LINKING ENTITIES TO A KNOWLEDGE BASE

by

WEI ZHANG

(B. COMP)

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE
2013

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

_____

Wei Zhang

Jan. 1, 2013

ACKNOWLEDGEMENTS

Thanks very much to my teachers, friends, and family. They in various ways have helped shape this work from beginning to end. It has been one very challenging and unforgettable adventure in my life.

First of all, I would like to express my thanks to my supervisors Dr. Jian Su and Prof. Chew-Lim Tan sincerely for their valuable advice and in-depth guidance throughout my Ph.D. study. In the past four and half years, I learned a lot from them. With their help, I learned how to survey a field of interest, how to discover interesting research topics, how to write research papers and how to do presentation clearly. Besides, during the last half year, I sincerely appreciate the help of Prof. Tan for the tuition waiver and the temporary job provided by Dr. Su. I also thank Dr. Su for her kind help in my life rather than only research, such as the advice and information on the job hunting of my wife, and the newspapers and digital recorder given to me for improving my spoken English.

I am also very thankful to Dr. Colin Keng-Yan Tan and Prof. Mong-Li Lee for serving as examiners of my graduate research paper and thesis proposal. Their valuable comments are of great help in my research work. Furthermore, during my internship at Microsoft Research Asia, I got generous help from my mentor Dr. Yunbo Cao and Dr. Chin-Yew Lin. I thank them very much.

My thanks also go to the following friends and lab-mates for their help throughout my studies: Yan-Chuan Sim, Bin Chen, Wenting Wang, Zhiqiang Toh, Man Lan, Long Qiu, Sinno Pan, Kai Wang, Zhijie He, Yun Huang and Bolan Su.

Last but not least, I would like to give my deepest gratitude to my parents

Yongcheng Zhang and Suyan Zhang, who always encourage and support me when I feel depressed. Meanwhile, I would also like to give my great thanks to my wife Chaoya Liu. Her kind help and support make everything I have possible. I dedicate this dissertation to my parents and my wife.

Wei Zhang

Jan. 1, 2013

TABLE OF CONTENTS

Linking Entities to a Knowledge Base

WEI ZHANG
National University of Singapore, 2013
Supervisors: Prof. Chew-Lim Tan and Dr. Jian Su

Abstract

The explosive growth in the amount of textual information brings a need for building a structured Knowledge Base (KB) to organize the knowledge scattered among these unstructured texts. On the other hand, the available KBs such as Wikipedia and Google Knowledge Graph which contain rich knowledge about the world's entities have been shown to form a valuable component for many natural language precessing (NLP) tasks. To populate or to utilize the KBs, we need to link the mentions of entities in text to their corresponding entries in the KB, which is called entity linking.

Most of state-of-the-art entity linking systems use annotated data to learn a classifier or ranker by supervised learning algorithms. Our research initially focuses on automatically labeling a large scale training corpus for the supervised learning algorithms, where we label the ambiguous mentions leveraging on their unambiguous synonyms. We also propose an instance selection strategy to select an informative, representative and diverse subset from the auto-generated data set.

Next, we introduce topic models to entity linking for measuring the context similarity between mention and KB entries. We propose a Wikipedia-LDA method to model the context as some hidden topics instead of only treating the

context as literal terms. We investigate the effectiveness of five subsets from Wikipedia categories to represent the underlying topics.

Besides, we propose a lazy learning model for entity linking, which can incorporate the query-specific information to the learning process by automatically labeling some data for the queried name. Then, instead of only using the labeled data set related with other names to train the linker, we propose to use the predictive structure shared by the two data sets which are related with queried name and other names respectively.

Finally, this thesis addresses entity linking task under a more challenging scenario, where we link the mentions in microblog to a KB in real time. We propose an unsupervised learning framework (USLF) which is based on three bipartite graphs to address the new challenges in microblog. Our USLF uses a Bayes method to model the three clues of disambiguation: the context information of query and entities, popularity knowledge of entities and clustering result on an additional tweet set. Besides, in our USLF, a tweet enrichment function is embedded based on the word similarity, which is calculated in the k principal component space of the word set with the help of auxiliary long text.

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

1.1 Motivations

The last decade has seen an explosive growth in the amount of textual information on the web and in some specific domains such as business reports and news articles. There is a need for building a structured Knowledge Base (KB) to organize the knowledge about the world's entities scattered among these unstructured texts. However, automatically populating an existing KB with the fresh information from unstructured texts requires linking the mentions of entities in text to their corresponding entries in the KB or highlighting these mentions as new entries to the current KB.

On the other hand, the available KBs such as Wikipedia [1], OpenCyc [2], KIM [3] (Popov et al., 2004), and Google Knowledge Graph [4] which contain rich knowledge about the world's entities have been shown to form a valuable component for many Natural Language Processing tasks such as text classification (Wang and Domeniconi, 2008), and cross-document coreference (Finin et al., 2009). However, to be able to utilize the KB resource, these applications also require linking the mentions of entities in text to their corresponding entries in the knowledge bases.

[1] http://www.wikipedia.org/
[2] http://www.opencyc.org/
[3] http://www.ontotext.com/kim
[4] http://www.google.com/insidesearch/features/search/knowledge.html

1.2 Entity Linking Benchmarks

Entity linking task has been proposed and studied in Text Analysis Conference (*TAC*) since 2009 (McNamee and Dang, 2009). In *TAC-09* [5], entity linking is defined as aligning a textual mention of an entity (entities are person, organization or geopolitical entities) in text to its appropriate entry in the knowledge base if such entity has an entry in the knowledge base, otherwise highlighting that the entity does not have an entry in the KB. More intuitively, given a news article or web blog with a mention, for example, the mention *SAS* in the following news article,

***SAS** creates R&D division to develop fraud and compliance applications*

*Business analytics firm **SAS** has created a R&D division to create applications to detect fraud and ensure compliance. The company has created the new **SAS** R&D Fraud and Compliance Solutions Division to address the need for enterprises to detect multichannel fraud and regulatory non-compliance...*

and given a KB, e.g. Wikipedia [6] (an example of the KB entry shown in Figure 1.1), entity linking system should return the correct KB entry for the mention if such entry exists in KB, otherwise return *NIL*. In the *SAS* example above, the KB entry for the software company shown in Figure 1.1 should be returned.

In the remainder of this thesis, we use "query" to denote the mention and its associated article.

In *TAC-09*, the KB derived from Wikipedia contains 818,741 different entries. Each KB entry as shown in Figure 1.1 consists of the Wikipedia Infobox [7]

[5] http://apl.jhu.edu/ paulmac/kbp.html
[6] http://www.wikipedia.org/
[7] http://en.wikipedia.org/wiki/Template:Infobox

Figure 1.1: An Example of Wikipedia Article

and the corresponding Wikipedia page text.

The annotated data set of *TAC-09* has 3,904 queries across three named entity (**NE**) types: Person (**PER**), Geo-Political Entity (**GPE**) and Organization (**ORG**). The documents containing the mentions are from a document collection which contains 1.3 million documents from newswire text. The details of the annotated data set can be found in Table 1.1.

Type	# Queries	in KB	NIL
PER	627	255	372
ORG	2710	1013	1697
GPE	567	407	160
All	3904	1675	2229

Table 1.1: Number of *TAC-09* Queries by NE Type and KB Presence

To evaluate entity linking systems, *TAC-09* officially used micro-averaged accuracy (i.e. the number of correct links divided by the total number of the queries).

Entity linking at *TAC-10* [8] (Ji et al., 2010) is a follow-on to the evaluation at *TAC-09*. The change in 2010 is to emphasize genre diversity by adding 0.4 million web documents into the document collection. Then, the document collection in *TAC-10* contains 1.7 million documents from newswire and blog text. The training and test corpora statistics including genre and NE type are listed in Table 1.2.

