ENTITY INFORMATION EXTRACTION USING STRUCTURED AND SEMI-STRUCTURED RESOURCES

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by

Avirup Sil
May, 2014

Examining Committee Members:

Dr. Alexander Yates, Advisory Chair, Computer and Information Sciences
Dr. Zoran Obradovic, Computer and Information Sciences
Dr. Yuhong Guo, Computer and Information Sciences
Dr. Silviu-Petru Cucerzan, External Member, Microsoft Research

ABSTRACT

ENTITY INFORMATION EXTRACTION USING STRUCTURED AND SEMI-STRUCTURED RESOURCES

by

Avirup Sil

Among all the tasks that exist in Information Extraction, Entity Linking, also referred to as entity disambiguation or entity resolution, is a new and important problem which has recently caught the attention of a lot of researchers in the Natural Language Processing (NLP) community. The task involves linking/matching a textual mention of a named-entity (like a person or a movie-name) to an appropriate entry in a database (e.g. Wikipedia or IMDB). If the database does not contain the entity it should return NIL (out-of-database) value.

Existing techniques for linking named entities in text mostly focus on Wikipedia as a target catalog of entities. Yet for many types of entities, such as restaurants and cult movies, relational databases exist that contain far more extensive information than Wikipedia. In this dissertation, we introduce a new framework, called Open-Database Entity Linking (Open-DB EL), in which a system must be able to resolve named entities to symbols in an arbitrary database, without requiring labeled data for each new database. In experiments on two domains, our Open-DB EL strategies outperform a state-of-the-art Wikipedia EL system by over 25% in accuracy.

Existing approaches typically perform EL using a pipeline architecture: they use a Named-Entity Recognition (NER) system to find the boundaries of mentions in text, and an EL system to connect the mentions to entries in structured or semi-structured repositories like Wikipedia. However, the two tasks are tightly coupled, and each type of system can benefit significantly from the kind of information provided by the other. We propose and develop a joint model for NER and EL, called NereL, that

takes a large set of candidate mentions from typical NER systems and a large set of candidate entity links from EL systems, and ranks the candidate mention-entity pairs together to make joint predictions. In NER and EL experiments across three datasets, NereL significantly outperforms or comes close to the performance of two state-of-the-art NER systems, and it outperforms 6 competing EL systems. On the benchmark MSNBC dataset, NereL provides a 60% reduction in error over the next-best NER system and a 68% reduction in error over the next-best EL system.

We also extend the idea of using semi-structured resources to a relatively less explored area of entity information extraction. Most previous work on information extraction from text has focused on named-entity recognition, entity linking, and relation extraction. Much less attention has been paid to extracting the temporal scope for relations between named-entities; for example, the relation `president-Of`(John F. Kennedy, USA) is true only in the time-frame (January 20, 1961 - November 22, 1963). In this dissertation we present a system for temporal scoping of relational facts, called TsrF which is trained on distant supervision based on the largest semi-structured resource available: Wikipedia. TsrF employs language models consisting of patterns automatically bootstrapped from sentences collected from Wikipedia pages that contain the main entity of a page and slot-fillers extracted from the infobox tuples. This proposed system achieves state-of-the-art results on 6 out of 7 relations on the benchmark Text Analysis Conference (TAC) 2013 dataset for the task of temporal slot filling (TSF). Overall, the system outperforms the next best system that participated in the TAC evaluation by 10 points on the TAC-TSF evaluation metric.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation and thank my advisor, Alexander Yates, who has introduced me to the world of Information Extraction and Natural Language Processing (NLP). Alex has always been a great advisor. His huge knowledge and enthusiasm and attention to details were of great inspiration throughout my studies. Without his patience, guidance and valuable advice this dissertation would not have been possible. Alex also has a major influence in shaping my critical thinking in scientific research. He has always provided me with excellent comments and brainstormed with me about new ideas from our first to our last projects during our meetings.

My special thanks goes out to Silviu Cucerzan, my committee member. Silviu has been my mentor during my internship at Microsoft Research. Silviu is an extremely knowledgeable researcher and one of the greatest human beings I have ever met. He not only advised me about state-of-the-art NLP research but also introduced me to the world of research outside of academia. I cherish every moment that I have spent with him: either brainstorming about some cool problem or staying awake the whole night at MSR to submit our system results to TAC.

I would also like to thank Ana-Maria Popescu, Diane Jass Ketelhut and Catherine Schifter for collaborating with me on research projects. Diane and Catherine have introduced me to the domain of Education Data Mining and I learned a lot for the two years that we collaborated on various projects for SAVE Science.

I am really thankful to both professors Zoran Obradovic and Vasilis Megalooikonomou for introducing me to the topics of data mining and database management respectively. The ideas that I gained during our discussions and coursework have been used throughout in my PhD work. A special thanks goes to Yuhong Guo, for her insightful

comments and refreshing interests in my research. I would also like to thank all the faculty members at Temple University for being kind and supportive during my PhD.

Outside research and department, I want to thank my wife and my friend, Devika Sil. She has always been the pillar of strength throughout my entire PhD research and my stay away from my home in Kolkata, India. She has been there when I was extremely upset with a paper rejection to the time when I was excited to get my first citation. Thank you Devika for being there for me. I would also like to thank my sister Avantika Sarkar and her husband, Sudipta Sarkar for providing me with moral support when I needed them the most. My cousin Aniketa Sarkar deserves special mention as she has been extremely helpful and cheerful during all the situations. Finally, I want to thank my parents, Jaya and Sunil Sil who have made me the person I am today; their love and their sacrifices have given me more than any son can ever ask for. My biggest dream ever in my life is to be a successful human-being like my parents.

To my family.

This dissertation is a result of their continuous love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

Information Extraction (IE) is the task of automatically extracting structured information such as entities and the relationships that exist between entities from unstructured text. The resulting structured information is often known as a Knowledge Base (KB). This task has tremendous application to the next generation of search which is moving away from keyword-based search to entity-based search. IE allows us to have complex queries over the huge collection of unstructured resources than just simple keyword-based traditional searches. It allows us to integrate both structured and semi-structured data that exists for free in the Web and also query them.

The field of IE spans the domains of machine learning, databases, information retrieval and has its roots in the topic of Natural Language Processing (NLP). Most IE research focuses on the task of Named-Entity Recognition (NER) and Relation Extraction. NER deals with extracting named-entities *e.g.* people, location and organization names from text. Relation Extraction involves extracting (binary) relational facts between named-entities *e.g.* born-In(Albert Einstein, Ulm) from plain text. The existence of three programs focused on these tasks: Message Understanding Conference (MUC) (*Sundheim* (1991); *Grishman and Sundheim* (1996); *Chinchor* (1998)) and Automatic Content Extraction (ACE) (*Doddington et al.* (2004)) and CoNLL-shared task (*Tjong Kim Sang and De Meulder* (2003)). The Entity Linking (EL) (*Bunescu and Pasca* (2006); *Cucerzan* (2007); *Mihalcea and Csomai* (2007); *Kulkarni et al.* (2009); *Ratinov et al.* (2011)) task goes one step further than NER by disambiguating entities to the largest available semi-structured resource, Wikipedia.

However, for data as diverse as the web, which consists of blogs, tweets, etc., contains varied types of entities, such as restaurants and cult movies which do not have any target entry in Wikipedia. In this dissertation, we first investigate techniques in which a system must be able to resolve named-entities to symbols in an arbitrary database, without requiring labeled data for each new database. We also investigate joint NER and EL techniques using both structured and semi-structured resources to predict entities in documents which boosts performance of both the tasks. Finally, we will show how to attach temporal information to relations between entities, a task which involves information extraction techniques using a semi-structured resource.

1.1 Tasks

1.1.1 Entity Linking

Entity Linking (EL) is the task of linking names mentioned in text with an established catalog of entities. Most previous work focuses on linking entities to their corresponding Wikipedia targets. Figure 1.1 shows an example of an EL system in action. Given some plain text, EL systems first extracts mentions of entities *e.g.* Tennessee, Philadelphia and Chris Johnson in this example text. Most systems use machine learning techniques which make use of features like cosine-similarity between the Wikipedia articles of the targets like Tennessee Titans (in this example) with the given source input text. Names of entities like Tennessee and Philadelphia have been linked to their corresponding targets in Wikipedia which are Tennessee Titans and Philadelphia Eagles, both football teams.

1.1.2 Temporal Scoping of Relational Facts

Relation extraction is the task of automatically extracting relational facts like (Jennifer Aniston, spouse-Of, Brad Pitt) from text *e.g.* " *Jennifer Aniston married*

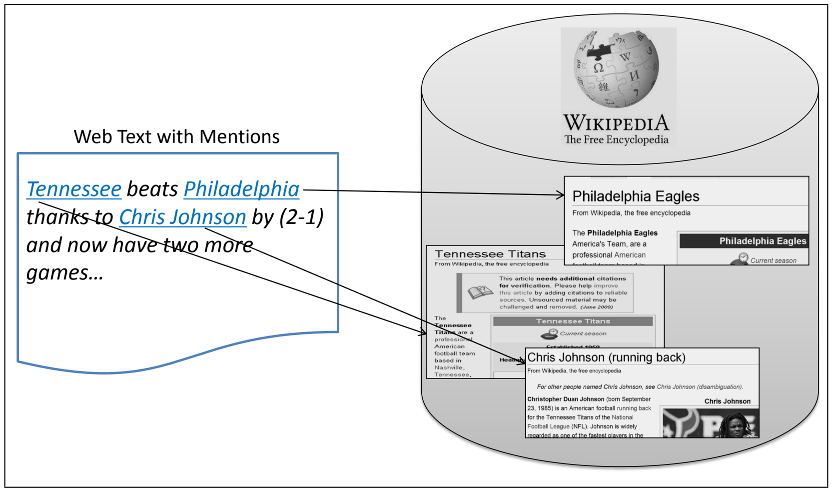

Figure 1.1: Example of Entity Linking.

Brad Pitt on July 29, 2000..". These facts are filled in a large repository called Knowledge Base (KB) which can be further used by other applications for tasks like question answering. Temporal Scoping of relational facts involve automatic extraction of temporal information about entities from text. Also, it involves classification of the temporal information into start, end and in (for entities still in a relationship) labels. Hence, in this example, a system should extract *July 29, 2000* as the start of the relation and from the text "*..Pitt divorced Aniston on 2005*" should extract *2005* as the end timestamp of the relation.

1.2 Problems and Limitations of current Entity Linking

1.2.1 Restriction of using only Wikipedia as a repository

EL requires a catalog of symbols, called referents, to which named-entities will be resolved. Most EL systems today use Wikipedia as the catalog of referents, but exclusive focus on Wikipedia as a target for EL systems has significant drawbacks: despite its breadth, Wikipedia still does not contain all or even most real-world entities mentioned in text. A widely noticed example is that it has poor coverage of entities that are mostly important in a small geographical region, such as hotels and restaurants, which are frequently discussed on the Web. Another example is the cult movie names which are present in review pages, tweets and blogs. However, these entities clearly exist in structured relational databases like Yelp or IMDB. The negative effects and sparsity of entities in Wikipedia is well documented in the EL literature. For example, *Lin et al.* (2012b) show that 33% of mentions in a corpus of 500 million Web documents cannot be linked to Wikipedia. Similarly, *McNamee et al.* (2009) illustrate that 57% of the named-entities in the Text Analysis Conference's (TAC) 2009 entity linking task refer to an entity that does not appear in Wikipedia.

Apart from the issue of coverage, a second problem is that identifying the essential attributes of an entity that help disambiguate it using the text in a Wikipedia article can be difficult, especially for shorter and less-structured documents like tweets or blogs. This is because in shorter documents contextual information about an entity is minimal. Wikipedia is obviously a highly valuable resource, but it should not be thought of as the only one.

In order to solve these problems, instead of using only Wikipedia as the target, we propose a novel approach to EL, which we refer to as the Open-Database Entity Linking or, in short, *Open-DB* EL (*Sil et al.* (2012a)). Here, the task is to resolve an entity to Wikipedia or to *any* relational database that meets mild conditions about the format of the data. As per EL literature, using structured data allows systems to attain high accuracy. For instance, Hoffart *et al.*'s EL system for YAGO (*Hoffart et al.* (2011)) uses domain-specific or database-specific EL techniques to outperform several state-of-the-art systems like *Cucerzan* (2007) and *Kulkarni et al.* (2009). We also know that there exists a huge numbers of databases on the Web and many of those are for specialized domains. Hence, a successful EL system for this task will cover entities that a Wikipedia EL or database-specific system cannot.

We explore two complementary learning strategies for Open-DB EL, both of which significantly relax the assumptions of traditional EL systems. The first strategy, a distant supervision approach, uses the relational information in a given database and a large corpus of unlabeled text to learn a database-specific model. This reduces the task of manual labeling which is an expensive effort. The second strategy, a domain adaptation approach, assumes a single source database that has accompanying labeled data. Classifiers in this setting must learn a model that transfers from the source database to any new database, without requiring new training data for the new database. In this scenario, once we have a trained EL model on some domain *e.g.* movies, we can use the model for any other database *e.g.* sports, without the need for labeled data for the new domain. Our experiments show that both strategies outperform a state-of-the-art Wikipedia EL system used in production by a major commercial search engine by wide margins without requiring any labeled data from the test domain, highlighting the significant advantage of having domain-specific relational data.

1.2.2 Limitations of Pipeline Entity Linking

Most previous EL systems depend on knowing where the set of names of entities in a text appear. Hence, almost all EL systems use a pipeline architecture. At first, they run a Named-Entity Recognition (NER) (*Finkel et al.* (2005); *Ratinov and Roth* (2009)) system on the text to identify the boundaries of names. Then, they use the names that were identified by the NER system, typically called *mentions* or surface forms, as the input set of names for the EL task. However, pipeline architectures for probabilistic models often come with a serious problem. Errors from the NER system propagate to the EL system. Any mention that goes undetected by the NER system clearly cannot be linked correctly by the downstream EL system. Other common errors from NER systems include incorrectly splitting a multi-word name into multiple mentions *e.g.* the mention "Blue Cross and Blue Shield of Alabama" is split into "Blue Cross" as one mention and "Blue Shield of Alabama" as a separate mention. This type of an error is often called over-segmentation error. Another class of error is the under-segmentation error which leads an EL system to generate fewer links and affects the recall of a system. Here, the NER system mistakenly joins together two mentions that happen to appear near one another *e.g.* "Orange County California" is detected as one mention whereas the correct one should have been "Orange County" as one mention and "California" as a separate mention. Finally, there exists mis-segmentation issues where NER systems leave off one or more words from a multi-word mention. Each of these errors can cause the downstream EL system to generate too many, too few, or just wrong links.

In response to the problems that exist in the traditional EL architecture, we propose and develop a joint NER and EL model, called NEREL (*Sil and Yates* (2013); *Sil* (2013)), that we use to re-rank candidate mentions and entity links produced by base NER and EL models. The intuition that we want to capture is how can both these task help each other. Our base NER techniques for example, would produce large number of candidate mentions from the phrases in an input text. Our base EL system would then propose several promising entity links for each candidate mention. The re-ranking model, NEREL then chooses among the set of all possible mention and entity link labelings for the whole phrase to determine the best choice. Since the re-ranking model considers labels for several mentions simultaneously, it can use features for known relationships between the entity links that exists between them. For efficiency, such features are not available to the pipeline models, which consider only one mention at a time. We use the pipeline models to prune the set of all possible candidate mentions and entity links to a manageable size while maintaining high recall. The re-ranking model can then use more sophisticated features for collective classification over this pruned set. Empirically, using this architecture, our proposed system significantly outperforms or comes close to the performance of two state-of-the-art NER systems, and it outperforms 6 competing EL systems. For example on the benchmark MSNBC dataset, NereL provides a 60% reduction in error over the next-best NER system and a 68% reduction in error over the next-best EL system.

1.3 Limitations of Entity Relation Extraction

Traditional information extraction research mostly focus on named-entity recognition (NER) and relation extraction. In relation extraction, the task is to build a large knowledge base with relational facts between entities. Most previous work (*Agichtein and Gravano* (2000); *Etzioni et al.* (2004)) on relation extraction by systems such as NeLL (*Carlson et al.* (2010)), KnowItAll (*Etzioni et al.* (2004)) and YAGO (*Suchanek et al.* (2007)) have targeted the extraction of entity tuples (*e.g.* CEO-Of(Steve Ballmer, Microsoft)) in order to build a large knowledge base of facts. All these systems assume that relational facts are time-invariant: something that is certainly not true. For example, the fact CEO-Of(Steve Ballmer, Microsoft) is true only between the time-frame (2000-2014). We argue that instead of just relational facts, storing their temporal scope can be important for a variety of reasons. Relational facts can be ranked and stored by virtue of their temporal information. As per *Alonso et al.* (2007), entity relations can be presented in a timeline which can provide a great user experience in a search engine or another user-interface. We have already mentioned that the next generation of search is based on entities. Without temporal information, current users would not be able to ask questions to search engine like "When did Carol Bartz become the CEO of Yahoo?" or "When was the iPhone invented?" Notice that both these question require temporal scoping of the `CEO-Of` and `invented` relation. Most state-of-the-art summarization techniques also require events to be ordered in the rank of their sentences. Without temporal ordering the sentences loose most of their semantics.

In this dissertation, we focus on this relatively less explored problem: attaching temporal scope to relation between entities. Recently the Text Analysis Conference (TAC) 2013 has also concentrated on this problem and has introduced the temporal slot filling (TSF) task (*Dang and Surdeanu* (2013)). The input to a TAC-TSF system is a binary relation *e.g.* `per:spouse`(Brad Pitt, Jennifer Aniston) and a document supporting the relation. The output of the system is a 4-tuple timestamp [T1, T2, T3, T4] where T1 and T2 are normalized dates that provide a range for the start date of the relation, and T3 and T4 provide the range for the end of the relationship. A TAC-TSF system must also output the offsets of the text that supports the temporal information extracted within a document, *e.g.* *"Pitt married Jennifer Aniston on July 29, 2000... the couple divorced five years later in 2005."* The 4-tuple normalized timestamp extracted from this example text is [2000-07-29, 2000-07-29, 2005-01-01, 2005-12-31].

We present TSRF (*Sil and Cucerzan* (2014);*Cucerzan and Sil* (2013)), a system for temporal scoping of relational facts. Since manual labeling of temporal expressions is expensive, TSRF is automatically trained using Wikipedia data. For every relation type, our approach uses distant supervision from Wikipedia infobox tuples, to learn a language model consisting of patterns from entity types and n-grams from Wikipedia sentences. Then it uses this trained relation-specific language model to extract the top-k sentences that support the given relation between the query entity and the slot filler. In a second stage, TSRF applies timestamp classification techniques that employ models which learn "Start", "End" and "In" predictors of entities in a relationship to automatically extract the best 4-tuple timestamp based on confidence values from the top sentences in the document. This proposed system achieves state-of-the-art results on 6 out of 7 relations on the benchmark Text Analysis Conference (TAC) 2013 dataset for the task of temporal slot filling (TSF). Overall, the system outperforms the next best system that participated in the TAC evaluation by 10 points on the TAC-TSF evaluation metric.

CHAPTER 2

LINKING NAMED ENTITIES TO ANY DATABASE

2.1 Introduction

Entity Linking (EL) is the task of linking names mentioned in text with an established catalog of entities (*Bunescu and Pasca* (2006); *Ratinov et al.* (2011)). It is a vital first step for semantic understanding of text, such as in grounded semantic parsing (*Kwiatkowski et al.* (2011)), as well as for information retrieval tasks like person name search (*Chen and Martin* (2007); *Mann and Yarowsky* (2003)).

EL requires a catalog of symbols, called referents, to which named-entities will be resolved. Most EL systems today use Wikipedia as the catalog of referents, but exclusive focus on Wikipedia as a target for EL systems has significant drawbacks: despite its breadth, Wikipedia still does not contain all or even most real-world entities mentioned in text. As one example, it has poor coverage of entities that are mostly important in a small geographical region, such as hotels and restaurants, which are widely discussed on the Web. 57% of the named-entities in the Text Analysis Conference's (TAC) 2009 entity linking task refer to an entity that does not appear in Wikipedia (*McNamee et al.* (2009)). As another example, we found just 16% of 156 cult movie titles we collected from the Web were covered in Wikipedia. In addition to coverage, a second problem is that identifying the essential attributes of an entity that help disambiguate it using the text in a Wikipedia article can be difficult, especially for shorter and less-polished articles. Wikipedia is clearly a highly valuable resource, but it should not be thought of as the only one.

Instead of relying solely on Wikipedia, we propose a novel approach to EL, which we refer to as *Open-DB* EL: the task is to resolve an entity to Wikipedia or to *any* relational database that meets mild conditions about the format of the data, described below. Leveraging structured, relational data should allow systems to achieve strong accuracy, as with domain-specific or database-specific EL techniques like Hoffart *et al.*'s EL system for YAGO (*Hoffart et al.* (2011)). And because of the availability of huge numbers of databases on the Web, many for specialized domains, a successful system for this task will cover entities that a Wikipedia EL or database-specific system cannot.

The Open-DB EL task comes with significant new challenges, in addition to its benefits. Existing EL systems make strong assumptions about the format and contents of the database being resolved to, such as the existence of disambiguation links or infoboxes in Wikipedia, or the existence of particular attributes and relations in a particular database. Existing systems often also assume the existence of manually-annotated training data with labels from the designated database. In the Open-DB case, both assumptions are overly restrictive.

