelNet: Building a Very Large Multilingual Semantic Network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 216–225, Uppsala, Sweden, 2010.
- [86] V. Nastase, M. Strube, B. Boerschinger, Căcilia Zirn, Anas Elghafari: WikiNet: A Very Large Scale Multi-Lingual Concept Network. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*, Malta, 2010.
- [87] M. Nickel, V. Tresp, H.-P. Kriegel. Factorizing YAGO: Scalable Machine Learning for Linked Data. In *Proceedings of the 21st World Wide Web Conference (WWW)*, pages 271–280, Lyon, France, 2012.
- [88] H. T. Nguyen, T. H. Cao: Named Entity Disambiguation on an Ontology Enriched by Wikipedia. In *Proceedings of the IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies (RIVF)*, pages 247–254, Ho Chi Minh City, Vietnam, 2008.
- [89] A. Okumura, E. Hovy: Building Japanese-English dictionary based on ontology for machine translation. In *Proceedings of the Human Language Technology Workshop (HLT)*, pages 141-146, Plainsboro, New Jersey, USA, 1994.
- [90] P. Sadalage, M. Fowler. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Addison-Wesley. 2012.
- [91] M. Pasca, B. Van Durme: What You Seek Is What You Get: Extraction of Class Attributes from Query Logs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2832–2837, Hyderabad, India, 2007.
- [92] M. Pasca, B. Van Durme: Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 19-27, Columbus, Ohio, USA, 2008.
- [93] A. Philpot, E. Hovy, P. Pantel: The Omega Ontology, in: *Ontology and the Lexicon*. Cambridge University Press 2008.
- [94] S. P. Ponzetto, M. Strube: Deriving a Large-Scale Taxonomy from Wikipedia. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1440–1445, Vancouver, British Columbia, Canada, 2007.
- [95] R. Srikant, R. Agrawal: Mining Sequential Patterns: Generalizations and Performance Im-

- provements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT)*, pages 3–17, Avignon, France, 1996.
- [96] A. Rahman, Vincent Ng: Inducing Fine-Grained Semantic Classes via Hierarchical and Collective Classification. In *Proceedings the International Conference on Computational Linguistics (COLING)*, pages 931-939, 2010.
- [97] J. Reisinger, M. Pasca: Latent Variable Models of Concept-Attribute Attachment. In *Proceedings of the 7th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language (ACL/AFNLP)*, pages 620–628, Singapore, 2009.
- [98] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, S. Vaithyanathan: An Algebraic Approach to Rule-based Information Extraction. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 933–942, Cancun, Mexico, 2008.
- [99] M. Richardson, P. Domingos: Markov Logic Networks. *Machine Learning* 62(1-2), pages 107–136 2006.
- [100] S. Riedel, L. Yao, A. McCallum. Modeling Relations and their Mentions without Labeled Text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 148-163, Barcelona, Spain, 2010.
- [101] A. Ritter, Mausam, O. Etzioni, S. Clark: Open Domain Event Extraction from Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1104-1112, Beijing, China, 2012.
- [102] J. Rusher, Semantic Web Advanced Development for Europe (SWAD-Europe), Workshop on Semantic Web Storage and Retrieval - Position Papers, 2004.
- [103] T. Sakaki, M. Okazaki, Y. Matsuo: Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 851-860, Raleigh, North Carolina, USA, 2010.
- [104] S. Sarawagi: Information Extraction. *Foundations and Trends in Databases* 1(3), 2008.
- [105] N. Shadbolt, W. Hall, T. Berners-Lee: The Semantic Web Revisited. *IEEE Intelligent Systems*, 2006.
- [106] R. Schank, R Abelson: Scripts, Plans, Goals and Understanding. *Lawrence Erlbaum Associates*, 1977.
- [107] F. M. Suchanek, G. Ifrim, G. Weikum: LEILA: Learning to Extract Information by Linguistic Analysis. In *2nd Workshop on Ontology Learning and Population*, 2006.
- [108] F. M. Suchanek, G. Kasneci, G. Weikum: Yago: a Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web (WWW)* pages, 697-706, Banff, Alberta, Canada, 2007.
- [109] F. M. Suchanek, M. Sozio, G. Weikum: SOFIE: A Self-organizing Framework for Information Extraction. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 631–640, Madrid, Spain, 2009.
- [110] B. M. Sundheim, N. A. Chinchor: Survey of the Message Understanding Conferences. In *Proceedings of the Human Language Technology Workshop (HLT)*, 1993. Tummarello10 G. Tummarello: Live Views on the Web of Data. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 1301-1304, Raleigh, North Carolina, USA, 2010.
- [111] A. Turing, Alan: Computing Machinery and Intelligence. *Mind* 59(236), pages 433–460, 1950.
- [112] P. Venetis, A. Halevy, J. Madhavan, M. Pasca, W. Shen, F. Wu, G. Miao, C. Wu: Recovering