Corpus	Genre	PER	ORG	GPE	All
Training	Newswire	627	2710	567	3904
	Web data	500	500	500	1500
Test	Newswire	500	500	500	1500
	Web data	250	250	250	750

Table 1.2: Number of Queries in Data Sets of *TAC-10*

This thesis will focus on the entity linking task as defined in *TAC-09 and TAC-10*. *TAC-11* [9] (Ji et al., 2011b) further requires entity linking systems to cluster *NIL* queries (the mentions do not have entry in KB) and then each cluster represents one entity. Then entity linking system in *TAC-11* is required

[8] http://nlp.cs.qc.cuny.edu/kbp/2010/
[9] http://nlp.cs.qc.cuny.edu/kbp/2011/

to: (1) judge whether each query can be linked to any KB entry; (2) cluster all queries linked with *NIL* into clusters. The system output can be viewed as a collection of various clusters: some clusters are labeled as KB entries.

In *TAC-11*, entity linking is also extended to a cross-lingual setting, in which the queries come from both English and Chinese.

Table 1.3 lists the training and test corpora statistics at *TAC-11*.

Corpus	Genre	PER	ORG	GPE	All
Training	Newswire (2009 test set)	627	2710	567	3904
	2010 Training Web data	500	500	500	1500
	2010 test Newswire	500	500	500	1500
	2010 test Web data	250	250	250	750
Test	Newswire	500	500	500	1500
	Web data	250	250	250	750

Table 1.3: Number of Queries in Data Sets of *TAC-11*

In the third evaluation campaign of Web People Search (WePS-3) [10], a task of *Online Reputation Management* was proposed (Amigo et al., 2010; Spina et al., 2011), which is the same as entity linking when KB only has one entry. Given a set of tweets containing an ambiguous company name, and given the home page of the company, this task is to filter out the tweets that do not refer the company. For each company, systems are provided with the company name (e.g. *apple*) used as a query to retrieve the stream of tweets to annotate, and a representative URL (e.g. http://www.apple.com) that identifies the target company. The input information per tweet consists of a tuple containing: the tweet identifier, the organization name, the query used to retrieve the tweet, the author identifier, the date and the tweet content. Systems have to label each tweet as related (that is, the tweet refers to the company) or unrelated (the tweet does

[10] http://nlp.uned.es/weps/

not refer to the given company).

1.3 Name Variation and Name Ambiguity

The major challenges in the task of linking entities in text to a KB are the problems: name variation and name ambiguity.

Name variation refers to the case that more than one name variations such as *alias, misspelling* and *acronym* refer to the same entity. For example, both *"48th State"* and *"The Grand Canyon State"* refer to *state of Arizona, U.S.*. Thus, the names for the same entity in the article and KB may be different, and entity linking is required to bridge these different names.

Name ambiguity refers to the case that more than one entities share the same name. For example, the mention *"AZ"* in the article may refer to the US state *state of Arizona*, the Italian airline *Alitalia*, the country *Azerbaijan*, or other entries in KB that have the same name, and entity linking system should figure out the correct KB entry based on the context of the mention.

1.4 Related Tasks

In some work, entity linking is also called *named entity disambiguation using Wikipedia* (Bunescu and Pasca, 2006; Cucerzan, 2007). Bunescu and Pasca (2006) employed several of the Wikipedia resources for entity disambiguation including *Wikipedia entity pages*, *redirect pages* [11], *categories* [12] and *hyperlinks*. These resources have been widely used in the entity linking systems recently. Cucerzan (2007) stressed that the entities in the same document

[11] http://en.wikipedia.org/wiki/Wikipedia:Redirect
[12] http://en.wikipedia.org/wiki/Help:Category

should be related and they linked all the entities in the document to the KB simultaneously by considering their global coherence. The details of the methods to utilize the Wikipedia resources and global coherence will be explored in Chapter 2.

A similar task to entity linking is *Wikification* (Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ratinov et al., 2011), which links expressions in text to their referent Wikipedia pages. Note that *Wikification* attempts to link all "interesting" expressions to Wikipedia, mimicking the link structure found in Wikipedia. In contrast, entity linking studies focus on linking named entities (person, organization or geopolitical entities).

More generally, resolving ambiguous names in Web People Search (WePS) (Artiles et al., 2007) and Cross-document Coreference (Bagga and Baldwin, 1998) disambiguates names by clustering the articles according to the entity mentioned. For example, given a set of articles all containing the mention "apple", the system output will be a collection of clusters and articles in each cluster refer to an entity such as the company *Apple Inc*, the fruit *Apple* and so on. This differs significantly from entity linking, which has a given entity list (i.e. the KB) to which we disambiguate the mentions.

Finally, a remote related task is word sense disambiguation (WSD) (Mihalcea and Moldovan, 1999; Kilgarriff and Rosenzweig, 2000; Edmonds and Cotton, 2001; Lee and Ng, 2002; Ando, 2006), which aims to assign dictionary meanings to the words in a corpus. The words in training and test corpora are usually the same. This is differs significantly from entity linking, where names in training and test corpora are different since the names of the entities form an infinite set and the name in the query is unseen. Another difference is that en-

tity linking disambiguates entities to millions of KB entries, not disambiguate words to a few pre-defined word senses. Thus, entity linking needs to generate the candidate list from the whole KB on the fly. Besides, entities are infinite and thus KB is partial. Entity linking needs to find *NIL* entity to populate the KB.

1.5 Thesis Contributions

This thesis focuses on the supervised learning approaches for entity linking. First, we automatically label the training data for entity linking. Next, we design the features for the learning process. Third, we popose the learning framework for entity linking. Finally, we focus on entity linking in short text.

We list the contributions of this thesis to the task of entity linking as follows:

1. Most existing studies on entity linking use thousands of annotated instances to learn a classifier or ranker (Dredze et al., 2010; Lehmann et al., 2010; Zheng et al., 2010; Zhang et al., 2010; Ploch, 2011; Ratinov et al., 2011) or to estimate parameters (Gottipati and Jiang, 2011; Han and Sun, 2011). Besides, from the analysis by (Ji et al., 2011b), all of the top systems from the participants in the shared task of *TAC-11* also use supervised learning approaches to solve entity linking problem.

   As manually creating a training corpus for entity linking is labor-intensive and costly, we propose a method to automatically label a large scale training corpus for entity linking, where we label the ambiguous mentions leveraging on their unambiguous synonyms. The basic idea is to take an article with an unambiguous mention only referring to an entity E_1,

and then replace the mention with a phrase which may refer to E_1, E_2 or other entities.

Furthermore, the distribution of the unambiguous mentions is not consistent with the mentions in the real world. Thus, to utilize the auto-labeled data set without compromising the accuracy of entity linking system, we also propose an instance selection strategy to select an informative, representative and diverse subset from this auto-labeled data set.

2. The traditional entity linking approaches treat the context of the mention and the text of a KB entry as literal term vectors where terms can be a bag of words, n-grams, noun phrases or/and co-occurring named entities, and they measure the similarity between the context of the mention and the text of the KB entry by the comparison of their weighted term vectors. Such literal matching suffers from the problems: lack of semantic information and sparsity issues. For example, the following two sentences describe the same "*Michael Jordan*", but literal matching can not correctly link them together as there is no shared term between them.

   (1) Michael Jordan (born February, 1963) is a former American professional basketball player.

   (2) Michael Jordan wins NBA MVP of 91-92 season.

   In this thesis, we introduce a topic model to entity linking to discover the underlying topics in the context of mention and KB entries. Then, the similarity between articles and KB entries also can be calculated in the semantic space of the hidden topics. For example, the two sentences above have the same topic "sports".