One important reason for the dominance of Wikipedia in EL is that it provides human-annotated data for a large number of domains and entities, via its internal link structure. Systems trained on Wikipedia data can be accurate for actors' names, politicians, and sports figures, without the need for specialized techniques for each domain. To avoid losing the generality provided by Wikipedia, we restrict our attention to systems that can resolve to new databases, without the need for additional human-annotated training data for each new database. Thus like Wikipedia EL systems, the resulting systems will resolve to ontologies from arbitrary domains with no extra human effort.

We investigate two complementary learning strategies for Open-DB EL, both of which significantly relax the assumptions of traditional EL systems. The first strategy, a distant supervision approach, uses the relational information in a given database and a large corpus of unlabeled text to learn a database-specific model. The second strategy, a domain adaptation approach, assumes a single source database that has accompanying labeled data. Classifiers in this setting must learn a model that transfers from the source database to any new database, without requiring new training data for the new database. Experiments show that both strategies outperform a state-of-the-art Wikipedia EL system by wide margins without requiring any labeled data from the test domain, highlighting the significant advantage of having domain-specific relational data.

The next section contrasts Open-DB EL with previous work. Section 2.3 formalizes the task. Sections 2.4 and 2.5 present our distant supervision strategy and domain-adaptation strategy, respectively. Section 2.6 introduces a technique that is a hybrid of the two learning strategies. Section 3.6 describes our experiments, and Section 3.7 concludes.

2.2 Previous Work

The earliest EL research works were authored by *Bunescu and Pasca* (2006) and *Cucerzan* (2007). Both these systems link mentions of entities to their corresponding targets in Wikipedia but there exists several differences in their approaches. Cucerzan uses Wikipedia disambiguation markup and several heuristics to extract mappings from surface forms of entities to their Wikipedia targets. For every entity in Wikipedia, a context vector for the entity is developed. In the next phase, using dot-product these vectors are compared with the context vectors of unknown entity surface forms. However, the major assumption in this work is that all entities have a corresponding Wikipedia target. This hypothesis suffers as per *McNamee et al.* (2009) for a significant number of entities in heterogeneous genres like news articles or blogs. *Bunescu and Pasca* (2006) learn a threshold to predict if a certain entity is not in Wikipedia. Overall, both systems heavily rely on this semi-structured resource: they use Wikipedia-based annotations *e.g.* category hierarchies and disambiguation links. *Cucerzan* (2007) uses a global disambiguation model, a framework which seems to provide little improvement over a local approach as shown by *Ratinov et al.* (2011). It also uses web search query logs to detect mentions of entities in documents which is a kind of resource which is certain powerful but not easily available for academic research. Another system which uses search engine query logs for mining features and disambiguation is the *Zhou et al.* (2010) wikifier[1]. Empirically this system has outperformed *Cucerzan* (2007) on web data *e.g.* Yahoo! News and MSNBC. Zhou *et al.* employs learning-to-rank techniques which accounts for the relative rank between the candidate entities. However, all these 3 systems rely on Wikipedia as the sole entity linking resource. Our experiments show that Open-DB EL approaches outperform *Zhou et al.* (2010) by a wide margin on web data like Yahoo! news showing the advantages of having relational data in addition to Wikipedia.

Another closely related topic of research deals with Person Name disambiguation. The main task here is to resolve ambiguity in personal name search. One of the most influential works in this domain is by *Mann and Yarowsky* (2003). Information extraction mining biographic data *e.g.* occupation, birth year and affiliation seems to provide features to these systems to disambiguate person names in the web. Using regular expressions help extracting biographic facts which further help in disambiguation. There exists the Web People Search Task (*Artiles et al.* (2009)) which deals with clustering of web pages for entity disambiguation. Our work deals with EL research which is more diverse as the task is to dismbiguate not just person names but also other entity mentions like location names *e.g.* Philadelphia which can refer to the city of Philadelphia and also to Philadelphia Phillies (the baseball team) *etc.*.

[1] The EL systems which link targets to Wikipedia are often known as wikifiers.

As mentioned above, restricting the catalog of referents to Wikipedia, as most recent EL systems do (*Bunescu and Pasca* (2006); *Mihalcea and Csomai* (2007); *Fader et al.* (2009); *Han and Zhao* (2009); *Kulkarni et al.* (2009); *Ratinov et al.* (2011)), can restrict the coverage of the system. *Zhou et al.* (2010) estimate that 23% of names in Yahoo! news articles have no referent in Wikipedia, and *Cucerzan* (2007) estimates the rate at 16% in MSNBC news articles. There is reason to suspect that these estimates are on the low side, however, as news tends to cover popular entities, which are most likely to appear in Wikipedia; the mentions in TAC's 2009 entity linking task are drawn from both newswire and blogs, and have a far higher rate (57%) of missing Wikipedia entries. *Lin et al.* (2012b) find that 33% of mentions in a corpus of 500 million Web documents cannot be linked to Wikipedia.

EL systems that are focused on specific domains (or verticals) greatly benefit from repositories of domain-specific knowledge, only a subset of which may be found in Wikipedia. For example, *Pantel and Fuxman* (2011) use a query-click graph to resolve names in search engine queries to a large product catalog from a commercial search engine, while *Dalvi et al.* (2009) and *Dalvi et al.* (2012) focus on movie and restaurant databases. *Bellare and McCallum* (2009) use the sequence information available in citation text to link author, title, and venue names to a publication database. The difference between our methods and *Bellare and McCallum* (2009) is that we do not rely on the domain-specific sequence/ordering information available in citation text. Open-DB EL systems work on any database, so they can serve as baselines for domain-specific EL tasks, as well as provide disambiguation for domains where no domain-specific EL system exists. Also, EL is a special case of Word Sense Disambiguation (WSD). The primary difference between us and the Lesk algorithm (*Lesk* (1986)) is that our databases have relational structure (c.f. Lesk's dictionary definitions) and we learn statistical models. In fact, our Open-DB EL techniques outperform a baseline model which uses the same intuition as the Lesk algorithm adapted to relational databases.

Numerous previous studies have considered distant or weak supervision from a single relational database as an alternative to manual supervision for information extraction (*Hoffmann et al.* (2011); *Weld et al.* (2009); *Bellare and McCallum* (2007); *Bunescu and Mooney* (2007); *Mintz et al.* (2009); *Riedel et al.* (2010); *Yao et al.* (2010)). In contrast to these systems, our distant supervision EL system provides a meta-algorithm for generating an EL system for *any* database and any entity type.

Existing domain adaptation or transfer learning approaches are inappropriate for the Open-DB EL task, either because they require labeled data in both the source and target domains (*Daumé III et al.* (2010); *Ben-David et al.* (2010)), or because they leverage some notion of distributional similarity between words in the source and target domains, which does not apply to the database symbols across the two domains. Instead, our domain adaptation technique uses *domain-independent* features of relational data, which apply regardless of the actual contents of the database, as explained further below.

2.3 The Open-DB EL Problem and Assumptions

2.3.1 Task Formulation

A *mention* is an occurrence of a named-entity in a document. Formally, a mention $m = (d, start, end)$ is a triple consisting of a document d, as well as a start and end position for the mention within the document. We say that d is the *context* of m. A *relational database* is a 2-tuple (S, R). Here, S is a set of symbols for constants, attributes, and relations in the database, and $R = \{r_1, \ldots, r_n\}$ is a set of relation instances of the form $r_i = \{(c_{1,1}, \ldots, c_{1,k_i}), \ldots, (c_{n_i,1}, \ldots, c_{n_i,k_i})\}$, where each c_j is taken from S, k_i is the *arity* of relation r_i and n_i is the number of known instances of r_i. We will write example database symbols in `teletype`, and mentions in "quotations."

For a particular database DB, we refer to its components as $DB.S$ and $DB.R$. For a set of databases D, define the set of referents as $S_D = (\bigcup_{DB \in D} DB.S) \cup \{OOD\}$, where OOD is a special symbol indicating something that is "out of database", or not found in any of the databases in D.

Given a corpus C, a set of mentions M that occur in C, and a set of databases D, the Open-DB EL task is to produce a function $f : M \rightarrow S_D$, which identifies an appropriate target symbol from one of the databases in D, or determines that the mention is OOD. Note that this problem formulation assumes no labeled data. This is significantly more challenging than traditional EL settings, but allows the system to generalize easily to any new database. In the domain adaptation section below, we relax this condition somewhat, to allow labeled data for a small number of initial databases; the system must then transfer what it learns from the labeled domains to any new database. Also note that the focus for this paper is disambiguation; we assume that the set of mentions are correctly demarcated in the input text. Previous systems, such as Lex (*Downey et al.* (2007)), have investigated the task of finding correct named-entity boundaries in text.

2.3.2 Assumptions

To allow our systems to handle arbitrary databases, we need to make some assumptions about a standard format for the data. We will assume that databases are provided in a particular form, called *Boyce-Codd Normal Form* (BCNF) (*Silberschatz et al.* (2010)). A relational schema is said to be in BCNF when all redundancy based on functional dependency has been removed, although other types of redundancy may still exist. Formally, a schema R is said to be in BCNF with respect to a set of functional dependencies $\mathcal{F}$ if for every one of the dependencies $(X \rightarrow Y) \in \mathcal{F}$, either

1. $Y \subset X$, meaning this is a trivial functional dependency, or

2. X is a *superkey*, meaning that X is a set of attributes that together define a

movie		
id	title	year
1	Next Door	1975
2	Next Door	2005
3	Next Door	2008
4	Next Door	2008
5	Next Door	2010
…	…	…

actor	
id	name
1	Nicole Kreux
2	Richard Ryan
3	Kristoffer Joner
4	Lee Perkins
5	Carla Valentine
…	…

acted_in		
movie_id	actor_id	role
5	1	Evelyn
5	2	Bruce
2	3	John
1	4	Kid
3	5	Elana
…	…	…

player			
id	name	height	position
1	Carlos Lee	6’2”	LF
2	Rob Bironas	6’0”	K
3	Chris Johnson	6’3”	3B
4	Chris Johnson	5’11”	RB
5	Chris Johnson	6’1”	DB
…	…	…	

team	
id	name
1	San Diego Padres
2	Houston Texans
3	Tennessee Titans
4	Oakland Raiders
5	Houston Astros
…	…

plays_for	
player_id	team_id
4	3
5	2
3	5
1	5
2	3
…	…

Figure 2.1: Example movie database (above) and sports database (below) in BCNF.

unique ID for the relation.

In practice, this is a relatively safe assumption as database designers often aim for even stricter normal forms. For databases not in BCNF, such as tables extracted from Web pages, standard algorithms exist for converting them into BCNF, given appropriate functional dependencies, although there are sets of functional dependencies for which BCNF is not achievable. Figure 2.1 shows two example databases in BCNF. We use these tables as examples throughout the paper.

We will additionally assume that all attributes, including names and nicknames, of entities that are covered by the database are treated as functional dependencies of the entity. Again, in practice, this is a fairly safe assumption as this is part of good database design, but if a database does not conform to this, then there will be some entities in the database that our algorithms cannot resolve to. This assumption implies that it is enough to use the set of superkeys for relations as the set of possible referents; our algorithms make use of this fact.

Our tests below focus on mentions that at least partially match some text field in a database, such as “Chris” and `Chris Johnson`. Disambiguating these mentions is already a significant challenge, and others have noted the difficulty of more diverse variations in names, such as nicknames (*Ratinov et al.* (2011)) and other rare surface forms; *Zhou et al.* (2010) use search engine query logs to help identify more variants, but such data is not readily available. We will assume the existence of a function $\mu(s, t)$ which indicates whether the text t is a valid surface form of database symbol s. Our experiments in Section 2.7.3 explore several possible simple definitions for this function.

2.4 A Distant Supervision Strategy for Open-DB EL

Our first approach to the Open-DB EL problem relies on the fact that, while many mentions are indeed ambiguous and difficult to resolve correctly, most mentions have only a very small number of possible referents in a given database. “Chris Johnson” is the name of doubtless thousands of people, but for articles that are reasonably well-aligned with our sports database, most of the time the name will refer to just three different people. Most sports names are in fact less ambiguous still. Thus, taking a corpus of unlabeled sports articles, we use the information in the database to provide (uncertain) labels, and then train a log-linear model from this probabilistically-labeled data.

This strategy requires a set of features for the model. Traditionally, such features would be hand-crafted for a particular domain and database. As a first step towards our Open-DB system, we present a log-linear model for disambiguation, as well as a simple feature-generation algorithm that produces a large set of useful features from a BCNF database. We then present a distant-supervision learning procedure for this model.

2.4.1 Disambiguation Model

Let S_D be the set of possible referents. We construct a vector of feature functions $\mathbf{f}(m, s)$ describing the degree to which m and $s \in S_D$ appear to match one another. The feature functions are described below. The model includes a vector of weights $\mathbf{w}$, one weight per feature function, and sets the probability of entity s given m and $\mathbf{w}$ as:

$$P(s|m, \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(m, s))}{\sum_{s' \in S_D} \exp(\mathbf{w} \cdot \mathbf{f}(m, s'))} \tag{2.4-1}$$

2.4.2 Database-driven Feature Generation

Figure 2.2 shows our algorithm for automatically generating feature functions $f_i(m, s)$ from a BCNF database. As mentioned above, we only need to consider resolving to database symbols s that are keys, or unique IDs, for some tuple in a database. For an entity in the database with key id, the feature generation algorithm generates two types of feature functions: attribute counts and similar entity counts. Each of these features measures the similarity between the information stored in the database about the entity id, and the information in the text in d surrounding mention m.

An *attribute count* feature function $f_{i,j}^{att}(m, id)$ for the jth attribute of relation r_i counts how many attributes of the entity id appear near m. For example, if id is `5` in the `movie` relation in Figure 2.1, the feature function for attribute `year` would count how often `2010` matches the text surrounding mention m. Defining precisely whether a database symbol "matches" a word or phrase is a subtle issue; we explore several possibilities in Section 2.7.3. In addition to attribute counts for attributes within a single relation, we also use attributes from relations that have been inner-joined on primary key and foreign key pairs. For example, for movies,

Algorithm: Feature Generation

Input: DB, a database in BCNF
Output: $\mathbf{F}$, a set of feature functions
Initialization: $\mathbf{F} \leftarrow \emptyset$

Attribute Count Feature Functions:

For each relation $r_i \in DB.R$
  For each j in $\{1, \ldots, k_i\}$
    Define function $f_{i,j}^{att}(m, id)$:
      $count \leftarrow 0$
      Identify the tuple $t \in r_i$ containing id
      $val \leftarrow t_j$
      $count \leftarrow count +$
          $ContextMatches(val, m)$
      return $count$
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{f_{i,j}^{att}\}$

Similar-Entity Count Feature Functions:

For each relation $r_i \in DB.R$
  For each j in $\{1, \ldots, k_i\}$
    Define function $f_{i,j}^{sim}(m, id)$:
      $count \leftarrow 0$
      Identify the tuple $t \in r_i$ containing id
      $val \leftarrow t_j$
      Identify the set of similar tuples T':
        $T' = \{t' | t' \in r_i, t'_j = val\}$
      For each tuple $t' \in T'$
        For each $j' \in \{1, \ldots, k_i\}$
          $val' \leftarrow t'_j$
          $count \leftarrow count +$
              $ContextMatches(val', m)$
      return $count$
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{f_{i,j}^{sim}\}$

Figure 2.2: Feature generation algorithm. The $ContextMatches(s, m)$ function counts how many times a string that matches database symbol s appears in the context of m. In our implementation, we use all of $d(m)$ as the context. Matching between strings and database symbols is discussed in Sec. 2.7.3.

we include attributes such as director name, genre, and actor name. High values for these attribute count features indicate that the text around m closely matches the information in the database about entity id, and therefore id is a strong candidate for the referent of m. We use the whole document as the context for finding matches, although other variants are worth future investigation.

A *similar entity count* feature function $f_{i,j}^{sim}(m, id)$ for the jth attribute in relation r_i counts how many entities similar to id are mentioned in the neighborhood of m. As an example, consider a mention of "Chris Johnson", $id = 3$, and the similar entity feature for the `position` attribute of the `players` relation in the sports database. The feature function would first identify that `3B` is the position of the player with $id = 3$. It would then identify all players that had the same position. Finally, it would count how often any attributes of this set of players appear near "Chris Johnson". Likewise, the similar entity feature for the `team id` attribute would count how many teammates of the player with $id = 3$ appear near "Chris Johnson". A high count for this teammate feature is a strong clue that id is the correct referent for m, while a high count for players of the same position is a weak but still valuable clue.

2.4.3 Parameter Estimation via Distant Supervision

Using string similarity, we can heuristically determine that three IDs with `name` attribute `Chris Johnson` are highly likely to be the correct target for a mention of "Chris Johnson". Our distant supervision parameter estimation strategy is to move as much probability mass as possible onto the set of realistic referents obtained via string similarity. Since our features rely on finding attributes and similar entities, the side effect of this strategy is that most of the probability mass for a particular mention is moved onto the one target ID with high attribute count and similar entity count features, thus disambiguating the entity. Although the string-similarity heuristic is typically noisy, the strong information in the database and the fact that many entity mentions are typically not ambiguous allows the technique to learn effectively from unlabeled text.

Let $\phi(m, DB)$ be a heuristic string-matching function that returns a set of plausible ID values in database DB for mention m. The objective function for this training procedure is a modified marginal log likelihood (MLL) function that encourages probability mass to be placed on the heuristically-matched targets:

$$MLL(M, \mathbf{w}) = \sum_{m \in M} \log \sum_{id \in \phi(m,DB)} P(id|m, \mathbf{w})$$

This objective is smooth but non-convex. We use a gradient-based optimization procedure that finds a local maximum. Our implementation uses an open-source version of the LBFG-S optimization technique (*Liu and Nocedal* (1989)). The gradient of our objective is given by

$$\frac{\partial LL(M, \mathbf{w})}{\partial w_i} = \sum_{m \in M} \mathbf{E}_{id \in \phi(m,DB)} \left[f_i(m, id) \right] - \mathbf{E}_{id \in DB.S} \left[f_i(m, id) \right]$$

where the expectations are taken according to $P(id|m, \mathbf{w})$.

2.5 A Domain-Adaptation Strategy for Open-DB EL

Our domain-adaptation strategy builds an Open-DB EL system by training it on labeled examples from an initial database or small set of initial databases. Unlike traditional EL, however, the purpose in Open-DB EL is to resolve to any database. Thus the strategy must take care to build a model that can transfer what it has learned to a new database, without requiring additional labeled data for the new database.

At first, the problem seems intractable — just because a system can disambiguate between "Next Door", the 2005 Norwegian film, and "Next Door", the 1975 short film by director Andrew Silver, that seems to provide little benefit for disambiguating between different athletes named "Andre Smith." The crux of the problem lies in the fact that database-driven features are domain-specific. Counting how many times the director of a movie appears is highly useful in the movie domain, but worthless in the sports domain.

Our solution works by re-defining the problem in such a way that we can define domain-independent and database-independent features. For example, rather than counting how often the director of a movie appears in the context around a movie mention, we create a domain-independent Count Att(m, s) feature function that counts how often *any* attribute of s appears in the context of m. For movies, Count Att will add together counts for appearances of a movie's production year and IMDB rating, among other attributes. In the sports domain, Count Att will add together counts for appearances of a player's height, position, salary, *etc.*. But in either domain, the feature is well-defined, and in either domain, larger values of the feature indicate a better match between m and s. Thus there is a hope for training a model with domain-independent features like Count Att on labeled data from one domain, say movies, and producing a model that has high accuracy on the sports domain.

We first formalize the notion of a domain adaptation EL model, and then describe our algorithm for producing such a model. We say that a *domain* consists of a database DB as well as a distribution $\mathcal{D}(\mathcal{M})$, where $\mathcal{M}$ is the space of mentions. For instance, the movie domain might consist of the Internet Movie Database (IMDB) and a distribution that places most probability mass on documents about movies and Hollywood stars. In *domain adaptation*, a system observes a set of training examples $(m, s, g(m, s))$, where instances $m \in \mathcal{M}$ are drawn from a *source* domain's distribution $\mathcal{D}_S$ and referents s are drawn from the source domain's database DB_S. The labels $g(m, s)$ are boolean values indicating a correct or incorrect match between the mention and referent. The system must then learn a hypothesis for classifying examples (m, s) drawn from a *target* domain's distribution $\mathcal{D}_T$ and database DB_T. Note that for domain adaptation, we cannot use the more traditional problem formulation in which the referent s is a label (*i.e.*, $s = g(m)$) for the mention, since the set of possible

Domain-Independent Feature Functions	
Count Att:	$\sum_{i,j} f_{i,j}^{att}(m,s)$
Count Sim:	$\sum_{i,j} f_{i,j}^{sim}(m,s)$
Count All:	Count Att + Count Sim
Count Unique:	$\sum_{i,j} \begin{cases} 0 & \text{if } f_{i,j}^{att}(m,s) = 0, \\ 1 & \text{if } f_{i,j}^{att}(m,s) > 0. \end{cases}$
Count Num:	$\sum_{i,j \mid j \text{is a numeric att.}} f_{i,j}^{att}(m,s)$

Table 2.1: Primary feature functions for a domain adaptation approach to EL. These features made the biggest difference in our experiments, but we also tested variations such as counting unique numeric attribute appearances, counting unique similar entities, counting relation name appearances, counting extended attributed appearances, and others.

referents changes from domain to domain, and therefore the output of g would be completely different from one domain to the next.