- Semantics of Tables on the Web. In *Proceedings of the VLDB Endowment*, PVLDB 4(9), pages, 528–538. 2011.
- [113] Word Wide Web Consortium (W3C): RDF Primer. <http://www.w3.org/TR/rdf-primer/>
- [114] C. Wang, K. Chakrabarti, T. Cheng and S. Chaudhuri Targeted Disambiguation of Ad-hoc, Homogeneous Sets of Named Entities . In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 719-728, Lyon, France, 2012.
- [115] C.Wang, J. Fan, A. Kalyanpur, D. Gondek: Relation Extraction with Relation Topics. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 126-136, Edinburgh, UK, 2011.
- [116] G. Weikum, G. Kasneci, M. Ramanath, F.M. Suchanek: Database and Information Retrieval Methods for Knowledge Discovery. *Communications of ACM* 52(4), 2009
- [117] G. Weikum, M. Theobald: From Information to Knowledge: Harvesting Entities and Relationships from Web Sources. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 65–76, Indianapolis, Indiana, USA, 2010.
- [118] D. S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, M. Skinner. Intelligence in Wikipedia. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 1609–1614, Chicago, Illinois, 2008.
- [119] D. S. Weld, R. Hoffmann, F. Wu. Using Wikipedia to Bootstrap Open Information Extraction. *SIGMOD Record*, 37(4), pages 62–68, 2008.
- [120] M.L. Wick, A. Culotta, K. Rohanimanesh, A. McCallum: An Entity Based Model for Coreference Resolution. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pages 365–376, Sparks, Nevada, USA, 2009.
- [121] T. White: Hadoop: The Definitive Guide. *O'Reilly*, 2009.
- [122] F. Wu, D. S. Weld. Automatically Refining the Wikipedia Infobox Ontology. In *Proceedings of the 17th International Conference on World Wide Web (WWW)*, pages 635–644, Beijing, China, 2008.
- [123] Wentao Wu, Hongsong Li, Haixun Wang, Kenny Q. Zhu: Towards a Probabilistic Taxonomy of Many Concepts. Technical Report MSR-TR-2011-25, Microsoft Research, 2011.
- [124] W. Wu, H. Li, H. Wang, K. Zhu: Probbase: A Probabilistic Taxonomy for Text Understanding. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 481–492, Scottsdale, AZ, USA, 2012.
- [125] M. Yakout, K. Ganjam, K. Chakrabarti, S. Chaudhuri, InfoGather: Entity Augmentation and Attribute Discovery By Holistic Matching with Web Tables. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 97–108, Scottsdale, AZ, USA, 2012.
- [126] W. Shen, A. Doan, J. F. Naughton, R. Ramakrishnan: Declarative Information Extraction using Datalog with Embedded Extraction Predicates. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB)*, pages 1033-1044, Vienna, Austria, 2007.
- [127] Y. Shinyama, S. Sekine: Preemptive Information Extraction using Unrestricted Relation Discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, New York, NY, USA, 2006.
- [128] K. Shvachko, H. Kuang, S. Radia, R. Chansler: The Hadoop Distributed File System. In *Proceedings of the 26th IEEE Conference on Massive Data Storage (MSST)*, Incline Village, Nevada USA, 2010.

- [129] F. Xu, H. Uszkoreit, H. Li: A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In *Proceedings of the Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, 2007.
- [130] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, S. Soderland: TextRunner: Open Information Extraction on the Web. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, pages 25-26, Rochester, NY, USA, 2007.
- [131] L. Yao, A. Haghighi, S. Riedel, A. McCallum: Structured Relation Discovery using Generative Models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1456-1466, Edinburgh, UK, 2011.
- [132] M. A. Yosef, S. Bauer, J. Hoffart, M. Spaniol, G. Weikum: HYENA: Hierarchical Type Classification for Entity Names. In *Proceedings the International Conference on Computational Linguistics (COLING)*, to appear, 2012.
- [133] G. P. Zarri: Automatic Representation of the Semantic Relationships Corresponding To A French Surface Expression. In *Proceedings of the 1st Applied Natural Language Processing Conference (ANLP)*, pages, 143–147, Santa Monica, California, USA, 1983.
- [134] J. Zhu, Z. Nie, X. Liu, B. Zhang, J.-R. Wen. StatSnowball: a Statistical Approach to Extracting Entity Relationships. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 101–110, Madrid, Spain, 2009.