3. Entities are infinite and thus names of the entities in the annotated data set are partial. This property of entity linking causes that supervised-learning based approaches disambiguate a new ambiguous name (e.g. *"AZ"*) based on the distribution knowledge learned from instances in the training set, which are related to other names (e.g. *"Hoffman"*, *"Chad Johnson"*, etc.). However, there are some gaps among the distributions of the instances related to different names, and these gaps hinder the further improvement of the previous entity linking approaches.

   To narrow down the gap between the query and the training data, this thesis proposes a lazy learning model, in which generalizing the model on the labeled data is delayed until a query is made. This allows the training instances specific to the name in the query (e.g.*"AZ"*) to be incorporated into the learning process.

4. This thesis also addresses entity linking task under a more challenging scenario, where we link the mentions in microblog to a KB in real time. We propose an unsupervised learning framework which is based on three bipartite graphs to address the new challenges in microblog.

   First, microblog messages are much shorter usually consisting of only not more than 140 characters. To address the problem of insufficient text in microblog, we propose a bipartite graph based mapping function to enrich the context of the microblog messages with the auxiliary long texts from web.

   Second, real-time entity linking in microblog requires a quick response time, as it needs to monitor the tweets that keep coming at a fast pace.

However, the important technique for traditional entity linking systems that gathers specific information for the query from external sources on-the-fly can not meet this requirement. In this thesis, to benefit from the external information: the microblog messages which also contain the name in the query, and also to guarantee a fast response time, we propose a bipartite graph based model to cluster the external tweets and then we store the cluster information into a multinomial model off-line.

Third, in microblog messages, people usually only use the short name of the entities, unlike news articles usually containing the unambiguous full name of the entity. This property of entity linking in microblog disables the method described above to automatically label training instances by leveraging the unambiguous synonyms. In this thesis, we propose to use the unsupervised clustering result to guide our supervised Bayesian model. Thus, our method does not need human labor to annotate training data.

Our publications related with the work presented in this thesis are as follows: the method to automatically label the training corpus is published in (Zhang et al., 2010), the instance selection strategy and topic model for entity linking are published in (Zhang et al., 2011b) and (Zhang et al., 2011c), and the lazy learning for entity linking is published in (Zhang et al., 2012).

1.6 Thesis Overview

The rest of the thesis is organized as follows,

- Chapter 2 conducts a literature survey on entity linking. The survey sum-

marizes the existing work on entity linking and also presents their pros and cons.

- Chapter 3 presents the method to automatically label the training corpus and applies the proposed instance selection strategy to this corpus.
- Chapter 4 introduces the topic model to entity linking, where we investigate both unsupervised and supervised topic models for entity linking.
- Chapter 5 describes the lazy learning which incorporates the query-specific information into entity linking.
- Chapter 6 presents our unsupervised learning framework for entity linking in microblog.
- Chapter 7 gives concluding remarks.

Chapter 2: STATE OF THE ART

2.1 General Architecture of Entity Linking System

Given a query q (a document d_q with a mention m_q) and a KB, our goal is to select the correct KB entry e from the KB. Ji et al. (2011b) summarize the existing work on entity linking and conclude a general architecture of entity linking system (see Figure 2.1). It includes four steps: (1) query expansion - expand the name in query into a richer set of variations using coreference resolution in the background document and also expand the name variation set of each entry in KB using Wikipedia; (2) candidate generation - finding all possible KB entries C_q=$\{c_1,..,c_N\}$ that a query might link to; (3) candidate ranking - rank the probabilities of all candidates C_q=$\{c_1,..,c_N\}$ and select the top KB entry c_n; (4) NIL detection - detect the NILs which got low confidence at matching the top KB entry c_n in step (3). Finally, the answer e is c_n or *NIL*. In the following subsections we will highlight the existing work and their effective techniques used in each step.

2.2 Query Expansion

As shown in Section 1.3, name variation causes that the names in the query and in the KB might be different. For example, the name in the query is "*US*", but "*United States*" in the knowledge base. Thus, entity linking systems firstly

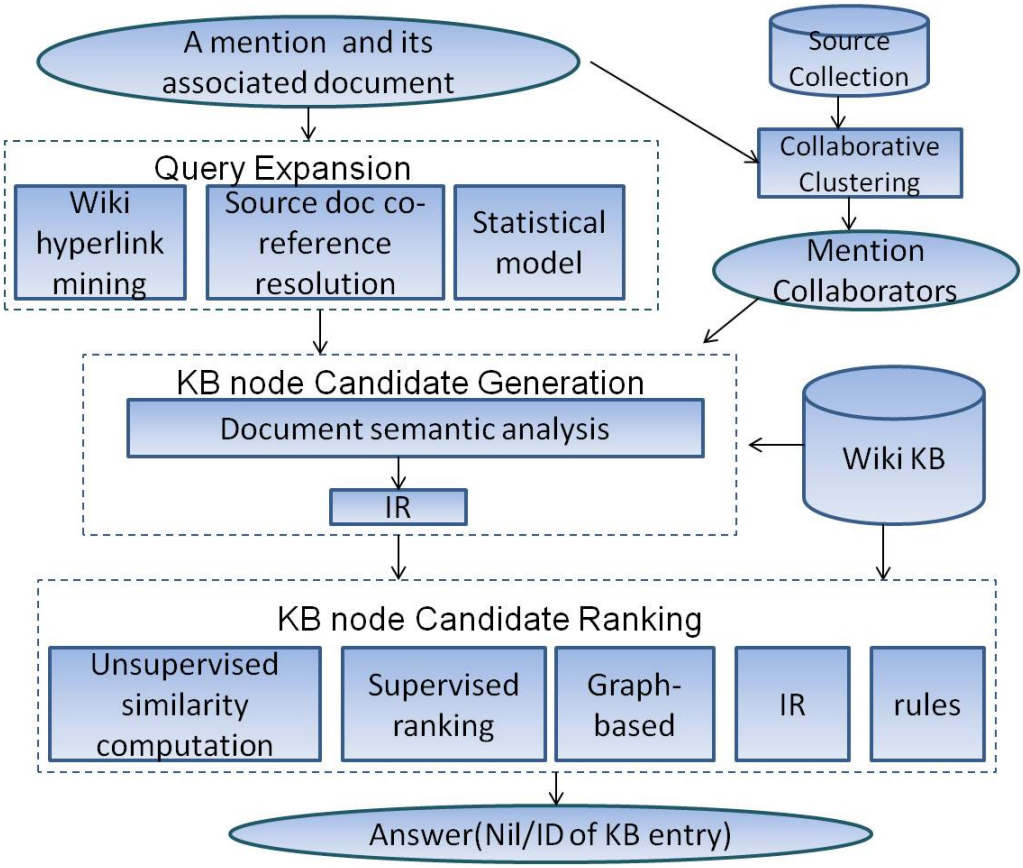


Figure 2.1: General Entity Linking System Architecture

expand the name into a richer set of variations for both query and the entries of knowledge base, expanding the name in the query to name variation set {*"US"*, *"USA"*, ...} and expanding the name of KB entry to the set {*"United States"*, *"USA"*, ...}. Then, we can bridge the mention and the KB entry by string matching (e.g. the shared string *"USA"*).

2.2.1 Name Variations from Background Document for the Mention in Query

Previous work (Srinivasan et al., 2009) used co-reference chain to find the name variations from the background document for the query. More specifically, the noun phrases co-occurring in the same chain with the mention of the query are selected to form the name variation set.

Besides, Gottipati and Jiang (2011) mined variations by running a Named Entity Recognizer on the background document. The recognized name strings which contain the string of the query mention are selected as the variations. For example, recognized name string - "*Sophia Coppola*" Vs. query mention- "*Coppola*") and recognized name string- "*Apple, Inc*" Vs. query mention- "*Apple*").