Table 2.1 lists the domain-independent features used in our domain adaptation model. These features use the attribute counts and similar entity counts from the distant supervision model as subroutines. By aggregating over those domain-dependent feature functions, the domain adaptation system arrives at feature functions that can be defined for *any* database, rather than for a specific database.

Note that there is a tradeoff between the domain adaptation technique and the distant supervision technique. The domain adaptation model has access to labeled data, unlike the distant supervision model. In addition, the domain adaptation model requires no text whatsoever from the target domain, not even an unlabeled corpus, to set weights for the target domain. Once trained, it is ready for EL over any database that meets our assumptions, out of the box. However, because the model needs to be able to transfer to arbitrary new domains, the domain adaptation model is restricted to domain-independent features, which are "coarser-grained." That is, the distant supervision model has the ability to place more weight on attributes like director rather than genre, or team rather than position, if those attributes are more discriminative. The domain adaptation model cannot place different weights on the different attributes, since those weights would not transfer across databases.

As with distant supervision, the domain adaptation strategy uses a log-linear model over these feature functions. We use standard techniques for training the model using labeled data from the source domain: conditional log likelihood (CLL) as the objective function, and LBFG-S for convex optimization.

$$CLL(L, \mathbf{w}) = \sum_{(m,id,label)\in L} \log P(label|m, id, \mathbf{w})$$

The training algorithm is guaranteed to converge to the globally optimal parameter setting for this objective function over the training data. The manually annotated data contains only positive examples; to generate negative examples, we use the same name-matching heuristic $\phi(m, DB)$ to identify a set of potentially confusing bad matches. On test data, we use the trained model to choose the id for a given m with the highest probability of being correct.

2.6 A Hybrid Model

The distant supervision and domain adaptation strategies use two very different sources of evidence for training a disambiguation classifier: the string-matching heuristic and unlabeled text from the target domain for the the distant supervision model, and aggregate features over labeled text from a separate domain for domain adaptation. This begs the question, do these sources of evidence complement one another? To address this question, we design a Hybrid model with features and training strategies from both distant supervision and domain adaptation.

The training data consists of a set L_S of labeled mentions from a source domain, a source database DB_S, a set of unlabeled mentions M_T from the target domain, and the target-domain database DB_T. The full feature set of the Hybrid model is the union of the distant supervision feature functions for the target domain and the domain-independent domain adaptation feature functions. Note that the distant supervision feature functions are domain-specific, so they almost always will be uniformly zero on L_S, but the domain adaptation feature functions will be activated on both L_S and M_T. The combined training objective for the Hybrid model is:

$$LL(L_S, M_T, \mathbf{w}) = CLL(L_S, \mathbf{w}) + MLL(M_T, \mathbf{w})$$

2.7 Experiments

Our experiments compare our strategies for Open-DB EL against one another, as well as against a Wikipedia EL system from previous work, on two domains: sports and movies.

2.7.1 Data

For the movie domain, we collected a set of 156 cult movie titles from an online movie site (www.olivefilms.com). For each movie title, we executed a Web search using a commercial search engine, and collected the top five documents for each title from the search engine's results. Nearly all top-five results included at least one mention of an entity not found in Wikipedia; overall, only 16% of the mentions could be linked to Wikipedia. After stripping javascript and html annotations, we removed documents with fewer than 50 words, leaving a total of 770 documents. We select one occurrence of any of the 156 movie titles from each document as our set of mentions. Many titles are ambiguous not just among different movies with the same name, but also among novels, plays, geographical entities, and assorted other types of entities. To provide labels for these mentions, we use both a movie database and Wikipedia. We downloaded the complete data dump from the online Internet Movie Database (IMDB, www.imdb.com). For our set of possible referents, we use the set of all key values in IMDB, and the set of all Wikipedia articles. Annotators manually labeled each mention using this set of referents. Table 3.5 shows summary statistics about this labeled data.

For the sports domain, we downloaded all player data from Yahoo!, Inc.'s sports database for the years 2011-2012 and two American sports leagues, the National Football League (NFL) and Major League Baseball (MLB). From the database, we extracted ambiguous player names and team names, including names like "Philadelphia" which may refer to `Philadelphia Eagles` in the NFL data, `Philadelphia Phillies` in the MLB data, or the city of `Philadelphia` itself (in both types of data). We then collected 1300 Yahoo! news articles which include a mention that partially matches at least one of these database symbols. We manually labeled a random sample of 564 mentions from this data, including 279 player name mentions and 285 city name mentions. Many player name and place name mentions are ambiguous between the two sports leagues, as well as with teams or players from other leagues. In order to focus on the hardest cases, we specifically exclude mentions like "Philadelphia" from the labeled data if any of their unambiguous completions appears in the same article (that is, if either of the team names "Philadelphia Eagles" or "Philadelphia Phillies" appears in the same article, we exclude the "Philadelphia" mention). As before, the set of possible referents includes the symbol OOD [2], key values from the sports database, and Wikipedia articles, and a given mention may be labeled with both a sports entity and a Wikipedia article, if appropriate.

2.7.2 Evaluation Metric

We report on a version of exact-match accuracy. The system chooses the most likely label $\hat{s}$ for each m. This is judged correct if $\hat{s}$ matches the correct label s exactly, or (in cases where both a Wikipedia and a database entity are considered correct)

[2] Often OOD is analogously termed as NIL in EL literature. Both terms indicate that the target entity is out of the reference knowledge base.

domain	$\lvert M\rvert$	$\mathbf{E}\lvert\phi(m, DB)\rvert$	OOD	Wiki
movies	770	2.6	13%	16%
sports	564	4.5	0%	100%

Table 2.2: Number of mentions, average number of referents per mention, % of mentions that are OOD, and % of mentions that are in Wikipedia in our movie and sports data.

if one of the labels matches $\hat{s}$ exactly. This metric allows systems to resolve against either reference, Wikipedia or another database, without requiring it to match both if the same entity appears in both references.

2.7.3 Exact or Partial Matching?

One important question in the design of our systems is how to determine the "match" between database symbols and text. This question comes into play in two components of our systems: it affects the computation of feature functions that count how often a match of some attribute is found in text, and it affects which set of heuristically-determined database entities are considered to be possible matches for a given mention.

We experiment with two different matching strategies between a symbol s and text t, exact matching and partial matching. Exact matching $\mu_{exact}(s, t)$ requires the sequence of characters in s to appear exactly (modulo character encoding) in t. For instance, the database value `Chris Johnson` would match "Chris Johnson", but not "C. Johnson" or "Johnson" in text. For partial matching, we used different tests for numeric and textual entities. For numeric entities, $\mu_{partial}$ matched s and t if the numeric value of one was within 10% of the other, so that `5312` would match "5,000." We made no attempt to convert numeric phrases, such as "3.6 million", into numeric values. For textual entities, $\mu_{partial}$ matched s and t if at least one token from each matched exactly. Thus `Chris Johnson` matches both "Chris" and "C. Johnson".

System	Accuracy
No-Wikipedia Domain Adapt.	0.61
DocSim-Wikipedia Domain Adapt.	**0.69**

Table 2.3: Including a simple document-similarity feature for comparing a mention's context with a Wikipedia page provides an 8% improvement over ignoring Wikipedia information.

We found $\mu_{partial}$ to be consistently superior for computing $\phi(m, DB)$, since it has much better recall for mentions like "Philadelphia". On the other hand, if we use $\mu_{partial}$ for computing our models' feature functions, like the Count Att(m, s) in the domain adaptation model, counts varied widely across domains. A simple version of the domain adaptation classifier (only the Count All and Count Unique features) trained on sports data and tested on movies achieved an accuracy of 24% using $\mu_{partial}$, compared with 61% using μ_{exact}. For all remaining tests, we used μ_{exact} for computing features, and $\mu_{partial}$ for computing $\phi(m, DB)$.

2.7.4 Incorporating Wikipedia referents

Thus far, all of our features work on relational data, not Wikipedia. In order to allow our systems to link to Wikipedia, we create a single "document similarity" feature describing the similarity between the text around a mention and the text appearing on a Wikipedia page. We build a vector space model of both the document containing the mention and the Wikipedia page, remove stopwords, and use cosine similarity to compute this feature.

To evaluate the effectiveness of this Wikipedia feature, we tested two versions of our domain adaptation system, both trained on sports data and tested on the movies domain. The first version involves no Wikipedia information whatsoever, thus it has no reason to select a Wikipedia article over *OOD*. The second system includes the document similarity feature. Table 2.3 shows the results of these systems. Encourag-

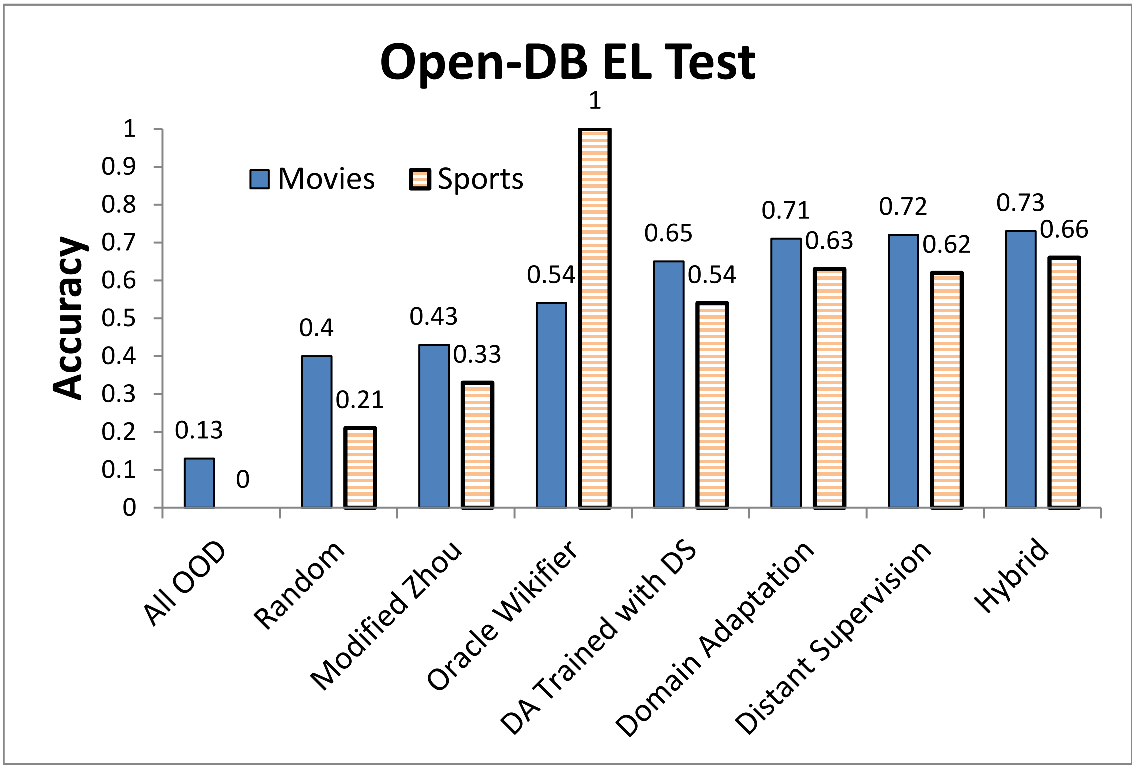

Figure 2.3: **All three Open-DB EL strategies outperform a state-of-the-art Wikipedia EL system by 25% or more on sports and movies, and outperform a Wikipedia EL system with oracle information by 14% or more on the movie data.** Differences between the Modified Zhou Wikifier and the Open-DB strategies are statistically significant ($p < 0.01$, Fisher's exact test) on both domains.

ingly, our single document similarity feature produces a significant improvement over the model without Wikipedia information, so we use this feature in all of our systems tested below. More sophisticated use of Wikipedia is certainly possible, and an important question for future work is how to combine Open-DB EL more seamlessly with Wikipedia EL.

2.7.5 Comparing Open-DB EL Strategies

For each domain, we compare our domain-adaptation strategy, distant supervision, and hybrid strategies. The domain-adaptation model is trained on the labeled data for sports when testing on movies, and *vice versa*. We use a movies test set of 180 mentions that is separate from the development data used for the above tests. For the distant supervision strategy, we use the entire collection of texts from each domain as input (1300 articles for sports, 770 articles for movies), with the labels removed during training.

We compare against a state-of-the-art Wikipedia EL system used in production by a major Web company. This system is a modified version of the system described by *Zhou et al.* (2010), where certain features have been removed for efficiency. We refer to this as the Modified-Zhou Wikifier. This system uses a gradient-boosted decision tree and multiple local and global features for computing the similarity between a mention's context and a Wikipedia article. We also test a hypothetical system, Oracle Wikifier, which is given no information about entities in IMDB, but is assumed to be able to correctly resolve any mention that refers to an entity found in Wikipedia. Thus, this system has perfect accuracy on mentions that can be found in Wikipedia, and accuracy similar to a baseline that predicts randomly on all mentions that fall outside of Wikipedia[3]. Oracle-Wikifier serves as an upper bound on systems that have no access to a domain-specific database. In addition, we compare against two standard baselines: a classifier that always predicts OOD, and a classifier that chooses randomly. Finally, we compare against a system that trains the domain adaptation model using distant supervision ("DA Trained with DS").

Figure 2.3 shows our results. All three Open-DB approaches outperform the baseline techniques on this test by wide margins, with the Hybrid model increasing by 30% or more over the random baseline. On the movie domain, the Hybrid model outperforms the Oracle Wikifier by nearly 20%. Encouragingly, the Hybrid model consistently outperforms both distant supervision and domain adaptation, suggesting that the two sources of evidence are partially complementary. Distant supervision

[3]Alternatively, one could make the oracle system predict OOD on all mentions that fall outside of Wikipedia. Random predictions perform better on our data.

performs better on the movies test, whereas domain adaptation has the advantage on sports. The differences among all three Open-DB approaches is relatively small, compared with the difference between these approaches and Oracle Wikifier on the movie data.

The domain adaptation system outperforms DA Trained with DS on both domains, suggesting that labeled data from a separate domain is better evidence for parameter estimates than unlabeled data from the same domain. The distant supervision system also outperforms DA Trained with DS on both domains, suggesting that the fine-grained, domain-specific features do in fact provide more helpful information than the coarser-grained, domain-independent features of the domain adaptation model.

All of the Open-DB EL systems outperform the Modified Zhou Wikifier on both data sets by a wide margin. In fact the Modified Zhou Wikifier has similar results on both domains, despite the fact that Wikipedia has far greater coverage on sports than movies. In part, the poor performance of the Modified Zhou Wikifier reflects the difficult nature of the task. In previous experiments on an MSNBC news test set it reached 85% accuracy, but a random classifier there achieved 60% accuracy compared with 21% on our sports data. Another difficulty with the Modified Zhou Wikifier is its strong preference for globally common entities. It consistently classifies mentions that are ambiguous between a city and a team (like "Chicago" in "Chicago sweeps the Red Sox") as cities when they should be resolved to teams, in large part because `Chicago` is a more common referent in general text than either of the baseball teams that play in that city. In sports articles, however, both meanings are common, and only the surrounding context can help determine the correct referent.

Besides wikifiers, EL systems may also be compared with dictionary-based word sense disambiguation techniques like the Lesk algorithm (*Lesk* (1986)). The Lesk algorithm is "open" in the sense that it works for arbitrary dictionaries, and it defines a vector space model of the dictionary definitions that may be likened to the attribute-value model in our representation of entities in the database. Our approach, however, estimates parameters for a statistical model from data, whereas the Lesk algorithm uses an equal weight for all attributes. To make an empirical comparison, we created a variant of the Lesk algorithm for relational data: we took the disambiguation model from Eqn. 2.4-1, supplied all of the features from the distant supervision model, and manually set $\mathbf{w} = \mathbf{1}$. This "relational Lesk" model achieves an accuracy of 0.11 on movies, and 0.15 on sports, significantly below the random baseline. Giving equal weight to noisy attributes like `genre` and more discriminative attributes like `director` significantly hurts the performance.

For both the movie and sports domain, approximately 80% of the Hybrid model's errors are because of predicting database symbols, when the correct referent is a Wikipedia page or *OOD*. This nearly always occurs because some words in the context of a mention match an attribute of an incorrect database referent. For instance, the `crime` genre is an attribute for several movies, but it also matches in contexts surrounding book titles and numerous other entities. In the movie domain, most of the remaining errors are incorrect *OOD* predictions for mentions that should resolve to the database, but the article contains no attributes or similar entities to the database entity. In the sports domain, many of the remaining errors were due to predicting incorrect player referents. Quite often, this was because the document discusses a fantasy sports league or team, where players from different professional sports teams are mixed together on a "fantasy team" belonging to a fan of the sport. Since players in the fantasy leagues have different teammates than they do in the database, these articles consistently confuse our methods.

2.8 Conclusion and Future Work

This chapter introduces the task of Open-DB Named Entity Linking, and presents two distinct strategies for solving this task. Experiments indicate that a mixture of the two strategies significantly outperforms a state-of-the-art Wikipedia EL system, on a dataset where Wikipedia has good coverage and on another dataset where Wikipedia has poor coverage. The results indicate that there is a significant benefit to leveraging other sources of knowledge in addition to Wikipedia, and that it is possible to leverage this knowledge without requiring labeled data for each new source. The initial success of these Open-DB EL approaches indicates that this task is a promising area for future research, including exciting extensions that link large numbers of domain-specific databases to text.

CHAPTER 3

JOINT NAMED-ENTITY RECOGNITION AND LINKING

3.1 Introduction

Existing EL systems depend on knowing where the set of names in a text appear. Nearly all EL systems therefore use a pipeline architecture: they run a Named-Entity Recognition (NER) (*Finkel et al.* (2005); *Ratinov and Roth* (2009)) system on the text to identify the boundaries of names. They then use the names that were identified by the NER system, typically called *mentions*, as the input set of names for the EL task. (Many NER systems provide additional information besides the boundaries of the mentions, such as a type label like Person, Location, or Organization. Since our EL system will link mentions to a relational database with a more extensive and fine-grained ontology, we ignore this extra output of the NER system, and consider just the mention boundaries.) Unfortunately, as is common with pipeline architectures for probabilistic models, errors from the NER system propagate to the EL system. Any mention that goes undetected by the NER system clearly cannot be linked correctly by the downstream EL system. Other common errors from NER systems include incorrectly splitting a multi-word name into multiple mentions; joining together two mentions that happen to appear near one another; or leaving off one or more words from a multi-word mention. Each of these errors can cause the downstream EL system to generate too many, too few, or just wrong links.

Figure 3.1 shows an example phrase that has been incorrectly processed by a

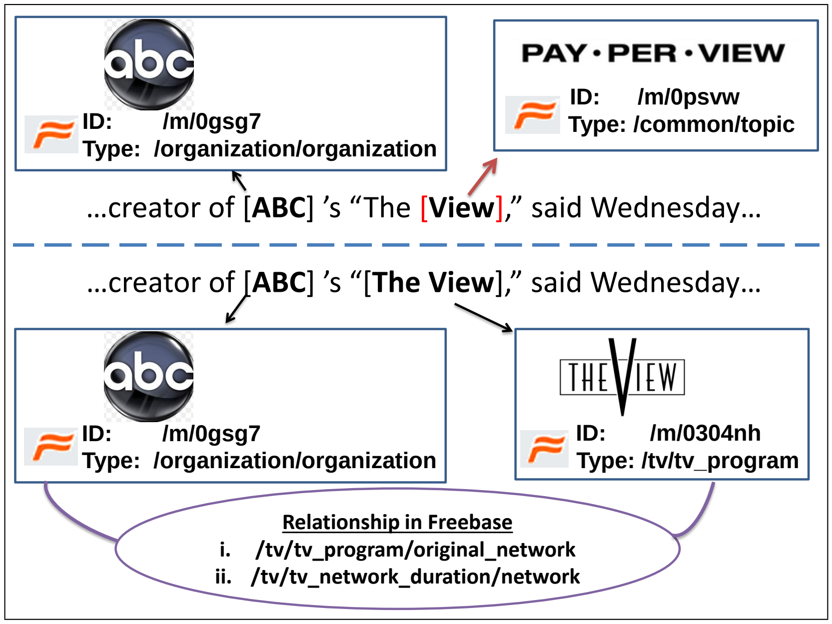

Figure 3.1: **Above:** Incorrect mention boundaries and entity link (for "View") from a state-of-the-art NER system and our pipeline EL system. **Below:** Correct output from our joint model, together with some of the additional relational information from Freebase that helps the joint model recover the correct mention boundaries and entity links.

pipeline of a state-of-the-art NER system (*Ratinov and Roth* (2009)) and our own baseline EL system that links to the Freebase database[1]. The word "the" commonly precedes certain mentions, as in "the [Rocky Mountains]," and so it is reasonable for the NER system to place the mention boundaries around just "View" in this example. Likewise, it is reasonable for an EL system to predict that "View" refers to "Pay-Per-View," a common method of watching certain television programs, since most of the surrounding text is related to television programming.

We propose (and develop) a joint NER and EL model that we use to re-rank candidate mentions and entity links produced by base NER and EL models. Our base NER techniques for example, would produce candidate mentions like ABC, View, The View, and Wednesday from the above phrase. Our base EL system would then propose several promising entity links for each candidate mention, including Pay-Per-

[1] www.freebase.com

View and The View (an American television program on a broadcast channel called ABC). The reranking model then chooses among the set of all possible mention and entity link labelings for the whole phrase to determine the best choice. Since the reranking model considers labels for ABC and (The) View simultaneously, it can use features for known relationships between the television channel ABC and the television program The View to encourage these as outputs. For efficiency, such features are not available to the pipeline models, which consider only one mention at a time. We use the pipeline models to prune the set of all possible candidate mentions and entity links to a manageable size while maintaining high recall. The reranking model can then use more sophisticated features for collective classification over this pruned set.

Previous work (*Cucerzan* (2007); *Milne and Witten* (2008); *Han and Zhao* (2009); *Kulkarni et al.* (2009)) has considered collective classification for entities, but never for the combination of NER and EL. *Ratinov et al.* (2011) argue that collective classification for entity linking provides only a small benefit over a purely-local model. However, we find that by combining NER and EL, the set of correct mentions that are available to be linked improves drastically, and EL performance improves as a result.

The remainder of this paper is organized as follows: The next section discusses related work. Section 3.3 describes the Freebase database, and why it is suited to the EL task. Section 3.4 presents our joint model for NER and EL. Section 3.5 describes our methods for finding candidate mentions and entities, including a pipeline EL system for Freebase. Section 3.6 gives our experimental results, and Section 3.7 concludes.

3.2 Related Work

All but two previous entity linking systems that we are aware of uses a pipeline architecture. That is, the system first attempts to identify the boundaries of named-entity mentions in text using NER techniques. Taking the output of NER as a given, the system then links the extracted mentions to a target catalog of entities. Since available NER tools are imperfect, missing and incorrect mention boundaries limit the performance of the EL system.

Ratinov et al. (2011) point out that finding mentions using NER is a non-trivial task, and hence their experiments report linking performance using gold-standard mentions, as do several other recent EL systems (*Sil et al.* (2012a); *Davis et al.* (2012); *Han et al.* (2011)). Nevertheless, in practice NER performance limits the performance of these EL systems. Our work relies on pipeline NER and EL techniques (*Bunescu and Pasca* (2006); *Mihalcea and Csomai* (2007); *Zhou et al.* (2010); *Ferragina and Scaiella* (2010); *Ratinov et al.* (2011); *Lin et al.* (2012a); *Meij et al.* (2012); *Mendes et al.* (2011); *Lin et al.* (2012b); *Hoffart et al.* (2011); *Mendes et al.* (2012); *Sil et al.* (2012a)) to provide high-quality candidate mentions and entity links. We then subject these candidates to further processing by a joint NER and EL model that outperforms state-of-the-art NER systems and state-of-the-art EL systems on their respective tasks.

Guo et al. (2013) is the closest to our work as they consider joint mention detection and entity disambiguation, but their techniques were developed for, and are best-suited for, microblog text. Because microblog texts are so short, Guo *et al.* can use computationally expensive machinery; they use a structural SVM algorithm (*Taskar et al.* (2003); *Tsochantaridis et al.* (2006)), which requires NP-hard inference. Their technique can consider nearly every token as part of a candidate mention, again because of the short length of microblog texts. In contrast, our techniques are better suited for longer documents. We use linear maximum-entropy models to re-rank a set of candidate mentions and entities provided by efficient NER and EL base models. A more minor difference is that Guo *et al.* link to Wikipedia; our technique links to both Wikipedia and Freebase—a large, user-contributed, relational database. Also, Guo *et al.*'s techniques cannot identify mention boundaries that have no corresponding Wikipedia entries, whereas our techniques can identify mentions with no corresponding entity in our reference set; we follow the Text Analysis Conference's (TAC) guidelines in linking such mentions to the special symbol NIL. *Cucerzan* (2012) is an extension of the *Cucerzan* (2007) system which performs full document entity extraction and disambiguation. However, the authors do not postpone mention boundary decisions for *all* mentions which our system performs. Also, our system performs re-ranking based on the links provided by a pipeline EL system and information extraction from both structured and semi-structured resources which is a novel aspect for an EL system. Finally, *Cucerzan* (2012) employ global models for EL which as per *Ratinov et al.* (2011) do not provide significant improvements over local models like *Sil et al.* (2012a).

Regarding reference sets, most existing research on entity linking falls into two categories: systems that use Wikipedia as a reference set (often called wikifiers) (*Bunescu and Pasca* (2006); *Mihalcea and Csomai* (2007); *Cucerzan* (2007); *Milne and Witten* (2008); *Fader et al.* (2009); *Han and Zhao* (2009); *Kulkarni et al.* (2009); *Zhou et al.* (2010); *Ferragina and Scaiella* (2010); *Ratinov et al.* (2011); *Lin et al.* (2012a); *Meij et al.* (2012)), and systems that use large relational databases like Yago or DBpedia as a reference set (*Mendes et al.* (2011); *Hoffart et al.* (2011); *Mendes et al.* (2012)). *Sil et al.* (2012a) use a combination of Wikipedia and database-independent features to create linkers for databases like IMDB and the Yahoo! Sports pages. In empirical comparisons, Sil *et al.* outperforms the *Zhou et al.* (2010) wikifier, and Hoffart *et al.*'s Yago linker (*Hoffart et al.* (2011)) outperforms another recent wikifier (*Kulkarni et al.* (2009)), suggesting that large relational databases hold promise as reference sets. Part of the reason may be that the large databases provide better coverage of named entities; *McNamee et al.* (2009) show that 57% of the named-entities in the Text Analysis Conference's (TAC) 2009 EL task do not have any referents in Wikipedia. *Lin et al.* (2012a) and *Lin et al.* (2012b) show that 33% of their mentions in a data set of 500 million Web documents cannot be linked to Wikipedia. However, empirical comparisons have so far been limited, and thus far none of the database linkers has compared against the state-of-the-art (*Ratinov et al.* (2011)) wikifier. Our system uses both Wikipedia and Freebase, a large, user-contributed relational database. We compare against a number of the best wikifiers and database linkers, and outperform all them in several different settings.

Previous work on EL with large relational databases like DBpedia (*Mendes et al.* (2011); *Mendes et al.* (2012)) and Yago (*Hoffart et al.* (2011)), have been successful in outperforming some wikifiers like *Kulkarni et al.* (2009), although not necessarily the current state-of-the-art Ratinov *et al.* wikifier (*Ratinov et al.* (2011)). Unlike Freebase, DBpedia and Yago are automatically extracted from text, and as a result they potentially contain noisier data. DBpedia, which extracts its data from Wikipedia, contains 3.7 million entities, around six times smaller than Freebase. NereL's methods also differ significantly from Hoffart *et al.*: NereL is capable of correctly identifying cases where a mention refers to an entity not in the target catalog; it leverages both a structured database and Wikipedia. Hoffart *et al.* attempt to solve an NP-hard dense subgraph problem through approximation techniques, which can be difficult to scale up. We restrict NereL to straightforward linear classifiers. In our previous chapter and *Sil et al.* (2012a), we introduce an Open-DB EL framework, in which the task is to resolve an entity to Wikipedia or any relational database. By leveraging structured, relational data our systems achieve strong accuracy, outperforming the *Zhou et al.* (2010) wikifier. Our results indicate that there is a significant benefit to leveraging other sources of knowledge in addition to Wikipedia, and that it is possible to leverage this knowledge without requiring labeled data for each new source. Because of these attractive properties, we use the *Sil et al.* (2012a)'s domain-independent system as a subset of our pipeline EL model. In *Sil et al.* (2012a), the techniques are tested only on domain-specific datasets where Wikipedia may not be as useful, and we cover two extremely different domains: sports and movies. We also make only very limited use of Wikipedia in our EL model, essentially computing a single cosine similarity feature. Also, that system does not perform any mention detection and assumes gold mentions as input. NereL performs joint NER and EL and covers far more domains through its use of Freebase, makes extensive use of Freebase-specific features as well as more extensive use of the information in Wikipedia. Experimentally we outperform both the *Sil et al.* (2012a) system and 5 other state-of-the-art EL systems on three benchmark EL datasets where Wikipedia has good coverage.

3.3 Entity Linking with Freebase

Most existing large-scale entity linkers use Wikipedia titles as a reference set. We chose to use a combination of Freebase and Wikipedia as our reference set because it provides us all of the information in the Wikipedia articles, as well as a well-developed ontology of types and binary relations available in Freebase. Freebase is a freely-available, user-contributed repository of structured data containing almost 40 million unique entities. In comparison, Wikipedia contains 4.2 million entities and DBpedia [2] contains 3.77 million entities (all numbers are for the English versions). The Freebase schema is divided into *domains*, each of which contains its own set of *types* (such as `people`, `books`, `bridges`, *etc.*) and *properties*, like `date of birth` for a person or `latitude` and `longitude` for a location. Freebase commons, which we use in our experiments, contains 86 different domains, including such diverse areas as schools, geography, sports, and astronomy. Every entity in Freebase is given

[2] http://dbpedia.org/About

Mention:	*Mike Tyson*
Candidates:	{`Mike Tyson, Mike Tyson's Punch-Out!!, Fallen Champ: The Untold Story of Mike Tyson,...`}
Mention:	*M. Tyson*
Candidates:	{`M. Tyson Gilpin Jr., John M. Tyson, Mike Tyson,...`}
Mention:	*Iron Mike*
Candidates:	{`Iron, Iron & Wine, Mike Tyson, ...`}

Table 3.1: Examples of automatic candidate generation using Freebase.

a unique *ID*, and separately a set of possible names that can refer to the entity, through the built-in `name` and `alias` relations. Many Freebase entities also have links to their corresponding Wikipedia pages, when available. Furthermore, Freebase includes a built-in, nontrivial API for finding candidate entity links for a given name, which is based on the user-contributed `name` and `alias` relations. Using only this API, we were able to identify a set of candidates that contain the correct entity for 672 mentions (90%). Thus Freebase's user contributions already come close to solving one of the challenges for EL. Standard practice (*Ratinov et al.* (2011); *Milne and Witten* (2008)) for evaluating EL models evaluates them over the gold set of mentions and a candidate set that always includes the correct candidate (if it exists in Freebase), so this difference does not show up in our empirical comparison with wikifiers. An informal inspection of Freebase's candidate set for variations of the name "Mike Tyson", as shown in Table 3.1, indicates that Freebase provides a nontrivial name-matching facility.

3.4 Joint Named-Entity Recognition and Entity Linking

3.4.1 Examples

While EL systems can go wrong even when provided correct mentions, we have found that in practice a large number of EL errors are caused by poor mention

Error Type	Bad Example	Correct Example
Under-segmentation	[Orange County, Calif.]→`Orange County`	[Orange County]→`Orange County` [Calif.]→`California`
Over-segmentation	[Blue Cross]→`BCBS of Alabama` and [Blue Shield of Alabama]→`BCBS of Alabama`	[Blue Cross and Blue Shield of Alabama]→ `BCBS of Alabama`
Mis-segmentation	[Florida Supreme]→`(FL Supreme Court)` Court	[Florida Supreme Court]→`FL Supreme Court`

Table 3.2: Examples of three kinds of NER errors that cause a pipeline EL system to miss links or add extra links. Mention boundaries are shown in [square brackets], and entity links are indicated by →.

boundaries. Table 3.2 shows examples of three common errors by NER systems that propagate to EL systems. Other types of errors include completely-missed mentions, and correct mentions but incorrect links. Joint NER and EL can help with many of these problems by allowing for features that, for instance, favor joining two mentions that link to the same entity over having the same entity repeated twice.

3.4.2 Problem Formulation

Labeled data for joint NER and EL consists of a set of documents D, and for each document $d \in D$, a set $Entities(d)$ of triples of the form (l, r, e). Here, l and r indicate the left and right boundaries of the mention of some named entity, and e indicates the unique identifier of the named entity in a reference set. Given a labeled training set T of this form, the task is to learn a model that can predict $Entities(d)$ for new documents d.

3.4.3 Decomposition into Connected Components

Classification over all possible sets of (mention, entity)-pairs for a document is generally intractable. Typical approaches to EL have addressed this by decomposing the task into a pipeline: first, identify the set of mentions using a standard NER model. Second, determine a local model for $P(E|d, l, r)$, where E is a random variable over entity ids e. Many recent EL approaches have investigated a third step, which is to select up to K tuples of entity ids $(e_1, \ldots, e_n)$, one e_i for each mention in the document found by the NER model, and then build a model to rank these tuples of entity ids (*Bunescu and Pasca* (2006); *Cucerzan* (2007)). However, *Ratinov et al.* (2011) argue that this kind of global model provides a relatively small improvement over the purely-local approach, and *Sil et al.* (2012a) dispense with the global classification step entirely.

We instead adopt a different approach to decomposing the task, which offers a way to jointly classify mention boundaries and entity links. Rather than assume that the first-stage NER model produces the correct mention boundaries for linking, we assume that it provides a high-recall set of potentially-overlapping candidate mentions. One way to produce this set is to develop a probabilistic NER model $P(M|d,l,r)$, where M is a binary random variable indicating whether l and r are the exact boundaries of a mention. By selecting a relatively low threshold for this distribution, the model can produce a high-recall set of candidate mentions for d that have some reasonable probability of being correct. In practice, we have found that an alternative approach of getting candidate mentions (described in Sec. 3.5.1), which uses an NP chunker and an existing NER system, to be efficient and effective. Let A be the set of all sequences $a_i = (m_1, \dots, m_{n_i})$ of mentions m such that each m is taken from the candidate set, and no two mentions in the same sequence have overlapping boundaries. As in the pipeline model, we also develop a local model $P(E|d,l,r)$, and select several promising candidates from the local model for each candidate mention. The joint classification task is over all tuples B of the form $b_i = (m_1, e_1, \dots, m_{n_i}, e_{n_i})$, where each e_j is taken from the local EL model's candidate set for m_j. We call these *entity-mention tuples*.

The cost of high recall in the set of candidate mentions is that the set of candidate mentions is large, and as a result B contains far too many candidate tuples for tractable classification. To decompose the problem, we observe that most words in (non-microblog) text are not part of named-entities. We adopt a simple heuristic of separating all candidate mentions that are three or more words apart. In all three of the corpora in our experiments, this simple heuristic results in a partitioning of the mentions into small sets of mentions that appear near one another. We refer to these sets as the *connected components* of d, or $CC(d)$. We perform classification over the set of entity-mention tuples $B(cc)$ that are formed using candidate mentions and entities within the same connected component $cc \in CC(d)$. As an example, the phrase "ABC's The View" from Figure 3.1 would constitute a connected component, since several candidate mentions (ABC and View) are separated by three or fewer tokens. Two of the entity-mention tuples for this connected component would be ([ABC], `ABC`, [View], `Pay-Per-View`) and ([ABC], `ABC`, [The View], `The View`).

In over 98% of our training documents, all of the connected components contained tuples b_i with $n_i \leq 10$ mentions. The few remaining large connected components were from documents describing the results of cricket matches, where long lists of player names were separated only by one or two statistics each. For such long sequences, we heuristically divide them into chunks of a length that is the maximum length that our implementation can handle, which is 15. With relatively low thresholds for selecting candidate mentions and entities, the number of tuples in each connected component in our training data was 31.02 on average, while still allowing a maximum possible recall of 96%. In contrast, the maximum possible recall for a pipeline EL system using a standard NER system in our datasets is 82%. The joint model adds an extra layer of processing on top of the pipeline model, but in experiments the re-ranking step has nearly identical running time as the combination of our pipeline NER and EL models. Overall, this simple decomposition leads to a tractable joint NER and EL classification task.

3.4.4 Re-ranking Model

We use a maximum-entropy model to estimate $P(b|d, cc)$, the probability of an entity-mention tuple b for a given connected component $cc \in CC(d)$. The model involves a vector of real-valued feature functions $\mathbf{f}(b, d, cc)$ and a vector of real weights $\mathbf{w}$, one weight per feature function. The probability is given by

$$P(b|d, cc, \mathbf{w}) = \frac{\exp\left(\mathbf{w} \cdot \mathbf{f}(b, d, cc)\right)}{\sum_{b' \in B(cc)} \exp\left(\mathbf{w} \cdot \mathbf{f}(b', d, cc)\right)} \tag{3.4-1}$$

We use L2-regularized conditional log likelihood (CLL) as the objective function for training:

$$CLL(T, \mathbf{w}) = \sum_{(b,d,cc) \in T} \log P(b|d, cc, \mathbf{w}) + \sigma \|\mathbf{w}\|_2^2$$

where $(b, d, cc) \in T$ indicates that b is the correct tuple of entities and mentions for connected component cc in document d in training set T, and σ is a regularization parameter. Note that this training objective treats each cc equally, even though some will contain more mentions and entities than others. In contrast, we use standard experimental evaluations (precision and recall) which will treat each mention and entity equally. We experimented with modified versions of the training objective to account for this difference, but these had little empirical impact, possibly because there was a relatively small variance of 1.02 in the number of mentions per b tuple.

We used LBFG-S for gradient-based convex optimization. The gradient for our objective function is given by

$$\frac{\partial CLL}{\partial w_i} = 2\sigma w_i + \sum_{(b,d,cc) \in T} f_i(b, d, cc) - \sum_{\substack{d \in T \\ cc \in CC(d) \\ b' \in B(cc)}} P(b'|d, cc, \mathbf{w}) f_i(b', d, cc)$$

The training algorithm is guaranteed to converge to the globally optimal parameter setting for our objective function over the training data.

3.4.5 Features

The primary advantage of the discriminative re-ranking approach is that it provides the flexibility to include many novel features into NER and EL. For instance, we include features that penalize NER boundaries that oversegment a mention by counting how often an entity-mention tuple b links to the same entity multiple times in a row. We also include features that help to learn and encourage common sequences of entity types, like (job title, employee name) and (city, country or state). These features help both to prevent undersegmentation in NER and to improve linking accuracy. Details of our features are below.

Token-level features: Given b, d, and cc, we define features that find the minimum (**MIN**), maximum (**MAX**) and the average (**AVG**) number of tokens inside the mentions in b, and a **NUM** feature for the total number of mentions in b. We also use features that measure how many tokens are not part of any mention in b, but are part of a mention in some other $b' \in B(cc)$. Let $Non\_Mention\_Tokens(b, cc)$ be the set of all such tokens, and $All\_Tokens(cc)$ be the set of all tokens for cc. We construct the **Non-Mention-Count** feature as $|Non\_Mention\_Tokens(b, cc)|$, and the **Non-Mention-Ratio** feature as $\frac{|Non\_Mention\_Tokens(b,cc)|}{|All\_Tokens(cc)|}$.

Capitalization: Similar to *Guo et al.* (2013), we include one binary feature to check if all candidate mentions in b are capitalized, one binary feature to check if all words in all mentions in b are capitalized, and one binary feature to check if there is any mention that has capitalized, non-mention tokens to its immediate left or right.

Features from entity links: We include a number of features that make use of the entity links in b, as well as the types and known relations between entities, as stored in Freebase. The most basic versions of these features include: **Count-Nil**, which counts the number of entities that link to NIL, to help penalize tuples b that are poor matches with Freebase; **Count-Exact-Match**, which counts the number of mentions whose surface form matches exactly with one of the names for the linked entity stored in Freebase; **All-Exact-Match**, which is true if all mentions in b match a Freebase name exactly; and **Acronym-Match**, if the mention's surface form is an acronym for a name or alias of the linked entity in Freebase.

The **Distinct-Links-Per-Mention** measures the ratio of distinct entities in b to mentions in b. This feature helps to penalize over-segmented phrases. This is because two mentions that should have been joined together often have high probability for links to the same entity, as in a bracketing of "[Home] [Depot]" that splits this name for a business into two parts, each of which has a high probability of linking to the company `Home Depot`. Typically, any value of less than 1 for this feature is an indication that b is oversegmented, as it is uncommon to refer to the same entity twice in the short span of text that makes up one of our connected components, although appositives are an obvious exception.

To further establish the relation between mention candidates and their links, we define two features that mimic the Normalized Google Distance (NGD) as in *Guo et al.* (2013); *Ratinov et al.* (2011); *Cilibrasi and Vitanyi* (2007). For every consecutive pair of entities (e, e') that belongs to mentions in b, the **NGD-Freebase** feature computes $\sum_{(e,e')\in b} \frac{|types(e) \cap types(e')|}{|types(e) \cup types(e')|}$. The Wikipedia version, **NGD-Wikipedia**, computes $\sum_{(e,e')\in b} \frac{|Wiki(e) \cap Wiki(e')|}{|Wiki(e) \cup Wiki(e')|}$, where $Wiki(e)$ denotes the set of tokens in the Wikipedia page of e, excluding stop-words.