2.2.2 Name Variations from Wikipedia for KB Entries

Entity linking systems also use external world knowledge to build the name variation set for each KB entry. Since *TAC* uses a KB derived from Wikipedia, and other KBs such as *OpenCyc* [1] and *KIM* [2] usually can be mapped to Wikipedia (Nguyen and Cao, 2008), entity linking systems (Bunescu and Pasca, 2006; Cucerzan, 2007; Varma et al., 2009; Bysani et al., 2010; Zhang et al., 2010; Zheng et al., 2010; Lehmann et al., 2010; Cassidy et al., 2011; Zhang et al., 2011b; Zhang et al., 2011c; Zhang et al., 2012) usually find the variations for the entries in KB by leveraging name variation sources in Wikipedia: *"titles of*

[1] http://www.opencyc.org/
[2] http://www.ontotext.com/kim

entity pages", *"disambiguation pages* [3] *"*, *"redirect pages* [4] *"*, *"bold texts from first paragraphs"* and *"anchor texts"*.

Entity Pages: An entity page is a Wikipedia article that contains information focused on one single entity, such as a person, a place, or a work of art. For example, Wikipedia contains a page titled "*Texas (TV series)*", which offers information about the soap opera that aired on *NBC* from 1980 until 1982.

Disambiguation Pages: Disambiguation pages are specially marked articles having as title a name form, typically followed by the word "disambiguation" (e.g., *"Texas (disambiguation)"*), and containing a list of references to pages for entities that are typically mentioned using that name form. This is more useful in extracting the abbreviations of entities, other possible names for an entity, etc.

Redirect Pages: A redirect page in Wikipedia is an aid to navigation. When a page in Wikipedia is redirected, it means that those set of pages are referring to the same entity. They often indicate synonym terms, but can also be abbreviations, more scientific or more common terms, frequent misspellings or alternative spellings, etc. For example, the article titled *"Another World in Texas"* contains a redirection to the article titled *"Texas (TV series)"*.

Bold Text from First Paragraph: In Wikipedia the first paragraph usually contains a summary of the article or most important information about the article, thus containing the most relevant words for that article. Entity linking systems extract phrases from the first paragraph of Wikipedia article that are written in bold font. This bold text generally refers to nick names, abbreviations, full names etc.

[3] http://en.wikipedia.org/wiki/Wikipedia:Disambiguation
[4] http://en.wikipedia.org/wiki/Wikipedia:Redirect

Anchor Texts: Wikipedia entity pages usually contain many hyperlinks referring to another Wikipedia entity page. Illustratively, the article for *Pam Long* contains a hyperlink which uses the name *"Texas"* to refer to *"Texas (TV series)"*.

2.2.3 Query Rewrite

Sometimes the mention in the query is acronym (capitalized word). Expanding a name in acronym form from the background document can effectively reduce the ambiguities of the mention under the assumption that two variants in the same document refer to the same entity. For example, the short form *TSE* in Wikipedia refers to 33 entries, but with its full name *Tokyo Stock Exchange*, which is unambiguous, we can directly link it to the correct entry without the needs of disambiguation.

Zhang et al. (2011a) and Cassidy et al. (2011) mined full form for acronyms using some common patterns such as *"full form (acronym)"* (e.g.*"All Basotho Convention (ABC)"*). However, the rule-based methods cannot capture complicated acronyms such as swapped or missed acronym letters (e.g. *"CCP" vs. "Communist Party of China"; "MD" vs. "Ministry of Defence"*) and multiple letters from expansion (*"MINDEF" vs. "Ministry of Defence"*). Zhang et al. (2011b) trained a statistical classifier to detect full form for the acronym from its context and achieved 15.1% accuracy improvement over state-of-the-art acronym expansion methods.

If the full form of the query is found from its context, entity linking systems will use the new mention with less ambiguity as the new query.

Besides, the mentions in the query sometimes are misspelled. Our work

(Zhang et al., 2010) also used the web tools such as "did you mean" feature in Google and Wikipedia to correct the misspelling of mention. For example, "*Abbot Nutrition*" can be corrected to "*Abbott Nutrition*" by Wikipedia "did you mean". Then, the correct spelling is used as the new query.

2.3 Candidate Generation

Because the knowledge base usually contains millions of entries, it is time-consuming to apply the disambiguation algorithm to the entire knowledge base. Thus, this step is conducted to filter out irrelevant KB entries and select from KB only a set of candidates that are potentially the correct match to the given query. Given the query and the name variations for each KB entry found in Section 2.2.2, the candidates can be selected by comparing the name string of the mention in query with the name strings in the variation set of each KB entry. The KB entry with a name string which matches the name of the mention is considered as a candidate.

To increase the recall of the candidate set, Dredze et al. (2010) further find more candidates using the following approach:

- The titles of the KB entries are wholly contained in or contain the mention (e.g., *Nationwide* and *Nationwide Insurance*).
- The first letters of the entity mention match the KB entry title (e.g., *OA* and *Olympic Airlines*).
- The title has a strong string similarity score with the entity mention. They include several measures of string similarity, including: character Dice score > 0.9, skip bigram Dice score > 0.6, and

Hamming distance $<= 2$.

Besides, unlike the above methods using only the mention in the query, (Srinivasan et al., 2009) and (Gottipati and Jiang, 2011) also used the name variation set of the mention (built in Section 2.2.1) to generate the candidates.

2.4 Candidate Ranking and NIL Detection

In entity linking, the methods of candidate generation normally achieve a recall of more than 95%. However, as the name ambiguity problem (See Section 1.3), more than one candidates are usually generated for a given mention. Therefore, the most crucial step for entity linking is ranking the KB candidates and then selecting the best KB entry. Note that the contributions of this thesis in the following chapters are all for this step of entity linking.

The ranking task can be formalized as follows. We are given a query q (i.e. a document d_q with a mention m_q) and its associated KB candidates C_q=$\{c_1,..,c_N\}$ generated in Section 2.3, and our goal is to select the correct KB entry e from the set C_q. Specifically, let $\phi_q(q,c_i)$ be a score function reflecting the likelihood that the candidate c_i is the correct KB entry for q. Then, a ranking model for linking is to solve the following optimization problem:

$$e = \arg\max_{c_i \in C_q} \phi_q(q, c_i) \tag{2.1}$$

The approaches exploited in existing entity linking systems can be generally categorized into two types:

(1)Unsupervised or weakly-supervised learning, in which annotated data is minimally used to tune thresholds and parameters, and the score function

is largely based on the unlabeled contexts. For example, The top 1 system of *TAC-09* (Varma et al., 2009) used the Information Retrieval tool Lucene [5] to index the candidates, and formulated the document d_q as the input to Lucene for ranking the candidates. Gottipati and Jiang (2011) used a KL-divergence retrieval model to rank the candidates.

(2)Supervised learning, in which a pair of query and KB entry is modeled as an instance for feature-based machine learning algorithms. Such a supervised model can be learned from the annotated training data based on many various features. The labels for training instances are positive (i.e. linked together) and negative (i.e. not linked together). As supervised learning uses the supervision from labeled instances to combine a large range of information such as surface and semantic knowledge in Table 2.1, supervised learning usually can obtain a better linking accuracy than unsupervised methods only leveraging the context information. Thus, more and more researchers have used supervised learning approaches for entity linking recently, including Support Vector Machines (SVM) classification (Zhang et al., 2010; Zhang et al., 2011b; Zhang et al., 2011c), Maximum Entropy (Cassidy et al., 2011; Chang et al., 2010), Ranking SVM (Dredze et al., 2010), ListNet (Zheng et al., 2010), Generative Model (Han and Sun, 2011), Logistic Regression Classifier (Monahan et al., 2011), Random Forests (Zhao et al., 2011) and Markov-Logic Network (Dai et al., 2010).

Support Vector Machines based ranking and Maximum Entropy based ranking are the most popular methods in entity linking. (Chen and Ji, 2011) observed that ListNet achieved the best performance compared to seven other

[5]http://lucene.apache.org/core/

ranking algorithms including SVMRank and Maximum Entropy at the same setting of features and resources.