Two features, **Entity-Type-PMI** and **Entity-Type-Product-PMI**, make use of Freebase's type system to find patterns of entities that commonly appear next to one another. Let $T(e)$ be the set of Freebase types for entity e. We remove common Freebase types which are associated with almost every entity in text, like `/location/location` or `/people/person`, since they have lower discriminating power. From the training data, the system first computes pointwise mutual information (PMI) *Turney* (2002) scores for the Freebase types of consecutive pairs of entities,

(e_1, e_2):

$$PMI(T(e_1), T(e_2)) = \frac{\sum_{(e,e')\in T} \mathbf{1}[T(e_1) = T(e) \wedge T(e_2) = T(e')]}{\sum_{e\in T} \mathbf{1}[T(e_1) = T(e)] \times \sum_{e\in T} \mathbf{1}[T(e_2) = T(e)]}$$

where the sum in the numerator is taken over consecutive pairs of entities (e, e') in training data. The feature **Entity-Type-PMI** adds these scores up for every consecutive (e_1, e_2) pair in b. The feature **Entity-Type-Product-PMI** does the same, but uses this alternative variant of the PMI score:

$$ProductPMI(T(e_1), T(e_2)) = \prod_{i,j} \frac{\sum_{(e,e')\in T} \mathbf{1}[T_i(e_1) \in T(e) \wedge T_j(e_2) \in T(e')]}{\sum_{e\in T} \mathbf{1}[T_i(e_1) \in T(e)] \times \sum_{e\in T} \mathbf{1}[T_j(e_2) \in T(e)]}$$

The **Binary-Relation-Count** feature encourages entity tuples b where the entities have a known relationship with one another. For instance, for the text "Home Depot CEO Robert Nardelli", we wish to encourage the tuple that links "Home Depot" to the entity id of the company `Home Depot`, and "Robert Nardelli" to the entity id of the person `Nardelli`. Freebase contains a relation called `organization_board_memberships` that indicates that `Nardelli` is a board member of `Home Depot`. **Binary-Relation-Count** counts the number of such relations between every consecutive pair of entities in b. This feature helps not just to improve entity linking accuracy; it also helps to prevent under-segmented phrases. For instance, an entity-mention tuple b that contains just a single mention for the whole phrase "Home Depot CEO Robert Nardelli" would have a lower count for this feature than the correctly-segmented version, even if the incorrect mention was linked to `Nardelli`.

Finally, we include a similar version of this feature that uses Wikipedia rather than Freebase. The **Wikipedia-Cooccurrence-Count** feature computes, for every pair of consecutive entities $(e, e') \in b$, the number of times that e' appears in the Wikipedia page for e, and *vice versa*. It adds these counts up to get a single number for b.

3.5 Base NER and EL Models

3.5.1 Overgenerating candidate mentions

Previous work on EL systems (*Ratinov et al.* (2011); *Guo et al.* (2013)) have argued that EL systems often suffer because of the mention detection phase. The impact is mostly because of false negative mentions: any missed mention by the NER system is also a missed entity for EL. To improve on the pipeline model, our joint system requires *overgeneration* of candidate mentions: that is, a high-recall set of mentions with just enough precision to prevent the number of false positives from overwhelming subsequent processing.

Rather than modify an existing NER system to overgenerate mentions, we adopt an approach of combining multiple existing systems and heuristic techniques. We use the mentions found by the state-of-the-art UIUC NER system[3]. In addition, we included strings matching the following heuristic tests as candidate mentions:

- All Noun-phrase (NP) chunks from the Illinois shallow parser (*Punyakanok and Roth* (2001)). We also retain all sub-expressions of up to 7 tokens of the NP chunks.
- All possible n-grams (for n $\leq$ 8) which match names and aliases of entities in Freebase.
- For any string selected by any of the above techniques, we add in one new mention that includes the word to the left, and another new mention that

[3] We used the best settings possible, which meant using both CoNLL training and development data as the training set.

includes the word to the right.

3.5.2 A Base Entity Linker using Freebase

Let mention $m = (d, l, r)$, and let S be a reference set containing entity ids and the special symbol NIL, which indicates an entity that does not match any id in the reference set. The task of a base (or local) EL model is to predict $P(E|d, l, r)$, where random variable E ranges over S. We call our base EL model BaseEL.

BaseEL, like our re-ranking model, is a maximum entropy model. Using Freebase and Wikipedia, we construct a vector of real-valued feature functions $\mathbf{f}(m, s)$ describing the degree to which m and $s \in S$ appear to match one another. The feature functions are described in the following subsections. The model includes a vector of real weights $\mathbf{w}$, one weight per feature function.

$$P(s|m, \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(m, s))}{\sum_{s' \in S} \exp(\mathbf{w} \cdot \mathbf{f}(m, s'))} \tag{3.5-2}$$

As with the joint model, we use L2-regularized CLL as the objective function, and LBFG-S for gradient-based convex optimization. We use only gold-standard mentions for training the base EL model. Our manually annotated data contains only positive examples for entity ids. To generate negative examples, we use Freebase's candidate generation technique (described in Section 3.3) to identify a set of potentially confusing bad matches. Let $\phi_{FB}(m)$ denote the set of candidate entities that Freebase returns for mention m, plus NIL and the correct entity s if s is not already in $\phi_{FB}(m)$. Using this set as the complete set of possible labels for m, the gradient for our objective function is given by

$$\frac{\partial CLL}{\partial w_i} = \sum_{(m,s) \in T} f_i(m, s) - \sum_{s' \in \phi_{FB}(m)} P(s'|m, \mathbf{w}) f_i(m, s') + 2\sigma w_i$$

3.5.2.1 Freebase Features

BaseEL uses a combination of information from Wikipedia and Freebase to judge the similarity between a mention m and Freebase entity s. We first describe how it extracts information from Freebase. Table 3.3 lists all of the Freebase features used in BaseEL. As a starting point, we leverage the four domain-independent features described in our initial work in *Sil et al.* (2012a), adapted to Freebase. Intuitively, these features measure the number of attribute values and attribute names for entity s in Freebase that appear in the context of m. We split the attribute counts into four different kinds, the **Count All** through **Count Label** features in Table 3.3. As an example, consider the entity `Timberlake`, a performing artist. This entity appears in Freebase relations called `film`, `band member`, and `date of birth`, among many others. Within these relations are tuples that contain `Timberlake`, as well as a mixture of other entity IDs and numeric values. For instance, in the `band member` relation is a tuple containing

$$t = (\texttt{Timberlake}, \texttt{'N Sync}, 1995, 2002, \texttt{Tenor})$$

We collect the names of all entity IDs like `'N Sync` that appear in such tuples into a set $Att_{name}(\texttt{Timberlake})$, and numeric values like 1995 are collected into a set $Att_{num}(\texttt{Timberlake})$. Freebase also contains attribute names for the attributes in each relation; for example, "band," "member," and "period (start)" are the names of three attributes in the `band member` relation. The names of the attributes for relations that `Timberlake` appears in are gathered into the set $Att_{label}(\texttt{Timberlake})$.

Consider this snippet of Web text:

> Last Saturday night we witnessed N'Sync band member Timberlake performing with Britney Spears for the first time.

For the mention of "Timberlake," the **Count All** and **Count Unique** features would have a value of 1 each, because the phrase "N Sync" in the text matches one of the names in Att_{name}(`Timberlake`). The **Count Label** feature would have a value of 2, since "band" and "member" in the Web text both match entries in Att_{label}(`Timberlake`). **Count Numeric** would have a value of zero.

Initial experimental results indicate that a linear classifier over these four simple features, adapted from Sil *et al.* (our previous work) to use on Freebase, already perform better than two state-of-the-art Wikifiers. However, Freebase has more information to exploit, beyond what these four features measure. We begin by observing that a fairly common error during training is to assign a high probability to an incorrect entity that has a very high **Count All** feature, but very low (or zero) counts for other features. Similarly, in some cases the model assigns a high probability to an incorrect entity with a high **Count Label** feature, but low values for other features. On the other hand, correct entities tend to have positive and somewhat balanced values for all four of these features. For example, a document describing a Justin Timberlake concert in London mentions "London" several times, and the singer's band and a well-known song once. The incorrect entity `Timberlake Wertenbaker`, a British playwright who resides in London, has a high value for **Count All**, but low values for all other features. The singer has a small but positive value for **Count All**, **Count Unique**, and **Count Label**. In response, we introduce an **Entropy** feature to measure the balance of the feature values across Sil *et al.*'s features. Entities which have relatively similar values for all four features will have a high value for the **Entropy** feature, and a low value if the original four features are unbalanced.

Our final kind of Freebase feature takes advantage of its type system. Every entity in Freebase is associated with one or more types, and the types are structured in a hierarchical taxonomy. For example, the performing artist entity `Timberlake` comes with the types: `Music_Artist`, `Music_Group_Member`, `TV_Actor`, `Film_Actor`, and others. Likewise, `Britney Spears` is a `Music_Artist`, `TV_Actor`, and `Lyricist`.

`Timberlake, Ohio` has the types `Town` and `Statistical_Region`. As the text in the snippet of Web text above suggests, seeing "Britney Spears" in the context of "Timberlake" is an indicator that "Timberlake" refers to the singer, rather than the town. In this example, `Britney Spears` and `Timberlake` share two types, while `Britney Spears` and `Timberlake, Ohio` share none.

Our **TF** feature counts, for each type t that an entity s belongs to, the number of mentions in the context of m that have any candidate entities with the same type. The feature then adds up these counts for all the types of s. We also compute variations of this feature that compute the average and max score over all of the different types of s, rather than the sum. Note that for the sake of efficiency, we make no attempt at collective classification in BaseEL (*c.f. Ratinov et al.* (2011)); the **TF** feature for "Timberlake" does not depend on the predicted entity for "Britney Spears", but rather on the set of all candidate entities for "Britney Spears."

Analogously to the well-known Inverse Document Frequency heuristic, we define a separate weighting scheme for measuring the importance of a given type for determining the similarity of s to the context of m. For instance, `Timberlake` also belongs to the type `Person`. However, this type is far too inclusive and common to be of much use for discriminating between entities in most circumstances. On the other hand, the type `Music_Artist` is much less common in text, so the fact that `Timberlake` shares this type with `Britney Spears` should be more informative than the fact that these two entities share the type `Person`. We first compute a Document Frequency score $DF(t)$ for all types t in Freebase by counting the number of documents in our training data that had some candidate entity belonging to that type. Let $\#D$ denote the total number of candidate entities in the training data. We then compute a **TF-IDF** feature by reweighting the type frequency for type t by

$$IDF(t) = \log \frac{\#D}{DF(t)}$$

Freebase Feature Functions	
COUNT ALL:	$\sum_{n \in Att_{name}(s)} \text{Count}(m, n)$
COUNT UNIQUE:	$\sum_{n \in Att_{name}(s)} \mathbf{1}[\text{Count}(m, n) > 0]$
COUNT NUM:	$\sum_{n \in Att_{num}(s)} \text{Count}(m, n)$
COUNT LABEL:	$\sum_{n \in Att_{label}(s)} \text{Count}(m, n)$
ENTROPY:	$\sum_{i \in \{\text{Sil } et\ al. \text{ feature indices}\}} -f_i(m, s) \log f_i(m, s)$
TF-SUM:	$\sum_{t \in \text{T}(s)} \sum_{s' \in \text{CE}(m)} \mathbf{1}[t \in T(s')]$
TF-MAX:	$\max_{t \in \text{T}(s)} \sum_{s' \in \text{CE}(m)} \mathbf{1}[t \in T(s')]$
TF-AVG:	$\text{avg}_{t \in \text{T}(s)} \sum_{s' \in \text{CE}(m)} \mathbf{1}[t \in T(s')]$
TF-IDF-SUM:	$\sum_{t \in \text{T}(s)} IDF(t) \left(\sum_{s' \in \text{CE}(m)} \mathbf{1}[t \in T(s')] \right)$
TF-IDF-MAX:	$\max_{t \in \text{T}(s)} IDF(t) \left(\sum_{s' \in \text{CE}(m)} \mathbf{1}[t \in T(s')] \right)$
TF-IDF-AVG:	$\underset{t \in \text{T}(s)}{\text{avg}}\ IDF(t) \left(\sum_{s' \in \text{CE}(m)} \mathbf{1}[t \in T(s')] \right)$

Table 3.3: BaseEL's feature functions for Freebase. Count(m, n) denotes the number of times that string n appears in the document containing mention m. T(s) denotes the set of Freebase types for entity s. CE(m) indicates the set of all candidate Freebase entities for all other mentions in the document containing m.

As with the TF feature, we compute three variations of the feature that sum, average, and max over all types t of a given entity s.

3.5.2.2 Wikipedia Features

For many Freebase entities s, there exists a link to the corresponding Wikipedia page, which we denote by $W(s)$. Hence, we have access to unstructured text which we can use to compute useful additional features that can help our linking algorithm decide if a target concept best describes the input mention. Table 3.4 describes these features. Our Wikipedia features are simplified versions of the features used in the local model of *Ratinov et al.* (2011). Ratinov *et al.* also consider a more complex "global" model that performs collective classification, with a small improvement in

Wikipedia Feature Functions
cosine similarity$(Text(W(s)), Text(m))$
cosine similarity$(Text(W(s)), Context(m))$
cosine similarity$(Context(W(s)), Text(m))$
cosine similarity$(Context(W(s)), Context(m))$
$\forall i \in \{1, .., 5\}\ .\ Text(m)_{tw_i}$
$\forall i \in \{1, .., 5\}\ .\ Context(m)_{tw_i}$

Table 3.4: BaseEL's Wikipedia feature functions.

performance over the local model. As mentioned previously, we forego this type of collective classification, although combining combining joint NER-EL and collective EL is an important task to consider for future work.

The basic intuition behind these Wikipedia features is that a mention m appearing in document d is more likely to refer to entity s described on Wikipedia page $W(s)$ if $W(s)$ has high textual similarity to d. For each Wikipedia page p, we construct a vector space model of p, which we denote as $Text(p)$. We also create a more localized vector space model of the entity for page p by creating a vector space model from the 100 closest words to the first occurrence of a mention of the entity on p, or the first 100 words of p if we cannot identify a mention of the entity. We call this vector space model $Context(p)$. Similarly, we create vector space models $Text(m)$ and $Context(m)$. We then use cosine similarity over these vector space models as features.

In addition to the features inspired by Ratinov *et al.*, we extracted the top 5 most frequently occurring words (excluding stop-words) from each Wikipedia target page p, which we refer to as tw_1 through tw_5. For each tw_i, we create two features, one that counts how often tw_i occurs in the document containing m, and one that counts how often tw_i occurs in the 100 word window surrounding m.

3.6 Experiments

We evaluate the performance of NereL first on the NER task against 2 state-of-the-art NER systems, and second on the EL task against 6 state-of-the-art EL systems.

Datasets: We use 3 standard datasets for EL as our test sets: the ACE dataset, as annotated by *Ratinov et al.* (2011) for Wikipedia links; the MSNBC dataset, as collected and annotated by *Cucerzan* (2007) for mention boundaries and Wikipedia links; and the CoNLL-2003 NER dataset (testb) (*Tjong Kim Sang and De Meulder* (2003)), with YAGO2 and Freebase IDs added by *Hoffart et al.* (2011). Table 3.5 provides key statistics on these datasets. For datasets annotated with Wikipedia links, we automatically extracted the corresponding Freebase IDs using Freebase's built-in links to Wikipedia. In less than 1% of the examples for all the 3 test sets, Wikipedia contained a correct entity, but Freebase did not. We left the Wikipedia page as the correct label for these examples, and our BaseEL and NereL systems always gets these examples wrong. In 10% of the MSNBC examples, Wikipedia had no entry, and the examples were marked NIL (ACE had no NIL entries). We left these NIL cases even when Freebase does contain a correct entity, to provide a fairer comparison with wikifiers. The MSNBC data consists of news articles covering 10 domains, 2 articles from each domain: Business, Entertainment, Politics, Science, etc, and is thus more diverse than either the CoNLL or ACE datasets. The ACE dataset includes hand-labeled entity links for only the first nominal mention of each annotated coreference chain, and there are many correct mentions and entities which have not been labeled. This gold standard allows us to estimate the recall of an NER system, but not the precision, since many of the system's guesses that don't match the gold standard may still be correct. To estimate precision for NER on this dataset, we took a random sample of 100 examples of each system's guesses, and manually judged whether they were correct or not. For EL, we measured precision and recall

domain	$\|M\|$	$\mathbf{E}\|\phi_{FB}(m)\|$	NIL	In Freebase
ACE	257	43.65	0%	100%
MSNBC	747	28.15	10%	89%
CoNLL	5616	27.85	20%	79%
Wikipedia	158715	12.62	0%	100%

Table 3.5: Number of mentions, average number of candidate referents, % of mentions that are NIL, and % of mentions that are in Freebase (and Wikipedia) in our datasets. We train our system on the Wikipedia dataset.

only for the gold-standard mentions in the dataset.

Our training dataset consists of 10,000 random Wikipedia pages, where all of the phrases that link to other Wikipedia articles are treated as mentions, and the target Wikipedia page is the label. The dataset is made available by Ratinov *et al.*. For training, we disregard mentions for which there is no Freebase entity that links to the target Wikipedia page; for the remaining mentions, we use the corresponding Freebase entity as the label. We ended up with 158,715 labeled mentions with an average of 12.62 candidates per mention. The total number of unique mentions in the data set is 77,230 with a total of 974,381 candidate entities and 643,810 unique candidate entities.

Evaluation Metric: For NER evaluation, we measure precision, recall and F1 scores as shown in (*Finkel et al.* (2005); *Ratinov and Roth* (2009)). To evaluate EL accuracy, we report on a Bag-of-Freebase ids (BOF) evaluation analogous to the Bag-of-Titles (BOT) F1 evaluation as introduced by *Milne and Witten* (2008); *Ratinov et al.* (2011). In BOF-F1, we compare the set of IDs output for a document with the gold set of IDs for that document (ignoring duplicates), and utilize standard precision, recall, and F1 measures.

To illustrate this evaluation metric, consider a test document that consists of the following labeled mentions: {(“Timberlake” → `Timberlake`), (“Spears” → `Spears`), (“Timberlake” → `Timberlake`)}. Suppose that a system predicts: {(Timberlake →

`Timberlake`), (Spears $\rightarrow$ `Spears`), (Timberlake $\rightarrow$ `Timberlake,_Ohio`)}. Hence, the Bag of Freebase ids for the gold set is the set {`Timberlake`, `Spears`}. Similarly, for the predictions, we obtain the set {`Timberlake`, `Spears`, `Timberlake,_Ohio`}. The system thus achieves a precision of 0.67 and a recall of 1.0 for this document. F1 scores for each document are macro-averaged to obtain a final F1 score for the whole dataset. We separately computed exact-match accuracy, but the results were very similar to the BOF-F1 score, and we chose to report BOF-F1 as it allows for comparison with previous systems. For more details on BOT-F1, see *Ratinov et al.* (2011). Note that by construction, BOT-F1 and BOF-F1 provide identical numbers for our experiment.

Competitors: We compare NereL with 2 state-of-the-art NER systems: the current state-of-the-art NER system for the benchmark CoNLL 2003 test set, the UIUC NER system (*Ratinov and Roth* (2009)); and the Stanford NER system (*Finkel et al.* (2005)). The former uses a regularized averaged perceptron model and external gazetteers for strong performance. The latter uses Conditional Random Fields and Gibbs sampling to incorporate long-distance dependencies into its NER model; the constraints ensure that the same phrase should be labeled consistently within the same document. We downloaded these two systems and ran the configurations that were reported to perform best on the CoNLL dataset. We found the results to be almost the same as reported by their authors.

The six EL competitors for NereL include: 1) Cuc07 (*Cucerzan* (2007)), which was the first system to use collective classification of entity links. 2) MW08 (*Milne and Witten* (2008)), which uses the set of unambiguous mentions in the text surrounding a mention to define the mention's context. MW08 uses Normalized Google Distance to compute the similarity between this context and the candidate Wikipedia entry. 3) Kul09 (*Kulkarni et al.* (2009)) is an extension of both Cuc07 and MW08. It uses a hill-climbing approach to compute the joint probability distribution of all entity link assignments for a document. 4) Rat11 (*Ratinov et al.* (2011)), the current state-of-the-art wikifier, has both a local and a global component. The local model uses techniques similar to traditional EL systems (*Mihalcea and Csomai* (2007); *Bunescu and Pasca* (2006)) and similar to the Wikipedia-features used in our BaseEL model. The global component uses the predictions of the local model for all mentions in a document as the context of a mention, and they use both NGD and PMI *Turney* (2002) to compute the similarity between this context and a candidate Wikipedia page. 5) Hof11 (*Hoffart et al.* (2011)), a state-of-the-art database EL system, links mentions to the YAGO2 knowledge base. Their technique uses a dense coherence graph algorithm to measure mention-entity similarity. 6) Sil12 represents our re-implementation of the *Sil et al.* (2012a) domain-independent database EL system, which we used as a starting point for our BaseEL model. The four wikifiers and Hof11 use a collective classification framework, but only for collectively classifying sets of entity links, not for combining NER and EL. Sil12 and BaseEL are purely local models.

Parameter Settings: NereL includes two important parameters: σ, the regularization weight; and the number of candidate entities we select from BaseEL. We set the value of σ by trying five possible values in the range [0.1, 10] on held-out data. We chose 5 random documents as development data from the CoNLL-2003 training set. We found $\sigma = 0.5$ to work best for our experiments. We chose to select 3 candidate entities from BaseEL for each candidate mention (or fewer if BaseELhad fewer than 3 links with nonzero probability), as this higher thresholds led to a computationally intractable number of entity-mention tuples. As mentioned previously, using 3 candidate links per mention resulted in a maximum possible recall for NereL of 0.96 across our test sets, which is significantly higher than the maximum possible recall of 0.82 for pipeline models.