Supervised learning based methods involve a lot of feature engineering and design. Table 2.1 summarizes the typical ranking features used in existing entity linking systems.

Contextual Features (**CF**) capture the intuition that a candidate c_i is more likely to be referred to by a mention m_q if the text of c_i has more words or named entities shared with the text of m_q.

Semantic Features (**SeF**) capture the intuition that a candidate c_i is more likely to be referred to by a mention m_q in d_p if the text of c_i has a high semantic similarity to d_q, or the named entity type of c_i is consistent with m_q.

Surface Features (**SuF**) represent the likelihood of a name referring to a specific KB entry. For example, *"Arizona"* is more likely refer to *"State of Arizona"* than *"AZ"*.

Generation Source (**GS**) allows disambiguation to make decision depending on the origin of the candidate.

For Popularity Features (**PF**), in *TAC09* and *TAC10* most systems had to rely on Web access (e.g. ranking of Wikipedia pages from search engine) to estimate the popularity of a candidate KB entry. In contrast, Han and Sun (2011) computed popularity based on the distribution of the name of the KB entry in a large document collection.

For *NIL* mentions detection, some work (Dredze et al., 2010; Han and Sun, 2011; Dai et al., 2010) added *NIL* to the candidate set and ranked it together with other candidates. The other work (Zhang et al., 2010; Zheng et al., 2010; Monahan et al., 2011) used an additional classification step on the top 1 candi-

Name	Description
*Contextual Features (**CF**)*	
Bag of Words*	The cosine similarity (tf.idf weighting) between query text and text of the candidate.
Genre	Genre of the query text (newswire, blog, ...)
Position*	Query name appears early in KB text
Co-occurring NEs*	Entities co-occurred, or involved in some attributes/relations/events with the query
*Semantic Features (**SeF**)*	
NE type*	True if NE type (i.e. Person, GPE, Organization) of the query and the candidate is consistent.
Profile	Slot fills of the query, KB attributes
KB Link Mining	Attributes extracted from the hyperlink graphs (in-links, out-links) of the KB article
*Surface Features (**SuF**)*	
Spelling Match*	Exact string match, acronym match, alias match, string match based on edit distance, ratio of longest common subsequence to total string length, name component match, first letter match for abbreviations, organization suffix word match
Name Gazetteer*	Organization and geo-political entity abbreviation gazetteers
*Generation Source (**GS**)*	
Wikipedia Source*	True for each Wikipedia source (i.e. “entity pages", “disambiguation pages", “redirect pages" and “anchor texts" (Section 2.2.2)) which generates the candidate
String Match*	For the candidate not generated from Wikipedia source, true if it is generated from full match.
*Popularity Features(**PF**)*	
Web	Top KB text ranked by search engine and its length
Frequency	Frequency in KB texts

Table 2.1: Feature Set for Ranking (* Features used in our experiments of the following chapters)

date to resolve this problem.

2.5 Mention Collaborators

Let $M = \{m_1, m_2, ...m_N\}$ denote the N mentions in a document and let $C_i = \{c_{i1}, c_{i2}, c_{in}\}$ denote the candidate set for mention m_i. The above methods for candidate ranking disambiguate each mention m_i separately, utilizing clues such as maximizing the textual similarity between the document and each KB entry in C_i. This section discusses an important technique for candidate ranking - Collaborative Ranking (also named as Collective Entity Linking or Global Algorithm), in which all N mentions $M = \{m_1, m_2, ...m_N\}$ in the document are disambiguated simultaneously to arrive at a $coherent$ set of KB entries $\{c_{1*}, c_{2*}, ..., c_{N*}\}$. *Coherent* set means Collaborative Ranking also maximizes the similarity among the N KB entries c_{1*}, c_{2*}, ... , c_{N*}. For example, if a mention of "*Michael Jordan*" refers to the computer scientist rather than the basketball player, then we would expect a mention of "*Monte Carlo*" in the same document to refer to the statistical technique rather than the location, as statistical techniques have a more close relation with the computer scientist.

Following this idea, Cucerzan (2007) and Radford (2010) extracted all entities in the context of a given query, and simultaneously disambiguated all the entities in the document to Wikipedia. As we know, each Wikipedia article has some category labels at its bottom [6]. Then, their Collaborative Ranking is to maximize the agreement among the category labels associated with the candidate KB entries c_{1*}, c_{2*}, ... , c_{N*}.

Fernandez et al. (2010), Han and Sun (2011) and Ratinov et al. (2011) used

[6] http://en.wikipedia.org/wiki/Help:Category

the Wikipedia hyperlink graph to estimate the coherence among the Wikipedia articles c_{1*}, c_{2*}, ... , c_{N*}.

The approaches (Chen and Ji, 2011; Cassidy et al., 2011) further extended this idea to cross-document level by constructing "collaborators" for each query and exploiting the global context from the entire collaborator cluster for each query. More specifically, given a query, they clustered the query document together with a large document collection on-the-fly. Then, the features for supervised algorithm also contain the cluster-level features such as maximum, minimum and average tfidf/entity similarities between the candidate and the document cluster which contains the query document.

Chapter 3: TRAINING DATA CREATION AND INSTANCE SELECTION

As discussed in Section 2.4, the current state-of-the-art entity linking systems are based on supervised learning approach, which require thousands of annotated mentions to achieve good performance. Figure 3.1 shows some examples of the annotated mentions.

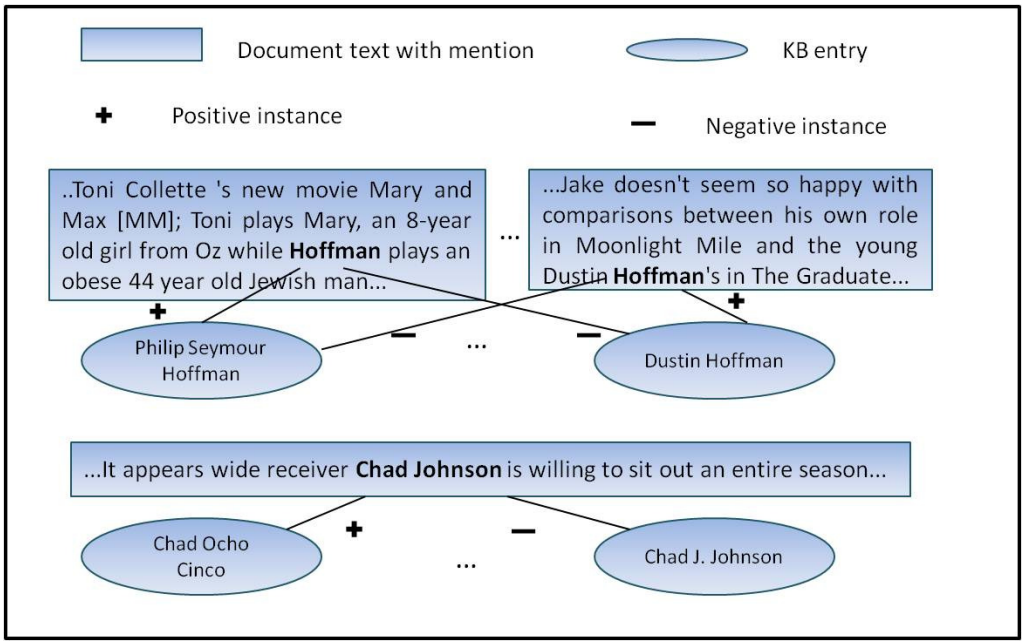


Figure 3.1: Annotated Mentions

Manually creating this training corpus for entity linking is labor-intensive and costly. Entity linking annotation is also highly dependent on the KB. When a KB for a new domain comes, the annotating process needs to be repeated.

Thus, in this chapter, we present a novel method to automatically generate a large scale annotated data set for ambiguous mentions leveraging on their unambiguous synonyms. Furthermore, the distribution of the unambiguous mentions is not consistent with the mentions in the real world. Thus, to utilize the auto-generated data set without compromising the accuracy, an instance selection strategy is proposed to select an informative, representative and diverse subset from the auto-generated dataset. During the iterative selection process, the batch sizes at each iteration change according to the variance of classifier's confidence or accuracy between batches in sequence, which not only makes the selection insensitive to the initial batch size, but also leads to a better performance.

We conduct evaluation on *TAC-10* data (Ji et al., 2010). Experiments show that our method achieves state-of-the-art performance without hard intensive work on annotating thousands of articles. Besides, the instance selection can make the dataset more balanced and it also produces a significant gain in entity linking performance.

3.1 Automatic Data Creation

This section will describe our approach to automatically generate the training set for the supervised learning approaches in candidate ranking (see Section 2.4). The basic idea is to take a document with an unambiguous reference to an entity E1 and replacing it with a phrase which may refer to E1, E2 or others.

Observation: Some full names for the entities in the world are unambiguous. This phenomenon also appears in the given document collection of entity

linking at *TAC-10*. The mentions "*Abbott Laboratories*" appearing at multiple locations in the document collection refer to the same entity "*a pharmaceuticals health care company*" in KB.

From this observation, our method takes into account the name variations of KB entry derived from (1) the title of Wikipedia Entity Pages (e.g. *"Abbott Laboratories"*); (2) the title of Wikipedia Redirect Pages (e.g. *"Abbott Labs"*) (see Section 2.2.2). As these name variations are unambiguous, their mentions in the articles can be linked with the correct KB entry without disambiguation. For example, the mention "*Abbott Laboratories*" can only match the title of KB entry *E0272065*. *E0272065* is the ID of the KB entry for this pharmaceuticals health care company. With the unambiguous name list for each KB entry, our method to annotate training instances for entity linking is as follows.

We first use an index and search tool to find the documents with these unambiguous mentions. For example, the name "*Abbott Laboratories*" occurs in documents *LDC2009T13* and *LDC2007T07* in *TAC* document collection. The chosen text indexing and searching tool is the well-known *Apache Lucene* information retrieval open-source library [1].

Next, to validate the consistency of NE type between entities in KB and in document, we run the retrieved documents through a Named Entity Recognizer, to tag the named entities in the documents. Then we link the document to the entity in KB if the document contains a named entity whose name exactly matches with the unambiguous name and type (i.e. Person, Organization and Geo-Political Entity) exactly matches with the type of entity in KB. For example, after Named Entity Recognition, "*Abbott Laboratories*" in document

[1] http://lucene.apache.org

LDC2009T13 is tagged as an Organization which is consistent with the entity type of *E0272065* in KB. Then, we link the "*Abbott Laboratories*" occurring in document *LDC2009T13* with entity *E0272065* in KB.

Finally, we replaced the mention in the selected documents with its ambiguous synonyms. For example, we replace the mention "*Abbott Laboratories*" in document *LDC2009T13* with an ambiguous synonym "*Abbott*", which can refer to many KB entries such as *E0064214*, *E0272065* and so on. Then, we can obtain the annotated instances for entity linking. Two examples of these instances are as follows, where one is positive and the other is negative.

$$(Abbott, LDC2009T13) \rightarrow E0272065$$
$$(Abbott, LDC2009T13) \nrightarrow E0064214$$

Following this approach, from the 1.7 million documents in *TAC-2010* text collection, we generate 45,000 annotated instances.

3.2 Instance Selection Strategy

In this section, we explore the method to effectively utilize the large-scale data auto-generated in Section 3.1. We generate the data set leveraging the unambiguous names. However, the distribution of the unambiguous mentions can not perfectly represent the real distribution of all the mentions. In the case of "*Abbott Laboratories*", more than ten "*Abbott*" mentions are linked to "*Abbott Laboratories*" entry in KB, but no "*Abbott*" example is linked to other entries like "*Bud Abbott*" "*Abbott Texas*", etc. Thus, we use an instance selection strategy to select a more balanced subset from the auto-annotated instances and reduce the effect of the distribution problem. However, the traditional

instance selection approaches (Brighton and Mellish, 2002; Liu and Motoda, 2002) only can solve two problems: 1) a large dataset causes response-time to become slow 2) the noisy instances affect accuracy, which are different from our needs here. We thus propose a new instance selection approach for selecting a more balanced subset from the auto-annotated instances. This instance selection strategy follows the same principle of active learning (Brinker, 2003; Shen et al., 2004) which aims to reduce the manual annotation effort on training instances through proposing only the useful candidates to annotators. As we already have a large set of automatic generated training instances, the selection here is a fully automatic process to get the useful and more balanced subset. As this process is guided by a development data set, the distribution of the selected instances is consistent with the real data set. Besides, during our iterative selection process, the batch sizes at each iteration change according to the variance of classifier's confidence or accuracy between batches in sequence. The details of our instance selection method is as follows.

We use the SVM classifier (Vapnik, 1995) to select the instances from the auto-generated data set. The initial classifier can be trained on a set of initial training instances, which can be a small part of the whole auto-generated data, or the limited manual annotated training instances available, e.g. those training data provided by *TAC-10*.

Our instance selection method is an iterative process. We select an informative, representative and diverse batch of instances based on current hyperplane and add them to the current training instance set at each iteration to further adjust the hyperplane for more accurate classification.

Schohn and Cohn (2000) show that an instance is informative for the learner

if the distance of its feature vector to hyperplane is not greater than that of the support vectors to the hyperplane. Thus, we use the distance as the measure to select informative instances. The distance of an instance's feature vector to the hyperplane is computed as Eq 3.1.

$$Dis(w) = |\sum_{i=1}^{N} \alpha_i y_i k(s_i, w) + b| \tag{3.1}$$

Where w is the feature vector of the instance, α_i, y_i and s_i correspond to the weight, class and feature vector of the i^{th} support vector respectively. N is the number of the support vectors.k is the kernel function and b denotes an intercept.

Next, we quantify the representativeness of an instance by its density (see Eq 3.2). Such density is defined as the average similarity between this instance and all other instances in the dataset. If an instance has the largest density among all the instances in the dataset, it can be regarded as the centroid of this set and also the most representative instance.

$$Density(w_i) = \frac{\sum_{j \neq i} sim(w_i, w_j)}{N - 1} \tag{3.2}$$

Where w_i is the instance in the dataset and N is the size of dataset. Sim is cosine similarity.

We combine the informativeness and representativeness by the function $\lambda(1 - Dis(w)) + (1 - \lambda)Density(w)$, in which Dis and Density are normalized first. The individual importance of each part in this function is adjusted by a trade off parameter λ (set to 0.5 in our experiment based on the active

learning work (Shen et al., 2004)). The instance with the maximum value of this function will be selected first to the batch. This instance will be compared individually with the selected instances in current batch to make sure their similarity is less than a threshold β. This is to diversify the training instance in the batch to maximize the contribution of each instance. We set β to the average similarity between the instances in the original dataset. When a batch of α instances is selected, we add them to the training instance set and retrain the classifier.

Such a batch learning process will stop at the peak confidence of the SVM classifier, since Vlachos (2008) shows that the confidence of the SVM classifier is consistent with its performance. The confidence can be estimated as the sum of the distances to hyperplane for the instances of an un-annotated development set. The development set guides the selection process to solve the distribution problem mentioned above. Alternatively, we can also leverage on some annotated development data and use accuracy instead to guide the selection process. We explore both approaches for different application scenarios in our experiments.