Results: Table 3.6 shows the performance of NereL on the NER task. NereL

Dataset	System	Prec	Rec	F1
ACE	UIUC	92.0	89.3	90.7
	Stanford	**97.0**	84.8	90.5
	NereL	92.0	**92.4**	**92.2**
MSNBC	UIUC	69.8	75.4	72.5
	Stanford	**86.0**	72.2	78.5
	NereL	83.7	**91.0**	**87.2**
CoNLL	UIUC	91.2	**90.5**	**90.9**
	Stanford	**95.1**	78.3	85.9
	NereL	86.8	89.5	88.2

Table 3.6: NereL outperforms two state-of-the-art NER systems on two out of three datasets, and outperforms one of them on the third dataset.

achieves the highest recall and F1 scores on two of the three datsets, and second-highest on the third dataset (CoNLL). The Stanford NER system has the best precision on all three datasets, and the UIUC system has the best overall performance on CoNLL. On MSNBC, NereL outperforms both the state-of-the-art UIUC and Stanford NER systems by 0.14 and 0.87 gains in F1 score respectively. On the other datasets, differences are smaller, but NereL consistently performs very well. We suspect that the gazetteers in the UIUC system help significantly on the CoNLL dataset. Another problem for NereL on CoNLL was the presence of several long chains of 20 or more entities, which occurred in several articles reporting the results of cricket matches. NereL tended to have much lower accuracy on these long chains of entities, which behaved very differently from sequences of entities in normal text.

Table 3.7 compares NereL with previously reported results by MW08 and Rat11 on the ACE and MSNBC datasets, as well as the results of our implementations of Sil12 and BaseEL. NereL achieves an F1 score of 85.9 on ACE and 84.6 on MSNBC, clearly outperforming all competitors. Compared with the state-of-the-art Rat11 wikifier, NereL improves by 0.086 and 0.097 F1 on ACE and MSNBC respectively. Part of this is due to better linking accuracy, probably resulting from

Systems	ACE Prec	ACE Rec	ACE F1	MSNBC Prec	MSNBC Rec	MSNBC F1
MW08*	-	-	72.8	-	-	68.5
Rat11*	-	-	77.3	-	-	74.9
Sil12*	82.1	74.7	78.2	87.6	67.1	75.9
BaseEL	84.6	77.0	80.7	**89.3**	68.4	77.5
NereL	**85.5**	**86.4**	**85.9**	85.8	**83.4**	**84.6**

Table 3.7: NereL outperforms all competitors on the ACE and MSNBC datasets. We obtain the numbers for MW08 and Rat11 from (*Ratinov et al.* (2011)), which does not report precision and recall. * indicates that the EL results are based on tests where gold-standard mentions are given as input.

	Systems			
	NereL	Hof11	Kul09	Cuc07
Prec@Rec=max	**84.22**	81.91	76.74	43.74

Table 3.8: NereL outperforms all previously-tested competitors on the CoNLL-2003 (testb) test set. Precision is shown at the highest recall achievable by each system. All competitors to NereL use only the correct mentions detected by the Stanford NER system as input, and thus their recall is limited by the recall of the Stanford NER system.

our use of Freebase's attributes, relations, and types for constructing features. Even the BaseEL and the relatively simple Sil12 system (ported to Freebase) outperform the Rat11 wikifier, despite the fact that the Rat11 wikifier is given gold-standard mentions as input, and BaseEL is given the output of the UIUC NER system.

However, another important aspect of NereL's performance is its ability to correct poor mention boundaries that propagate to errors in linking. Compared with BaseEL, NereL achieves significantly higher recall and F1 than BaseEL, similar to the differences between NereL's NER performance and the performance of the UIUC NER system that BaseEL uses in its pipeline. Overall, NereL improves by 0.052 and 0.071 F1 over BaseEL, its closest competitor, on these two benchmark datasets.

Table 3.8 illustrates the EL performance of NereL on the CoNLL dataset. In order to compare with *Hoffart et al.* (2011), we replicate their experimental evaluation: systems are evaluated according to their precision at their maximum possible recall level. Furthermore, no mention candidates with NIL links are considered; in Hoffart *et al.*'s experiments, this resulted in the removal of 20% of the mention candidates. Although not acceptable as per TAC guidelines, we did this for a fairer comparison. In Hoffart *et al.*'s evaluation, each system was supplied with the mentions from the Stanford NER system, and thus their recall was limited by the recall of the Stanford NER system. Hoffart *et al.* do not report this recall, but in our experiments we found that the Stanford system had a recall of 0.783 on the CoNLL test set, or 0.801 when excluding NIL mentions. As before, NereL uses its overgeneration techniques to generate mention candidates. Its recall on this test set is 0.826; thus NereL outperforms the next-best system, Hoff11, in both precision and recall. Clearly NereL outperforms all the other systems as well. The precision numbers for the other systems were obtained from *Hoffart et al.* (2011).

3.7 Conclusion

Much of the previous research on entity linking has gone into improving linking accuracy over gold-standard mentions, but we observe that many of the common errors made by entity linkers in practice have to do with the pipeline architecture, which propagates errors from named-entity recognition systems to the entity linkers. We propose a re-ranking model that performs joint named entity recognition and entity linking. The discriminative re-ranking framework allows us to introduce features into the model that capture the dependency between entity linking decisions and mention boundary decisions, which existing models do not handle. Furthermore, the model can handle collective classification of entity links, at least for nearby groups of entities. The joint NER and EL model has strong empirical results in our initial experiments, outperforming a number of state-of-the-art NER and EL systems on several benchmark datasets while remaining computationally inexpensive. This illustrates that many of the problems with the pipeline architecture for EL can be overcome by the re-ranking framework.

3.8 Open Research Directions

For future work we want to explore joint models for EL and coreference resolution and relation extraction. Current models for coreference resolution usually rely on a pipeline architecture using anaphoric as well as syntactic features where each candidate mention-pairs are considered independently of the other. We believe that these traditional models can be improved using re-ranking algorithms similar to NereL as described in this chapter. Similar techniques can also be extended to the task of relation extraction, where mentions in a particular document can be automatically tagged with their target Freebase links and further processing can be performed by exploring relationships stored in Freebase among them. This can further boost the performance of a relation extraction system.

CHAPTER 4

TEMPORAL SCOPING OF ENTITY RELATIONS

4.1 Introduction

In our previous chapters, we discuss some of the popular tasks in the field of Information Extraction and our contribution. We learn that traditional information extraction research mostly focus on named-entity recognition (NER), entity linking (EL) and relation extraction. In relation extraction, the task is to build a large knowledge base with relational facts between entities. Previous work (*Agichtein and Gravano* (2000); *Etzioni et al.* (2004)) on relation extraction by systems such as NeLL (*Carlson et al.* (2010)), KnowItAll (*Etzioni et al.* (2004)) and YAGO (*Suchanek et al.* (2007)) have targeted the extraction of entity tuples (*e.g.* president-Of(George W. Bush, USA)) in order to build a large knowledge base of facts. However, all these systems assume that relational facts are time-invariant. This assumption is certainly not always true *e.g.* president-Of(George W. Bush, USA) holds within the time-frame (2001-2009). In this chapter, we focus on this relatively less explored problem: attaching temporal scope to relation between entities. Recently the Text Analysis Conference (TAC) in 2013 also focused on this problem and introduced the temporal slot filling (TSF) task (*Dang and Surdeanu* (2013)). The input to a TAC-TSF system is a binary relation *e.g.* `per:spouse`(Brad Pitt, Jennifer Aniston) and a document supporting the relation. The output of the system is a 4-tuple timestamp [T1, T2, T3, T4] where T1 and T2 are normalized dates that provide a range for the start date of the relation, and T3 and T4 provide the range for the end of the relationship. A TAC-TSF system must also output the offsets of the text that supports the temporal information extracted within a document, *e.g.* *"Pitt married Jennifer Aniston on July 29, 2000... the couple divorced five years later in 2005."* The 4-tuple normalized timestamp extracted from this example text is [2000-07-29, 2000-07-29, 2005-01-01, 2005-12-31]. More details about the task are provided by *Dang and Surdeanu* (2013).

We describe TSRF, a system for temporal scoping of relational facts. For every relation type, TSRF uses distant supervision from Wikipedia infobox tuples to learn a language model consisting of patterns of entity types, categories, and word n-grams. Then it uses this trained relation-specific language model to extract the top k sentences that support the given relation between the query entity and the slot filler. In a second stage, TSRF performs timestamp classification by employing models which learn "Start", "End" and "In" predictors of entities in a relationship; it computes the best 4-tuple timestamp [T1, T2, T3, T4] based on the confidence values associated to the top sentences extracted.

Following the TAC-TSF task for 2013, we train and evaluate TSRF for the seven broad relations shown below. In each case per refers to a person and org refers to an organization.

- per:spouse
- per:title
- per:employee_or_member_of
- per:city_of_residence
- per:stateorprovince_of_residence
- per:country_of_residence
- org:top_employee/member

The `per:spouse` relation indicates the marriage relationship between entities *e.g.*

Tom Cruise and Katie Holmes. The `per:title` relation shows how entities like Barack Obama are related to a title like President. The `per:employee_or_member_of` relation shows how an entity like Carol Bartz is related to Yahoo!: she is the employee of the latter entity. As opposed to this relation, TAC treats the `org:top_employee/member` relation slightly differently: here the concern is how an organization like Microsoft is related to its top employee or CEO, Steve Ballmer. The other relations are residence relations broken down into `per:city_of_residence`, `per:stateorprovince_of_residence` and `per:country_of_residence`. For the (person) entity Bill Gates, the relations are denoted as `per:city_of_residence` (Bill Gates, Medina), `per:stateorprovince_of_residence` (Bill Gates, Washington) and `per:country_of_residence` (Bill Gates, USA).

4.2 Related Work

To our knowledge, there are only a small number of systems that have tackled the temporal scoping of relations task. YAGO (*Wang et al.* (2010)) extracts temporal facts using regular expressions in Wikipedia infoboxes, while PRAVDA (*Wang et al.* (2011)) uses a combination of textual patterns and graph-based re-ranking techniques to extract facts and their temporal scopes simultaneously. Both systems augment an existing KB with temporal facts similarly to the CoTS system by *Talukdar et al.* (2012a); *Talukdar et al.* (2012b). However, their underlying techniques are not applicable to arbitrary text. In contrast, TSRF automatically bootstraps patterns to learn relation-specific language models, which can be used then for processing any text.

CoTS is a recent system that is part of CMU's NELL (*Carlson et al.* (2010)) project, which performs temporal scoping of relational facts by using manually edited temporal order constraints. While manual ordering is appealing and can lead to high accuracy, it is impractical from a scalability perspective. Additionally, the main goal of CoTS is to predict temporal ordering of relations rather than to scope temporally individual facts. Our proposed system automatically extracts text patterns, which uses then to perform temporal classification based on gradient boosted decision trees (GBDT) *Friedman* (2001).

The TempEval task (*Pustejovsky and Verhagen* (2009)) mainly focused on temporal event ordering. Systems such as *Chambers et al.* (2007) and *Bethard and Martin* (2007) have been successful in extracting temporally related events. Another branch of research by *Sil et al.* (2010); *Sil and Yates* (2011b) has been to automatically extract STRIPS representation (*Fikes and Nilsson* (1971)) from web text *i.e.* extracting states of the world before and after an event takes place. However, all these prior work focuses on temporal ordering of either events or states of the world and does not extract timestamps for events. Our system not only extracts temporal expressions but also produces an ordering of the timestamps of relational facts between entities.

The Stanford SUTime by *Chang and Manning* (2012) is a well-known *temporal tagger* used to resolve temporal surface forms. SUTime is a rule-based temporal tagger completely built on patterns based on regular expression. Given as input some English text in tokenized format, SUTime extracts temporal expressions and predicts annotations for more enhancement. The output of the system contains annotations in the form of TIMEX3 tags (*Pustejovsky et al.* (2003)) and is used to markup times, events and their temporal relations in text. However, this system cannot attach the extracted temporal expressions to events just by itself. We use SUTime as part of our extraction tool to normalize the timestamps in web text.

The current state-of-the-art systems for TSF are the RPI-Blender system by *Artiles et al.* (2011) and the UNED system by *Garrido et al.* (2012);*Garrido et al.* (2011). These systems obtained the top scores in the 2011 TAC TSF evaluation by outperforming the other participants such as the Stanford Distant Supervision system (*Surdeanu et al.* (2011)). Similar to our work, these systems use distant supervision to assign temporal labels to relations extracted from text. While we use Wikipedia infoboxes in conjunction with Wikipedia text, the RPI-Blender and UNED systems use tuples from structured repositories like Freebase. There are major differences in terms of learning strategies of these systems. The UNED system uses a rich graph-based document-level representation to generate novel features whereas RPI-Blender uses an ensemble of classifiers combining flat features based on surface text and dependency paths with tree kernels. Our system employs language models based on Wikipedia annotated automatically with entity tags in a GBDT learning framework. A minor difference between TSRF and RPI-Blender is that the latter makes use of an additional temporal label (Start-And-End) for facts within a time range. TSRF employs Start, End, and In labels.

4.3 The Temporal Slot Filling Task

4.3.1 Input

The input to any TAC-TSF system is a binary relation between two entities and a document that supports the fact *e.g.* the `per:spouse`(Brad Pitt, Jennifer Aniston) and the document id AFP_ENG_20081208.0592 from a given collection . We explain the input format in more details in this section. As instantiated for the relation `per:spouse` between the entities Brad Pitt and Jennifer Aniston, the input format is shown in Table 4.1. The field Column 1 contains a unique query ID for the relation. Column 2 is the name of the relationship, which also encodes the type of the target entity. Column 3 contains the name of the query entity, i.e., the subject of the relation. Column 4 contains a valid document ID and Column 5 indicates the slot-filler entity. Column 6 through 8 are offsets of the slot-filler, query entity and the relationship justification in the given text. Column 9 contains a confidence score set to 1 to indicate that the relation is correct. Columns 10 and 11 contain the IDs in the KBP knowledge base of the entity and filler, respectively. All of the above are provided by TAC. For the query in this example, a TSF system has to scope temporally the `per:spouse` relation between Brad Pitt and Jennifer Aniston.

Column 1: TEMP72211	Column 7: 1492
Column 2: per:spouse	Column 8: 1311
Column 3: Brad Pitt	Column 9: 1.0
Column 4: AFP_ENG_20081208.0592	Column 10: E0566375
Column 5: Jennifer Aniston	Column 11: E0082980
Column 6: 1098	

Table 4.1: Input to a TSF System.

4.3.2 Output

Similar to the regular slot filling task in TAC, the TSF output includes the offsets for at least one entity mention and up to two temporal mentions used for the extraction and normalization of hypothesized answer. For instance, assume that a system extracts the relative timestamp "Monday" and normalizes it to "2010-10-04" using the document date from this document:

```
<DOC>
<DOCID> AFP_ENG_20101004.0053.LDC2010T13 </DOCID>
<DOCTYPE SOURCE=‘‘newswire’’> NEWS STORY </DOCTYPE>
<DATETIME> 2010-10-04 </DATETIME>
<BODY>
<HEADLINE>
Twitter co-founder steps down as CEO
</HEADLINE>
<TEXT>
<P>
Twitter co-founder Evan Williams announced on Monday
that he was stepping down as chief executive..
</DOC>
```

The system must report the offsets for both "Monday" in the text body and "2010-10-04" in the DATETIME block for the justification.

The TAC-TSF task uses the following representation for the temporal information extracted: For each relation provided in the input, TSF systems must produce a 4-tuple of dates: [T1, T2, T3, T4], which indicates that the relation is true for a period beginning at some point in time between T1 and T2 and ending at some time between T3 and T4. By convention, a hyphen in one of the positions implies a lack of a constraint. Thus [-, 20120101, 20120101, -] implies that the relation was true starting on or before January 1, 2012 and ending on or after January 1, 2012. As discussed in the TAC 2011 pilot study *Ji et al.* (2011), there are situations like recurring events that cannot be covered by this representation, such as repeated marriages between two persons. However, the most common situations for the relations covered in this task are captured correctly by this 4-tuple representation.

4.4 Challenges

In this section, we discuss some of main challenges encountered in building a temporal scoping system.

4.4.1 Lack of Annotated Data

Annotation of data for this task is expensive, as the human annotators must have extensive background knowledge and need to analyze the evidence in text and reliable knowledge resources. As per *Ji et al.* (2013), a large team of human annotators were able to generate only 1,172 training instances for 8 slots for KBP 2011. The authors of the study concluded that such amount of data is not enough for training a supervised temporal scoping system. They also noted that only 32% of employee_Of queries were found to have potential temporal arguments, and only one third of the queries could have reliable start or end dates.

4.4.2 Date Normalization

Sometimes temporal knowledge is not stated explicitly in terms of dates or timestamps. For example, from the text "they got married on Valentine's Day" a system can extract Valentine's Day as the surface form of the start of the `per:spouse` relation. However, for a temporal scoping system it needs to normalize the temporal string to the date of February 14 and the year to which the document refers to explicitly in text or implicitly (e.g., the year in which the document was published).

4.4.3 Lexico-Syntactic Variety

A relation can be specified in text by employing numerous syntactic and lexical constructions; *e.g.* for the `per:spouse` relation the patterns "got married on [DATE]" and "vowed to spend eternity on [DATE]" have the same meaning.

4.4.4 Inferred Meaning

A temporal scoping system also needs to learn the inter-dependence of relations, and how one event affects another. For instance, in our automatically generated training data, we learn that a `death` event specified by n-grams like "was assassinated" affects the `per:title` relation, and it indicates that the relationship ended at that point.

4.4.5 Pattern Trustworthiness

A temporal scoping system should also be able to model the trustworthiness of text patterns, as well as the evolution of patterns that indicate a relationship over time. For example, in current news streams, the birth to a child does not imply that a couple is married, although it does carry a strong signal about the marriage relationship.

4.5 Learning to Attach Temporal Scope

4.5.1 Automatically Generating Training Data

As outlined in Section 4.4, one of the biggest challenges of a temporal scoping system is the lack of annotated data to create a strong information extraction system. As per previous work on relation extraction such as *Mintz et al.* (2009) has shown distant supervision to be highly effective to build a large supervised classifier. Similar to supervisied classfication techniques, some advantages of using distant supervision are:

- Allows to build a classifier with lots of features
- Supervised by the detailed user-contributed knowledge
- No need to expand patterns iteratively

Mintz et al. (2009) also point out that similar to unsupervised systems, distant supervision allows:

- Using large amounts of unlabeled data like the web or social media
- Techniques which are not sensitive to genre issues present in the training data

We follow *Cucerzan* (2007); *Weld et al.* (2009); *Cucerzan* (2012) as they start from the premise that richness of the Wikipedia collection in its current form, whether semantic, lexical, syntactic, or structural, is a key enabler in redefining the state-of-the-art for many NLP and IR task. Our target is to use distant supervision from Wikipedia data to build an automatic temporal scoping system. However, for most relations in Wikipedia, we find that it does not indicate specific start or end dates. In addition to this, we need our system to be able to predict whether two entities are currently in a relationship or not based on the document date as well. Hence, in our first step, we build an automatic system which would take as input a binary relation between two entities *e.g.* `per:spouse`(Brad Pitt, Jennifer Aniston) and a number of documents. The system should be able to extract the highly ranked/relevant sentences which indicates that the two entities are in the concerned relationship. In the next step, we build another system which takes as input the top k sentences generated in the previous step and extracts temporal labels for the input relation. Note that our target is to develop algorithms which are going to be able to transferable across relations. We elaborate on these two system components further.

4.5.1.1 Using Wikipedia as a Resource for Distant Supervision

Wikipedia is the largest freely available encyclopedic collection, which is built and organized as a user-contributed knowledge base (KB) of entities. The current version of the English Wikipedia contains information about 4.2 million entities. In addition to the plain text about these entities, Wikipedia also contains structured components. One of these is the *infobox*. Infoboxes contain information about a large number of relations for the target entity of the Wikipedia page, *e.g.* names of spouses, birth and death dates, residence *etc.*. Similar to structured databases, the infoboxes contain the most important/useful relations in which entities take part, while the text of Wikipedia pages contains mentions and descriptions of these relations. Because of this, Wikipedia can be seen as a knowledge repository that contains parallel structured and unstructured information about entities, and therefore, can be employed more easily than Freebase or other structured databases for building a relation extraction system. Figure 4.1 shows how sentences from Wikipedia can be used to train a system for the temporal slot filling task.

4.5.1.2 Extracting Relevant Sentences

For every relation, we extract slot-filler names from Infoboxes of each Wikipedia article. We also leverage Wikipedia's rich interlinking model to automatically retrieve labeled entity mentions in text. Because the format of the text values provided by different users for the infobox attributes can vary greatly, we rely on regular expressions to extract slot-filler names from the infoboxes. For every relation targeted, we build a large set of regular expressions to extract entity names and filter out noise *e.g.* html tags, redundant text *etc.*.

To extract all occurrences of named-entities in the Wikipedia text, we relabel each Wikipedia article with Wikipedia interlinks by employing an implementation of the entity linking (EL) system by *Cucerzan* (2012), which obtained the top scores for the EL task in successive TAC evaluations. This implementation takes into account and preserves the interlinks created by the Wikipedia contributors, and extracts all other entity mentions and links them to Wikipedia pages if possible or hypothesizes coreference chains for the mentions of entities that are not in Wikipedia. The latter are extremely important when the slot-filler for a relation is an entity that does not have a Wikipedia page, as often is the case with spouses or other family members of famous people.

As stated in Section 4.4, temporal information in text is specified in various surface forms. As stated in Section 4.4, temporal information in text is specified in various forms. To resolve temporal mentions, we use the Stanford SUTime *Chang and Manning* (2012) *temporal tagger*. SUTime is a rule-based temporal tagger that employs regular expression. Its input is English text in tokenized format; its output contains annotations in the form of TIMEX3 tags. TIMEX3 is a part of the TimeML annotation language as introduced by *Pustejovsky et al.* (2003) and is used to markup date and time, events, and their temporal relations in text. When processing Web text, we encounter date expressions that contain a relative time *e.g.* "last Thursday". To resolve them to actual dates/time is a non-trivial task. However, the heuristic of employing the document's publication date as the reference works very well in practice *e.g.* for a document published on 2011-07-05, SUTime resolves "last Thursday" to 2011-06-30. It provides temporal tags in the following labels: Time, Duration, Set and Interval. For our experiments we used Time and Duration.

After running the Stanford SUTime, which automatically converts date expressions to their normalized form, we collect sets of contiguous sentences from the page that contain one mention of the targeted entity and one mention of the slot-filler, as extracted by the entity linking system. We then build a large language model by bootstrapping textual patterns supporting the relations, similar to *Agichtein and Gravano* (2000). The general intuition is that a set of sentences that mention the two entities are likely to state something about relationships in which they are.

Let every input document (or a Wikipedia page) be represented as D with n sentences $S = s_1, s_2, \ldots, s_n$. We will represent each sentence as a feature vector $x = \{x_1, x_2, ..., x_n\}$ and $(x_i \in R^d)$ with the target variable $(0 < y_i < 1)$. Assume that we have in total N datapoints collected automatically from millions of Wikipedia articles. The relationship under consideration between query_entity q and slot_filler z is denoted as $r_i \in \{1, \ldots, 7\}$ where $r_i = 1$ indicates *e.g.* the per:spouse relation. The label y_i indicates as to what degree sentence x_i is relevant to relation r_i and ranges from $y_i = 0$ ("irrelevant") to $y_i = 1$ ("perfect match"). We assume that a sentence vector x_i is a d dimensional feature vector that incorporates all the sufficient statistics about the relation and the sentence as features. For example, for the per:spouse relation one feature is "does the input sentence contain the n-gram "QUERY_ENTITY got married"". Both q and z can have various surface forms which we resolve to their canonical target form using the state-of-the-art EL system. Our first task is to extract relevant sentences $\tilde{s} \in S$ such that $\tilde{s}$ specifies q and z being in relation r. Notice that we build different models for different r; however the learning process of the algorithm is the same.

For instance, we were able to extract 61,872 tuples of query entity and slot filler relations from Wikipedia for the `per:spouse` relation. Figure **??** shows how we extract

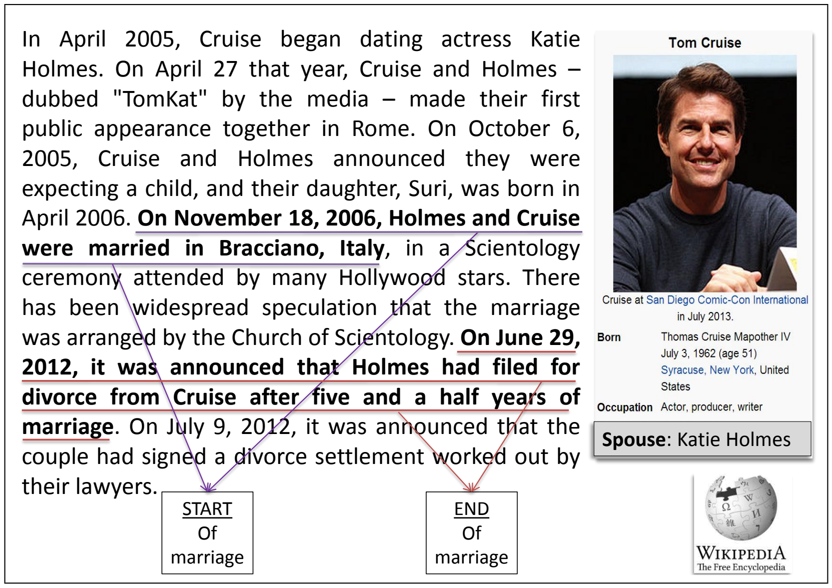

Figure 4.1: Extracting relevant sentences using query entity and slot-filler names from Wikipedia for the `per:spouse` relation.

relevant sentences using slot-filler names from Wikipedia. Consider the following text (already processed by our EL system and Stanford SUTime) taken from the Wikipedia page of Tom Cruise:

> On [November 18, 2006$|_{2006-11-18}$], [Holmes$|_{Katie_Holmes}$] and [Cruise$|_{Tom_Cruise}$] were married in [Bracciano$|_{Bracciano}$] . . .
>
> On [June 29, 2012$|_{2012-06-29}$], [Holmes$|_{Katie_Holmes}$] filed for divorce from [Cruise$|_{Tom_Cruise}$] after five and a half years of marriage.

Considering Tom Cruise as the query entity and his wife Katie Holmes as the slot filler for the `per:spouse` relation, we normalize the above text to the following form to extract features:

> On _DATE, SLOT_FILLER and QUERY_ENTITY were married in _LOCATION . . .
>
> On _DATE, SLOT_FILLER filed for divorce from QUERY_ENTITY after five and a half years of marriage.

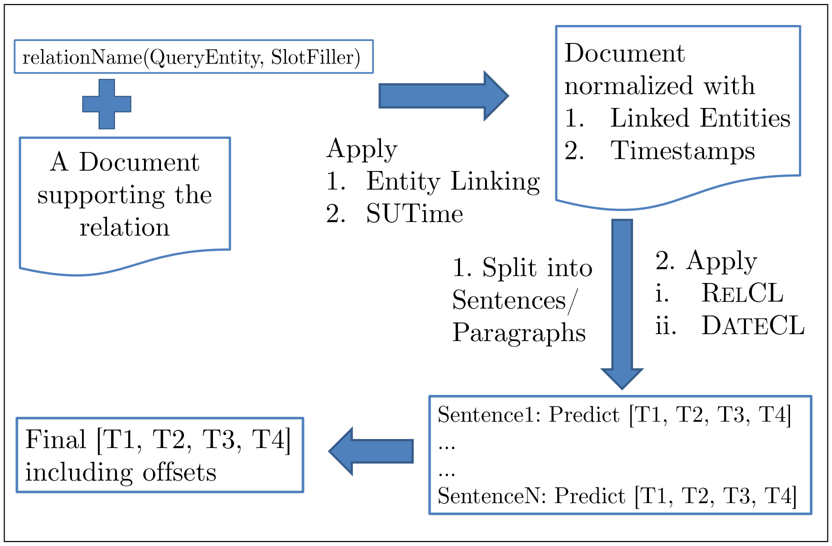

Figure 4.2: Architecture of our system. Every input document is processed by the Cucerzan EL system and the Stanford SUTime system. Temporal information is then extracted automatically using RelCL and DateCL.

Our language model consists of n-grams ($n \leq 5$) like "SLOT_FILLER and QUERY_ENTITY were married", "SLOT_FILLER filed for divorce from" which provides clues for the marriage relation. These n-grams are then used as features with an implementation of a gradient boosted decision trees classifier similar to that described by *Burges* (2010). We also use features provided by the EL system which are based on entity types and categories. We call this "relationship" classifier RelCL. The output of this step is a ranked list of sentences which indicate whether there exists a relationship between the query entity and the slot filler.

4.5.1.3 Our Learning Algorithm

In this section we discuss the learning algorithm used to train TsrF. Recall that our objective is to rank the sentences in a document based on the premise that whether the entities q and z are in relation r_i. We know that regression can be used to solve ranking problems even when they are fundamentally different problems. For instance, in our previous work such as *Sil and Yates* (2011a) and *Sil and Yates*

(2011b) we have used SVMs for ranking event preconditions and in *Sil et al.* (2012b) for ranking answers to questions. *Cucerzan* (2012) have used logistic regression for ranking entities. We explored the sentence-ranking problem using Gradient Boosted Decision Trees (GBDT) to learn temporal scope for entity relations. GBDTs can achieve high accuracy as they can easily combine features of different scale and missing values. In our experiments, GBDTs have outperformed both SVMs and MaxEnt models.

Let $F(x)$ be a d-dimensional function for our sentences represented in terms of features. In our scenario we want to solve $\min_{x \in \mathcal{R}^d} F(x)$. Note that in gradient descent we have to start with an initial guess x_0 of a minimizer. Then at each step we compute the gradient of the objective function F at the current iterate x_k, denoted by $\nabla F(x_k)$. To obtain the next iterate we use the negative gradient as the search direction as follows:

$$x_{k+1} = x_k - \alpha_k \nabla F(x_k)$$

Here, α_k is the step size which is chosen by a line-search (exploring other techniques to choose α_k is left as part of future work).

We are going to use regression for the purpose of ranking relevant sentences, hence we assume that we are given a training set $(x_i, y_i)_{i=1}^{N}$, and we seek to find a function h such that $y_i \approx h(x_i); i = 1, \ldots, N$. For our task, we use the square loss function (although other functions such as the negative binomial log-likelihood can also be used), *i.e.*, we compute the difference between y_i and $h(x_i)$ denoted by $(y_i - h(x_i))^2$. Our objective then is to find a function $h(x)$ to solve the following minimization problem,

$$\min_{h \in \mathcal{H}} L(h) \equiv \min_{h \in \mathcal{H}} \frac{1}{2} \sum_{i=1}^{N} (y_i - h(x_i))^2$$

Here, $\mathcal{H}$ is a pre-defined function class similar to the class of polynomials that do not exceed a certain degree. To minimize the functional $L(h)$, we can perform gradient descent in the function space. Hence, at the current iterate $h_k(x)$ we compute the gradient of $L(h)$ with respect to h and form the next iterate as follows:

$$h_{k+1}(x) = h_k(x) - \alpha_k \nabla L(h_k(x)).$$

The solution to the above problem is non-trivial as we can not compute $\nabla L(h_k(x))$ for all x. Hence, we can only compute it at a finite sample as shown in:

$$\nabla L(h_k(x_i)) = -(y_i - h_k(x_i)), \quad i = 1, \ldots, N.$$

The main objective of functional gradient descent is to obtain an approximation of the negative gradient $-\nabla L(h_k(x))$ to form the next iterate by computing a function that interpolates the above sample values. We fit a regression tree to the sample values for the approximation task as shown in Figure 4.3. Using a smaller regression tree at each step, the model computes residuals obtained in the previous step. Since we use the stochastic variant of GBDT, hence for computing the loss function, the model absorbs several samples instead of using the whole training data. This also prevents over-fitting. For more details of this algorithm we direct the readers to *Friedman* (2001); *Burges* (2010).

We use a development set (selected from our Wikipedia dump) which is independent from the training set to choose the optimal parameter settings for our GBDT model. These parameters include M, the number of regression trees and η, the shrinkage factor.

4.5.1.4 Predicting Relevant Sentences

On our unseen test data, we apply our trained model and obtain a score for each new sentence s with entities q and z as a triple (s, q, z) in a concerned relationship. Note that each s is turned into a feature vector as shown in the previous sections.

Algorithm: GBDT for TsrF

Input: i. Training data $(x_i, y_i)_{i=1}^{N}$
ii. The number of regression trees M
iii. η, the shrinkage factor
Initialization: $h_0(x) = \sum_{i=1}^{N} \frac{y_i}{N}$
For $k = 1$ to M
(a) For i = 1 to N
Compute the negative gradient
$r_{ik} = y_i - h_{k-1}(x_i)$
(b) Fit a J-node regression tree to $\{r_{ik}\}_{i=1,\ldots,N}$
giving terminal regions $R_{jk}, j = 1, \ldots, J_k$.
(c) For $j = 1, \ldots, J_k$, compute
$\gamma_{jk} = \sum_{x_i \in R_{jk}} (y_i - h(x_i))/|\{i : x_i \in R_{jk}\}|$,
average of the residual in each terminal region.
(d) Update:
$h_k(x) = h_{k-1}(x) + \eta(\sum_{j=1}^{J_k} \gamma_{jk} I(x \in R_{jk}))$,
where η is the shrinkage factor, and $I(\cdot)$ is the indicator function.
End for

Figure 4.3: Our learning algorithm using GBDTs.

Among all the triples having q and z, we choose the top k triples having the highest score denoted by $(\tilde{s}, q, z)$. We choose k based on the performance of TsrF on our development set.

4.5.1.5 Extracting Timestamps

To predict the timestamps for each relation, we build another classifier, DateCL (similar to that described in the previous section) using language models for "Start", "End" and "In" predictors of relationship. The "Start" model predicts T1, T2; "End" predicts T3, T4 and "In" predicts T2, T3.

Raw Trigger Features: Similar to previous work by *Sil et al.* (2010); *Sil and Yates* (2011b) on using discriminator words as features, each of these models compose of "Trigger Words" that indicate when a relationship began or ended. These triggers are chosen by domain experts from the language model automatically bootstrapped from Wikipedia. Future directions include how to automatically learn these triggers. For example, for the `per:spouse` relation, the triggers for "Start" would contain n-grams like "married since _DATE" and "married SLOT_FILLER on"; the "End" model would have n-grams like "estranged husband QUERY_ENTITY", "split in _DATE", and finally "In" consists of "happily married", "QUERY_ENTITY with his wife" *etc.*. For a input sentence with query entity q and slot-filler z, our first class of raw trigger features consists of **cosine-similarity**(Text(q, z), Triggers(r)) where $r \in Start, End, In$. Here, Text(q, z) indicates the full sentence as context. We also have another feature that computes **cosine-similarity**(Context(q, z), Triggers(r)) which constructs a "mini-sentence" Context(q, z) from the original by choosing 3 words before and after the query entity and slot-filler ignoring duplicates.

External Event Triggers: Our system also considers the presence of other events as triggers *e.g.* a "death" event signaled by "SLOT_FILLER died" might imply that a relationship ended on that timestamp. Similarly, a birth event can imply that an entity started living in a particular location *e.g.* the `per:born-In`(Obama, Honolulu) relation from the sentence "President Obama was born in Honolulu in 1961" indicates that T1 = 1961-01-01 and T2 = 1961-12-31.

At each step, TsrF extracts the top timestamps for predicting "Start", "End" and "In" based on the confidence values of DateCL.

Similar to previous work by *Artiles et al.* (2011), we aggregate and update the extracted timestamps using the following heuristics:

Step 1: Initialize T= [-∞, +∞, -∞,+ ∞]

Step 2: Iterate through the classified timestamps

Step 3: For a new T' aggregate :

$$T\&\&T' = [max(t_1, t_1'), min(t_2, t_2'), max(t_3, t_3'), min(t_4, t_4')]$$

Update only if: $t_1 \leq t_2$; $t_3 \leq t_4$; $t_1 \leq t_4$

This novel two-step classification strategy removes noise introduced by distant supervision training and decides if the extracted (entity, filler, timestamp) tuples belong to the relation under consideration or not. For example, for the `per:spouse` relation between the entities Brad Pitt and Jennifer Aniston, TsrF extracts sentences like "..*On November 22, 2001, Pitt made a guest appearance in the television series Friends, playing a man with a grudge against Rachel Green, played by Jennifer Aniston..*" and "*Pitt met Jennifer Aniston in 1998 and married her in a private wedding ceremony in Malibu on July 29, 2000..*". The system automatically discards extracting temporal information from the former sentence even though the sentence clearly specifies both the two entities. Obviously, it is able to realize based on the language model for the marriage relation that the text is talking about the two entities being in another relation: "acted-In" which indicates entities acting/appearing in the same television show. The second sentence is a positive example for the marriage relation and TsrF extracts the temporal scope July 29, 2000 and attaches the START label to it. Most previous systems do not perform this noise removal step which is a critical component evident in our empirical evaluation as we show in the next section.

4.6 Experiments

We evaluate the performance of TsrF against 2 state-of-the-art systems for the TSF task as well as several other temporal scoping systems available in the literature.

Datasets: For evaluation, we train our system on the infobox tuples and sentences extracted from the Wikipedia dump of May 2013. We set aside a portion of the dump as our development data. We chose to use the top-relevant n-grams based on the performance on the development data as features. We employ then the TAC evaluation data, which is publicly available through LDC.

Evaluation Metric: We utilize the evaluation metric developed for TAC (*Dang and Surdeanu* (2013)). In order for a temporal constraint (T1-T4) to be valid, the document must justify both the query relation (which is similar to the regular English slot filling task) and the temporal constraint. Since the time information provided in text may be approximate, the TAC metric measures the similarity of each constraint in the key and system response. Formally, if the date in the gold standard is k_i, while the date hypothesized by the system is r_i, and $d_i = |k_i - r_i|$ is their difference measured in years, then the score for the set of temporal constraints on a slot is computed as:

$$S(slot) = \frac{1}{4}\sum_{i=1}^{4}\frac{c}{c+d_i} \tag{4.6-1}$$

$$c = \begin{cases} c_{overconstrained} \\ c_{vagueness} \end{cases}$$

Here, $c_{overconstrained}$ when (i $\in$ {1,3} AND $r_i > k_i$) OR (i $\in$ {2,4} AND $r_i < k_i$) and $c_{vagueness}$, otherwise.TAC sets the constant c to one year, so that predictions that differ from the gold standard by one year get 50% credit. The absence of a constraint in T1 or T3 is treated as a value of $-\infty$ and the absence of a constraint in T2 or T4 is treated as $+\infty$, which lead to zero-value terms in the scoring sum. Therefore, the overall achievable score has a range between 0 and 1. For more details on the scoring, see *Dang and Surdeanu* (2013).

Competitiors: We compare TsrF against several state-of-the-art systems from the literature for the TSF task: **(i)** RPI-Blender (*Artiles et al.* (2011)), **(ii)** CMU-NeLL (*Talukdar et al.* (2012a);*Talukdar et al.* (2012b)), **(iii)** UNED (*Garrido et al.* (2012);*Garrido et al.* (2011)) and **(iv)** Abby-Compreno (*Kozlova et al.* (2012)). We note that comparison with all temporal systems in the literature is beyond the scope of this dissertation. RPI-Blender and UNED obtained the top scores in the 2011 TAC TSF evaluation and thus, could be considered as the state-of-the-art at the time. Most of these systems use distant supervision to assign temporal labels to tuples of (entity, filler, time) extracted from text. The RPI-Blender and UNED systems use training tuples from Freebase as opposed to TsrF which uses Wikipedia infoboxes. Similar

	S1	S2	S3	S4	S5	S6	S7	**ALL**	StDev
Baseline	24.70	17.40	15.18	17.83	14.75	21.08	23.20	19.10	3.60
TsrF	**31.94**	**36.06**	**32.85**	**40.12**	**33.04**	**31.85**	27.35	**33.15**	3.66
RPI-Blender	31.19	13.07	14.93	26.71	29.04	17.24	**34.68**	23.42	7.98
UNED	26.20	6.88	8.16	15.24	14.47	14.41	19.34	14.79	6.07
CMU-NeLL	19.95	7.46	8.47	16.52	13.43	5.65	11.95	11.53	4.77
Abby-Compreno	0.0	2.42	8.56	0.0	13.50	7.91	0.0	5.14	4.99
LDC	69.87	60.22	58.26	72.27	81.10	54.07	91.18	68.84	12.32

Table 4.2: Results for the TAC-TSF 2013 test set, overall and for individual slots. The slots notation is: S1: org:top members employees, S2: per:city of residence, S3: per:country of residence, S4: per:employee or member of, S5: per:spouse, S6: per:statesorprovince of residence, S7: per:title. The score for the output created by the LDC experts is also shown.

to TsrF, most systems use Start, End, and In labels, with RPI-Blender adding an additional one (Start-And-End) for facts within a time range between two dates in one sentence. RPI-Blender uses an ensemble of classifiers combining flat features (surface text, dependency paths) with tree kernels. The UNED system uses a rich graph-based document-level representation to generate novel features. The CMU-NeLL system relies on a joint inference framework which leverages fact-specific temporal constraints, and weak supervision in the form of a few seed examples. We also compare our system with a reasonable baseline similar to *Ji et al.* (2011). This baseline makes the simple assumption that the corresponding relation is valid at the document date. That is, it creates a "within" tuple as follows: $< -\infty$, doc_date, doc_date, $+\infty >$. Hence, this baseline system for a particular relation always predicts T2=T3= Date of the Document. For instance, consider for the input relation, `per:spouse`(George Bush, Laura Bush) and the text: "*President Bush and his wife Laura Bush paid a visit to a non-governmental organization..*" , the baseline system extracts T2=T3= 2007-03-09 (date of the document).

Results: Table 4.2 lists our TSF results, overall and for each individual slot. We also compare the system outputs against an output generated by human experts (LDC). Note that, even though the evaluation dataset contained 273 queries, only

201 were actually scored: 5 queries were dropped because neither the systems' nor LDC's output contained correct slot fillers. The other 67 queries are not considered as the gold temporal annotations produces a gold standard tuple that has an invalid temporal interval because a temporal interval is valid only if $T1 \leq T2, T3 \leq T4$ and $T1 \leq T4$. Also note that these scores currently ignore the textual justification generated by the systems, i.e., we score TSF outputs solely based on the formula introduced in Eq. 4.6-1. Implementing a metric for evaluating temporal justifications is left as future work by TAC.