We now need to decide how to set the batch size α at each iteration. It is straightforward to set a fixed batch size α (**Fixed Number**), which never changes during the process. However, there are some limitations as demonstrated in our experiments of this chapter. First, the performance is sensitive to the batch size. Second, if we set the batch size too big, it will impede further improvement allowed by small batch size. But if we set the batch size too small from the beginning, it will dramatically increase the number of iterations needed which will make the selection too slow. To resolve the above issues,

we change the batch size according to the variance of classifier's confidence on an un-annotated set. Thus, we assign an integer to α_1 and α_2 in the first two iterations, and $\alpha_i (i > 2)$ in the i^{th} iteration is computed as Eq. 3.3 (**Flexible Number**).

$$\alpha_i = \frac{\alpha_{i-1} * (con_{i-1} - con_{i-2})}{con_{i-2} - con_{i-3}} \tag{3.3}$$

where con_i is the confidence of the classifier on the un-annotated dataset at i^{th} iteration.

Algorithm 3.1 summarizes the selection procedure.

3.3 Experiments and Discussions

In our experiments, we use *TAC-10* data set and adopt micro-averaged accuracy (See Section 1.2) to evaluate our auto-generated annotations and instance selection strategy. For pre-processing, we perform sentence boundary detection derived from Stanford parser (Klein and Manning, 2003), named entity recognition using a SVM based system trained and tested on ACE 2005 with 92.5(P) 84.3(R) 88.2(F), and co-reference resolution using a SVM based resolver trained and tested on ACE 2005 with 79.5%(P), 66.7%(R) and 72.5%(F). In our implementation, we use the binary SVM^{Light} developed by Joachims (1999), and the classifier is trained with default parameters.

3.3.1 With and Without Manual Annotated Data

Table 3.1 shows the results for evaluating our auto-generated data set and the instance selection strategy with Flexible Number Scheme. For instance selec-

Algorithm 3.1 Instance Selection Strategy

Input: Initial Training Set $T = \{T_1, T_2, ..., T_m\}$
Original Set where we select instance $A = \{A_1, A_2, ..., A_n\}$
$BatchSet$ with the maximal size α

Output: Training Set $T = \{T_1, T_2, ..., T_m\}$

Initialization: $BatchSet = \phi$

Loop until the confidence/accuracy of the classifier on a development set does not increase

(a) Train a Classifier on T

(b) $BatchSet = \phi$

(c) Update α according to Eq. 3.3

(d) **Loop** until $BatchSet$ is full

i. Select A_i with maximal value P from A
$P = \lambda(1 - Dis(w)) + (1 - \lambda)Density(w)$

ii. RepeatFlag=false

iii. **Loop** for each A_k in $BatchSet$

- **If** $Sim(A_i, A_k) > \beta$ **Then**,
  - RepeatFlag=true
  - Stop the **Loop**

iv. **If** RepeatFlag==false **Then**,

- Add A_i to $BatchSet$
- Remove A_i from A

(e) $T = T \bigcup BatchSet$

tion, this set of experiments check the variance of classifier's confidence on the unannotated test set to stop the process of selection and to decide the batch size α of Flexible Number Scheme.

Methods	ALL	NIL	in KB	ORG	GPE	PER
Auto_Gen	81.2	81.8	80.5	80.8	72.5	90.3
Auto_Gen+IS	85.2	87.5	82.5	84.4	78.5	92.8
TAC	83.2	88.2	77.2	82.1	75.1	92.5
TAC+Auto_Gen	82.2	83.8	80.4	81.7	75.6	89.5
TAC+Auto_Gen+IS	85.5	87.7	82.9	84.7	78.9	92.8

Table 3.1: Results of Entity Linking for Instance Selection

We first evaluate the effectiveness of our instance selection strategy if no manually annotated data is available. In the first block of Table 3.1, we compare the performances of the systems with and without instance selection. *"Auto_Gen"* uses the auto-generated dataset described at Section 3.1 as the training set directly, and *"Auto_Gen+IS"* applies our instance selection to the auto-generated data for training. In the instance selection process, we use the KB entries with more than 15 linked documents in the auto-generated data as our Initial Training Set (1,800 instances) to train a classifier, and then use this classifier to select instance from the auto-generated dataset. The first block of Table 3.1 shows that our instance selection gives significant improvements ($\rho < 0.05$, χ^2 test). These improvements show our selection strategy makes the training set more balanced and it can effectively reduce the effect of distribution problem that the auto- generated data set only contains unambiguous mentions.

We further evaluate our instance selection strategy when a large manually annotated data is available in the second block of Table 3.1. *"TAC"* is trained

on the manually annotated *TAC-10* training set. *"TAC+Auto_Gen"* is trained on *TAC-10* set and the auto-generated set. *"TAC+Auto_Gen+IS"* uses *TAC-10* training set as the Initial Training Set, and applies instance selection process to the auto-generated data. Comparing *"TAC+ Auto_Gen"* with *"TAC"*, we can see that the unbalanced distribution caused serious problem which even pull down the performance achieved by the large manual annotation alone. The experiment results of *"TAC"* and *"TAC+Auto_Gen+IS"* show that our instance selection strategy appears very necessary to bring further improvements over the large manually annotated dataset (5,404 instances). These significant ($\rho < 0.05$,χ^2 test) improvements are achieved by incorporating more training instances in a reasonable way.

Comparing the performance of *"Auto_Gen+IS"* with *"TAC"* in Table 3.1, we can find that our method performs better without hard intensive work on annotating 5,404 articles. This proves that using our instance selection can save labor without compromise of entity linking accuracy. The pretty much same performance of *"Auto_Gen+IS"* with *"TAC+Auto_Gen+IS"* also confirms the above conclusion.

3.3.2 Fixed Size Vs. Changing Size

We are also interested in the effectiveness of the two schemes (i.e. Fixed Number and Flexible Number) of setting the batch size α mentioned in Section 3.2. In Figure 3.2, we set the batch size α in Fixed Number scheme and α_1, α_2 in Flexible Number scheme, to different numbers from 50 to 140 increasing 10 each time. We conduct instance selection to the auto-generated data. Figure 3.2 shows that flexible batch size outperforms the fixed size for entity linking. Es-

pecially, the improvement at $\alpha = 50, 60$ and 70 is significant ($\rho < 0.05$,χ^2 test). This proves that batch size should be in line with the variance of the classifier's confidence at each iteration of instance selection. Furthermore, in this Figure, the performance of flexible batch size is more stable than the Fixed Number scheme. This shows that Flexible Number scheme makes the entity linking system insensitive to the initial batch size during instance selection process. Thus the initial batch size of the experiments in Table 3.1 is set to 80, which we believe that very similar performance can be achieved even with a different initial size. Another fact is that the selection process is similar to active learning, which needs to manually annotate the selected instances in each batch. Thus, being a generic approach, the batch size changing method proposed in this chapter can also benefit active learning for other tasks.

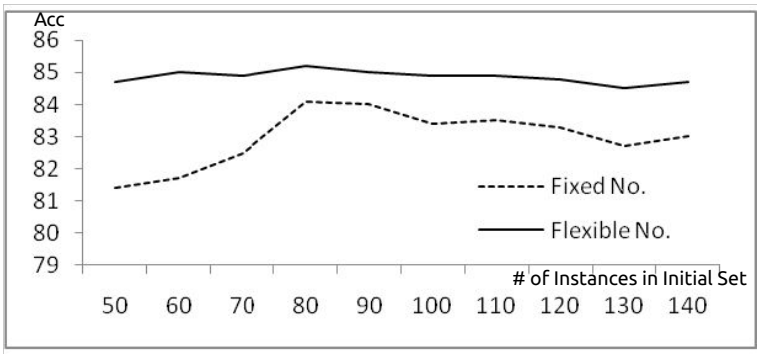


Figure 3.2: Performance Curves for Two Batch Size Schemes

3.4 (Un-)Annotated Development Set

In the above study, we directly use the test set without annotations as the development set for instance selection to optimize our solution to the application data. Such an approach will be useful when the application set is available in

advance as in the case with TAC benchmarks. When the application set is unavailable beforehand, in other words, the articles to be linked only arrive one after the other in linking stage, we leverage on the accuracy on annotated development set for the instance selection. Figure 3.3 shows the performances on different sizes of annotated development set. The results show that the different sizes contribute more or less same performances. We only need to use a small amount of annotated development data, 500 articles in our study to guide the instance selection to achieve similar performance as with unannotated test set being development data.