We observe that TSRF outperforms all the other (non-human) systems in overall score and all individual slots except per:title. In fact, TSRF outperforms the next best system RPI-Blender by 10 points and UNED by 19 points. Both these systems obtained the top score in TAC 2011 outperforming other systems like the Stanford system (*Surdeanu et al.* (2011)). Interestingly, TSRF also outperforms CMU-NeLL (by 21 points) which is part of the NeLL system and contains a huge KB of relational facts already extracted from the web and uses web corpus like the Google N-gram [1] corpus. We believe that this performance boost is partially due to the fact that TSRF uses a language model to clean up the noise introduced by distant supervision before the actual temporal classification step. Recall that this language model learned that n-grams such as "FILLER and ENTITY were married" which indicates that the entities in the `per:spouse` relation. Another reason might be that our learning algorithm (GBDT) is highly effective to address the TSF task as the extracted n-grams are used as features in a gradient boosted decision tree classifier that decides if the extracted (entity, filler, time) tuples belong to the relation under consideration or not. Clearly, this noise removal step appears to bring a significant difference between TSRF and the second-best system RPI-Blender in Table 4.2 and indicates the importance of noise removal for the task of TSF.

[1] http://books.google.com/ngrams/datasets

We also see that only two out of the five systems, TsrF and RPI-Blender outperform the "within" baseline. It is also important to note that our system exhibits a balanced performance on the relations on which it was tested. As shown in column StDev in Table 4.2, our systems achieves the lowest standard deviation in the performance across the relations tested. It is interesting to note also that TsrF achieves the best performance on the `employee_Of` and `residence_Of` (cities) relations even though the system development was done on the *spouse* relation as an encouraging sign that our distant supervision algorithm can be transferred successfully across relations for domain-specific temporal scoping. For the other TSF systems we observe a visible pattern: the slots corresponding to locations of residence like `per:cities_of_residence`, `per:countries_of_residence` and `per:statesorprovinces_of_residence` perform generally worse than average. The slots that correspond to employment relations like `org:top_members_employees` and `per:employee_or_member_of`) perform better than average. This indicate that (at least, in this dataset) attaching temporal scope for residence relations is a harder task than for the employment relations.

Another point to note is that the top performer, TsrF achieves approximately 48% of human performance (LDC) showing its ability to automatically learn temporal scoping of facts. Compared to this, the median systems are at 21% of human performance proving that TSF is a difficult task and suggests that there exists more room for future improvement.

Finally, Table 4.3 shows another reason that gives TsrF an edge in obtaining the best score. The *Cucerzan* (2012) system, which we employed, is a state-of-the-art system for extracting and linking entities and resolving coreference chains. By using this system, we have been able to extract slot-filler mentions with a precision of 96.8% at 66.4% recall, which is substantially higher than the extraction results of all other systems. Encouragingly, the performance of this component also comes close to that of the LDC annotators, which obtained a precision of 97.3% at 72.5% recall.

	TSF Accuracy	SF F1	SF Prec	SF Recall
LDC	68.8	83.1	97.3	72.5
TsrF	**33.1**	**77.3**	**96.8**	**64.4**
RPI-Blender	23.4	51.8	69.2	41.4
UNED	14.8	46.6	69.9	35.0
CMU-NeLL	11.5	32.2	38.5	27.6
Abby-Compreno	5.1	18.5	53.6	11.2

Table 4.3: The above results show the extraction accuracy of slot-filler mentions. The results indicate that identifying correct mentions of fillers is a challenge where TsrF clearly outperforms all non-human systems.

4.7 Conclusion and Future Work

This chapter describes TsrF an automatic temporal scoping system that requires no manual labeling effort. The system uses distant supervision from Wikipedia to obtain a large training set of tuples for training. It uses a novel two-step classification to remove the noise introduced by the distant supervision training. The same algorithm was employed for multiple relations and exhibited similarly high accuracy. Experimentally, the system outperforms by a large margin several other systems that addresses this relatively less explored problem. Future directions of development include extracting joint slot filler names and temporal information, and exploring the changes over time in Wikipedia observed for a query entity and a slot filler in the targeted relation. For this temporal scoping system, we have used Wikipedia as it offers both structured components with the infobox relations and parallel text that support those relations. In the future, we also would like to explore other structured resources like Freebase, which has significantly more entities than Wikipedia and also make use of Web corpus apart from Wikipedia text.

CHAPTER 5

CONCLUSION

This dissertation focuses on the use of structured and semi-semistructured resources for the task of information extraction. Specifically we made use of the largest available structured repository for movies: IMDB, a database for sports: Yahoo! sports used for a commercial search engine and Freebase, which is the largest freely available user-contributed structured knowledge base. We also used Wikipedia, which is the biggest semi-structured encyclopedia. Using these repositories, we have introduced state-of-the-art named-entity recognition and linking systems. We have also developed a system which performs temporal scoping of relations.

We analyzed the limitation of current Wikipedia-based Entity Linking systems and introduced the task of Open-DB Entity Linking. In general, we presented two distinct strategies for solving this task: one based on domain adaptation and the other on distant supervision which significantly outperforms a state-of-the-art Wikipedia EL system used by a commercial search engine, on a dataset where Wikipedia has good coverage and on another dataset where Wikipedia has poor coverage. The results indicate that there is a significant benefit to leveraging other sources of knowledge in addition to Wikipedia, and that it is possible to leverage this knowledge without requiring labeled data for each new source. The initial success of these Open-DB EL approaches indicates that this task is a promising area for future research, including exciting extensions that link large numbers of domain-specific databases to text. For example, one could link a large number of "tail" entities (*e.g.* person-names) to public profile databases like LinkedIn or Facebook. Also, using information from multiple databases might require performing schema-matching which is a promising area of future research.

We have also shown how most previous research on entity linking suffer because of their pipeline architecture, which propagates errors from named-entity recognition systems to the entity linkers. In response, we introduced NereL a re-ranking model that performs joint named entity recognition and entity linking. The NereL model uses novel features that capture the dependency between entity linking decisions and mention boundary decisions, which existing EL models do not handle. Furthermore, the model can handle collective classification of entity links, at least for nearby groups of entities. The joint NER and EL model has strong empirical results in our initial experiments, outperforming a number of state-of-the-art NER and EL systems on several benchmark datasets. This illustrates that many of the problems with the pipeline architecture for EL can be overcome by the re-ranking framework.

For future work we want to look at performing relation extraction and entity linking jointly. Most recent work such as *Singh et al.* (2013) concentrate on extracting relations and entities jointly but the relations are predefined as in the Automatic Content Extraction (ACE) 2004 task. However, we would like our work to be able to extract newer relations not present in knowledge bases like Freebase and which are automatically discovered from un-structured text like the web. Contrary to *Singh et al.* (2013), we want to explore re-ranking frameworks for that task as well, given its success in our joint NER and EL research. We would also like to perform joint visual and textual entity linking. We observe that in very short social media data like comments in Flickr data often contain their associated image. In such short text with no available context it would be extremely helpful to extract information from the associated image and text and match with the possible disambiguations in existing KBs like Freebase which also contain images for a large number of entities.

Finally, we also introduced TsrF an automatic temporal scoping system, which requires no manual labeling effort and uses distant supervision from Wikipedia to obtain a huge training set of tuples to build a supervised classifier. The same algorithm can be transferred across multiple relations while maintaining high accuracy. Also, TsrF uses a novel two-step classification to remove the noise introduced by the distant supervision training. Experimentally we have been able to show that TsrF outperforms several systems from the literature that addresses this relatively less explored problem. Future directions include extracting joint slot filler names and temporal information and exploring Wikipedia track changes between a query entity and a slot filler relation. Also, in our future work we want to explore information extraction from noisy social media data like Twitter data. Both NER and EL for Twitter are a relatively challenging domain of research and the available systems have a long way to go. Using less restrictive features like capitalization, resolving hashtags and spelling correction systems would be the correct path to take to solve the next generation of information extraction research. We also want to explore more diverse types of relations which our current system does not cover.

BIBLIOGRAPHY

Agichtein, E., and L. Gravano (2000), Snowball: Extracting relations from large plain-text collections, in *Procs. of the Fifth ACM International Conference on Digital Libraries*.

Alonso, O., M. Gertz, and R. Baeza-Yates (2007), On the value of temporal information in information retrieval, in *ACM SIGIR Forum*, vol. 41, pp. 35–41, ACM.

Artiles, J., J. Gonzalo, and S. Sekine (2009), Weps 2 evaluation campaign: overview of the web people search clustering task, in *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, p. 9.

Artiles, J., Q. Li, T. Cassidy, S. Tamang, and H. Ji (2011), CUNY BLENDER TACKBP2011 Temporal Slot Filling System Description, in *TAC*.

Bellare, K., and A. McCallum (2007), Learning extractors from unlabeled text using relevant databases, in *Sixth International Workshop on Information Integration on the Web*.

Bellare, K., and A. McCallum (2009), Generalized Expectation Criteria for Bootstrapping Extractors using Record-Text Alignment, in *Empirical Methods in Natural Language Processing (EMNLP-09)*.

Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan (2010), A theory of learning from different domains, *Machine Learning*, *79*, 151–175.

Bethard, S., and J. H. Martin (2007), Cu-tmp: Temporal relation classification using syntactic and semantic features, in *Proceedings of the 4th International Workshop on Semantic Evaluations*, pp. 129–132, Association for Computational Linguistics.

Bunescu, R., and R. Mooney (2007), Learning to extract relations from the web using minimal supervision, in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

Bunescu, R., and M. Pasca (2006), Using encyclopedic knowledge for named entity disambiguation, in *EACL*.

Burges, C. (2010), From ranknet to lambdarank to lambdamart: An overview, *Learning*, *11*, 23–581.

Carlson, A., J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell (2010), Toward an architecture for never-ending language learning., in *AAAI*.

Chambers, N., S. Wang, and D. Jurafsky (2007), Classifying temporal relations between events, in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pp. 173–176, Association for Computational Linguistics.

Chang, A. X., and C. Manning (2012), Sutime: A library for recognizing and normalizing time expressions., in *LREC*, pp. 3735–3740.

Chen, Y., and J. Martin (2007), Towards Robust Unsupervised Personal Name Disambiguation, in *EMNLP*, pp. 190–198.

Chinchor, N. (1998), Overview of muc-7/met-2.

Cilibrasi, R., and P. Vitanyi (2007), The google similarity distance, *IEEE Transactions on Knowledge and Data Engineering*, *19*(3), 370–383.

Cucerzan, S. (2007), Large-scale named entity disambiguation based on wikipedia data, in *EMNLP-CoNLL*, pp. 708–716.

Cucerzan, S. (2012), The MSR System for Entity Linking at TAC 2012, in *TAC*.

Cucerzan, S., and A. Sil (2013), The MSR Systems for Entity Linking and Temporal Slot Filling at TAC 2013, in *Proceedings of the Text Analysis Conference (TAC)*.

Dalvi, N. N., R. Kumar, B. Pang, and A. Tomkins (2009), Matching Reviews to Objects using a Language Model, in *EMNLP*, pp. 609–618.

Dalvi, N. N., R. Kumar, and B. Pang (2012), Object matching in tweets with spatial models, in *WSDM*, pp. 43–52.

Dang, H. T., and M. Surdeanu (2013), Task description for knowledge-base population at TAC 2013, in *TAC*.

Daumé III, H., A. Kumar, and A. Saha (2010), Frustratingly easy semi-supervised domain adaptation, in *Proceedings of the ACL Workshop on Domain Adaptation (DANLP)*.

Davis, A., A. Veloso, A. S. da Silva, W. Meira Jr, and A. H. Laender (2012), Named entity disambiguation in streaming data, in *ACL*.

Doddington, G. R., A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel (2004), The automatic content extraction (ace) program-tasks, data, and evaluation., in *LREC*.

Downey, D., M. Broadhead, and O. Etzioni (2007), Locating complex named entities in web text, in *Procs. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*.

Etzioni, O., M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates (2004), Web-Scale Information Extraction in KnowItAll, in *WWW*, pp. 100–110, New York City, New York.

Fader, A., S. Soderland, and O. Etzioni (2009), Scaling wikipedia-based named entity disambiguation to arbitrary web text, in *WikiAI 09 - IJCAI Workshop*.

Ferragina, P., and U. Scaiella (2010), Tagme: on-the-fly annotation of short text fragments (by wikipedia entities), in *CIKM*.

Fikes, R., and N. Nilsson (1971), STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, *2*(3/4), 189–208.

Finkel, J. R., T. Grenager, and C. D. Manning (2005), Incorporating non-local information into information extraction systems by gibbs sampling, in *ACL*.

Friedman, J. H. (2001), Greedy function approximation: a gradient boosting machine, *Annals of Statistics*, pp. 1189–1232.

Garrido, G., B. Cabaleiro, A. Penas, A. Rodrigo, and D. Spina (2011), A distant supervised learning system for the tac-kbp slot filling and temporal slot filling tasks, in *TAC*.

Garrido, G., A. Penas, B. Cabaleiro, and A. Rodrigo (2012), Temporally anchored relation extraction, in *ACL*.

Grishman, R., and B. Sundheim (1996), Message understanding conference-6: A brief history., in *COLING*, vol. 96, pp. 466–471.

Guo, S., M.-W. Chang, and E. Kıcıman (2013), To link or not to link? a study on end-to-end tweet entity linking, in *NAACL*.

Han, X., and J. Zhao (2009), Named entity disambiguation by leveraging Wikipedia semantic knowledge, in *CIKM*, pp. 215–224.

Han, X., L. Sun, and J. Zhao (2011), Collective entity linking in web text: a graph-based method, in *SIGIR*.

Hoffart, J., M. A. Yosef, I. Bordino, H. Furstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum1 (2011), Robust Disambiguation of Named Entities in Text, in *EMNLP*, pp. 782–792.

Hoffmann, R., C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld (2011), Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations, in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ji, H., R. Grishman, and H. T. Dang (2011), Overview of the tac2011 knowledge base population track, in *TAC*.

Ji, H., T. Cassidy, Q. Li, and S. Tamang (2013), Tackling representation, annotation and classification challenges for temporal knowledge base population, *Knowledge and Information Systems*, pp. 1–36.

Kozlova, E., M. Maria, P. Elena, and T. Popova (2012), The compreno semantic model as an integral framework for a multilingual lexical database, in *3rd Workshop on Cognitive Aspects of the Lexicon (CogALex-III)*.

Kulkarni, S., A. Singh, G. Ramakrishnan, and S. Chakrabarti (2009), Collective annotation of wikipedia entities in web text, in *KDD*, pp. 457–466.

Kwiatkowski, T., L. Zettlemoyer, S. Goldwater, and M. Steedman (2011), Lexical Generalization in CCG Grammar Induction for Semantic Parsing, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lesk, M. (1986), Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone, in *Proceedings of the SIGDOC Conference*.

Lin, T., Mausam, and O. Etzioni (2012a), No Noun Phrase Left Behind: Detecting and Typing Unlinkable Entities, in *EMNLP*.

Lin, T., Mausam, and O. Etzioni (2012b), Entity Linking at Web Scale, in *AKBC-WEKEX*.

Liu, D., and J. Nocedal (1989), On the limited memory method for large scale optimization, *Mathematical Programming B*, *45*(3), 503–528.

Mann, G., and D. Yarowsky (2003), Unsupervised personal name disambiguation, in *CoNLL*.

McNamee, P., et al. (2009), HLTCOE Approaches to Knowledge Base Population at TAC 2009, in *Text Analysis Conference*.

Meij, E., W. Weerkamp, and M. de Rijke (2012), Adding semantics to microblog posts, in *WSDM*.

Mendes, P. N., M. Jakob, and C. Bizer (2011), Evaluating DBpedia Spotlight for the TAC-KBP Entity Linking Task, in *TAC*.

Mendes, P. N., M. Jakob, and C. Bizer (2012), DBpedia for NLP: A Multilingual Cross-domain Knowledge Base, in *LREC*.

Mihalcea, R., and A. Csomai (2007), Wikify!: Linking documents to encyclopedic knowledge, in *CIKM*, pp. 233–242.

Milne, D., and I. H. Witten (2008), Learning to link with wikipedia, in *CIKM*.

Mintz, M., S. Bills, R. Snow, and D. Jurafsky (2009), Distant supervision for relation extraction without labeled data, in *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-2009)*, pp. 1003–1011.

Pantel, P., and A. Fuxman (2011), Jigs and Lures: Associating Web Queries with Structured Entities, in *ACL*.

Punyakanok, V., and D. Roth (2001), The use of classifiers in sequential inference.

Pustejovsky, J., and M. Verhagen (2009), Semeval-2010 task 13: evaluating events, time expressions, and temporal relations (tempeval-2), in *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pp. 112–116, Association for Computational Linguistics.

Pustejovsky, J., J. M. Castano, R. Ingria, R. Sauri, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev (2003), Timeml: Robust specification of event and temporal expressions in text., *New directions in question answering*, *3*, 28–34.

Ratinov, L., and D. Roth (2009), Design challenges and misconceptions in named entity recognition, in *CoNLL*.

Ratinov, L., D. Roth, D. Downey, and M. Anderson (2011), Local and global algorithms for disambiguation to wikipedia, in *ACL*.

Riedel, S., L. Yao, and A. McCallum (2010), Modeling relations and their mentions without labeled text, in *Proceedings of the Sixteenth European Conference on Machine Learning (ECML-2010)*, pp. 148–163.

Sil, A. (2013), Exploring Re-ranking Approaches for Joint Named-Entity Recognition and Linking, in *Proceedings of the Sixth Workshop on Ph.D. students in Information and Knowledge Management (PIKM)*, pp. 11–18, ACM.

Sil, A., and S. Cucerzan (2014), Temporal Scoping of Relational Facts based on Wikipedia Data, in *Proceedings of the Computational Natural Language Learning (CoNLL)*.

Sil, A., and A. Yates (2011a), Machine Reading between the Lines: A Simple Evaluation Framework for Extracted Knowledge Bases, in *Workshop on Information Extraction and Knowledge Acquisition (IEKA)*.

Sil, A., and A. Yates (2011b), Extracting STRIPS representations of actions and events, in *RANLP*.

Sil, A., and A. Yates (2013), Re-ranking for Joint Named-Entity Recognition and Linking, in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*.

Sil, A., F. Huang, and A. Yates (2010), Extracting action and event semantics from web text, in *AAAI Fall Symposium on Common-Sense Knowledge (CSK)*.

Sil, A., E. Cronin, P. Nie, Y. Yang, A.-M. Popescu, and A. Yates (2012a), Linking Named Entities to Any Database, in *EMNLP-CoNLL*.

Sil, A., A. Shelton, D. J. Ketelhut, and A. Yates (2012b), Automatic Grading of Scientific Inquiry, in *The 7th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 7)*.

Silberschatz, A., H. F. Korth, and S. Sudarshan (2010), *Database System Concepts*, sixth ed., McGraw-Hill.

Singh, S., S. Riedel, B. Martin, J. Zheng, and A. McCallum (2013), Joint inference of entities, relations, and coreference, in *Proceedings of the 2013 workshop on Automated knowledge base construction*, pp. 1–6, ACM.

Suchanek, F. M., G. Kasneci, and G. Weikum (2007), Yago: a core of semantic knowledge, in *WWW*.

Sundheim, B. M. (1991), Overview of the third message understanding evaluation and conference, in *Proceedings of the 3rd conference on Message understanding*, pp. 3–16, Association for Computational Linguistics.

Surdeanu, M., S. Gupta, J. Bauer, D. McClosky, A. X. Chang, V. I. Spitkovsky, and C. D. Manning (2011), Stanfords distantly-supervised slot-filling system, in *Proceedings of the Text Analytics Conference*.

Talukdar, P. P., D. Wijaya, and T. Mitchell (2012a), Coupled temporal scoping of relational facts, in *WSDM*.

Talukdar, P. P., D. Wijaya, and T. Mitchell (2012b), Acquiring temporal constraints between relations, in *CIKM*.

Taskar, B., C. Guestrin, and D. Koller (2003), Max-margin markov networks, *NIPS*.

Tjong Kim Sang, E. F., and F. De Meulder (2003), Introduction to the conll-2003 shared task: Language-independent named entity recognition, in *Seventh Conference on Natural language learning at HLT-NAACL 2003-Volume 4*.

Tsochantaridis, I., T. Joachims, T. Hofmann, Y. Altun, and Y. Singer (2006), Large margin methods for structured and interdependent output variables, *JMLR*.

Turney, P. D. (2002), Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews, in *Procs. of ACL*, pp. 417–424.

Wang, Y., M. Zhu, L. Qu, M. Spaniol, and G. Weikum (2010), Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia, in *Proceedings of the 13th International Conference on Extending Database Technology*, pp. 697–700, ACM.

Wang, Y., B. Yang, L. Qu, M. Spaniol, and G. Weikum (2011), Harvesting facts from textual web sources by constrained label propagation, in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 837–846, ACM.

Weld, D. S., R. Hoffmann, and F. Wu (2009), Using Wikipedia to Bootstrap Open Information Extraction, in *ACM SIGMOD Record*.

Yao, L., S. Riedel, and A. McCallum (2010), Collective cross-document relation extraction without labelled data, in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*, pp. 1013–1023.

Zhou, Y., L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney (2010), Resolving surface forms to wikipedia topics, in *Coling*, pp. 1335–1343.