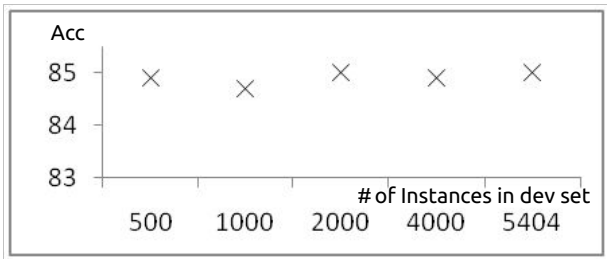


Figure 3.3: Performance for Annotated Development Data

3.5 Conclusions

In this chapter, we create a large corpus for entity linking by an automatic method. Furthermore, we proposed a batch size changing instance selection strategy to reduce the effect of distribution problem in the auto-generated data. It makes entity linking system achieve state-of-the-art performance without hard labor. Meanwhile, the flexible batch size not only makes the selection insensitive to the initial batch size, but also leads to a better performance than the fixed batch size. Being a generic approach, the batch size changing method proposed in this Section can also benefit active learning for other tasks.

Chapter 4: TOPICAL FEATURES FOR ENTITY LINKING

In the last chapter, we discussed the training instance generation for the supervised learning in entity linking. This chapter focuses on the feature design for the supervised learning in entity linking.

For the feature design in candidate ranking (see Section 2.4), there has been much existing work which demonstrates modeling context is an important part of measuring the similarity between query text and the KB candidate. However, the traditional approach for entity linking treats the context as a bag of words, n-grams, noun phrases or/and co-occurring named entities, and measures context similarity between query document and KB entry by the comparison of the weighted literal term vectors. Such literal matching suffers from sparseness issue. For example, consider the following four sentences with mention *Michael Jordan*:

1. *Michael Jordan* is a leading researcher in machine learning and artificial intelligence.

2. *Michael Jordan* is currently a full professor at the University of California, Berkeley.

3. *Michael Jordan* (born February, 1963) is a former American professional basketball player.

4. *Michael Jordan* wins NBA MVP of 91-92 season.

As there is no shared term among these sentences, literal matching can not correctly disambiguate the four *Michael Jordan* mentions. In this example, the semantic knowledge underlying the words is needed. If we can detect that Sentence 1 and Sentence 2 are related with topic "academic", but Sentence 3 and Sentence 4 are related with topic "sports", we would correctly link the mentions referring to the same person together. Topic modeling approaches have been introduced to the similar task cross-document coreference such as (Kozareva and Ravi, 2011).

In this chapter, we propose a Wikipedia-LDA model to effectively mine the semantic knowledge from the contexts of the mentions. Such topic model allows us to measure the similarity between articles and KB entries in the semantic space of hidden topics.

We conduct evaluation on *TAC-10* data (Ji et al., 2010). Experiments show that the Wikipedia-LDA model is able to effectively capture the underlying semantic information and produce statistically significant improvement over literal matching alone.

4.1 Latent Dirichlet Allocation (LDA)

LDA (Blei et al., 2003) defines a *topic* underlying a document collection to be a distribution over a fixed vocabulary. For example, the *genetics* topic has words about genetics with high probability and the *evolutionary biology* topic has words about evolutionary biology with high probability. Then, the documents are represented as random mixtures over the $topics$. More formally, with the following definition,

- A word is defined to be an item from a vocabulary indexed by $\{1, ..., V\}$.

- A document is a sequence of N words denoted by $W = (w_1, w_2, ..., w_N)$, where w_n is the nth word in the sequence.
- A corpus is a collection of M documents denoted by $D = \{W_1, W_2, ..., W_M\}$

LDA defines the the following process to generate each document W in a corpus D (see Figure 4.1):

1. Choose $N \sim Poisson(\varepsilon)$.
2. Choose $\theta \sim Dir(\alpha)$.
3. For each of the N words w_n :
   (a) Choose a topic $z_n \sim Multinomial(\theta)$.
   (b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Note that the dimensionality k of the Dirichlet distribution (and thus the dimensionality of the topic variable z) is assumed known and fixed, and the word probabilities are parameterized by a $k \times V$ matrix β where $\beta_{ij} = p(w^j = 1|z^i = 1)$.

For our task, we train LDA models on Wikipedia texts of KB, where the text of each entry is treated as a document. Once the model is trained, we map the document where the name mention appears and the text of KB entry, to the hidden topic space by calculating the topic proportions θ. Then, the probability over each topic for KB entry and the query document is learned. Thus, we can calculate the context similarity in the *K*-dimensional topic space by their Hellinger distance as Eq. 4.1.

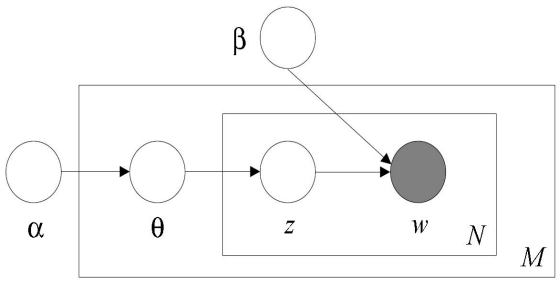


Figure 4.1: Graphical Model Representation of Latent Dirichlet Allocation

$$Similarity_{d,e} = \sum_{k=1}^{K} \left(\sqrt{\overrightarrow{\theta_{d,k}}} - \sqrt{\overrightarrow{\theta_{e,k}}} \right)^2 \tag{4.1}$$

Finally, such semantic similarity can be combined with other term matching features to SVM ranker and classifier for entity linking.

4.2 Wikipedia-LDA Model

The number k of topics in Latent Dirichlet Allocation (LDA) have to be defined by human experience before the learning process, and the topics are hidden. In fact, Wikipedia has explicitly defined the topics of articles as Wikipedia categories [1]. For example, the category labels for the Wikipedia article of basketball player "*Michael Jordan*" are *African-American basketball players, Shooting guards, Baseball players from New York* and so on. As manually defined for Wikipedia, they are better and more suitable to model our KB topics. Our experiments also prove that using Wikipedia categories as the topics can further improve entity linking systems.

In the similar task cross-document coreference (Han and Zhao, 2009) and

[1] http://en.wikipedia.org/wiki/Help:Category

other tasks (e.g. text classification) (Wang and Domeniconi, 2008), Wikipedia concepts are used to model the text. However, Wikipedia concept is a kind of entity-level topic. This differs from our approach, where we use the cross-entity topic Wikipedia Categories to represent the semantic knowledge.

In this Section, we model the contexts as the distributions over Wikipedia categories. Then, the similarity between the contexts can be measured in a semantically meaningful space. Finally, such semantic similarity, together with other base features, is incorporated in the trainable models to learn the ranker and classifier.

4.2.1 Modeling the Contexts as Distributions over Wikipedia Categories

Wikipedia requires contributors to assign categories to each article, which are defined as "major topics that are likely to be useful to someone reading the article". Thus, Wikipedia can serve as a document collection with multiple topical labels, where we can learn the posterior distribution over words for each topical label (i.e. Wikipedia category). Then, from the observed word in the context of mention and KB entry, we can estimate the distribution of the contexts over the Wikipedia categories. To obtain this distribution, we use a supervised Latent Dirichlet Allocation (LDA) model-labeled LDA defined by Ramage et al. (2009), which represents state-of-the-art method for multi-labeled text classification. It performs better on collections with more semantically diverse labels, which we need in order to leverage on the large semantically diverse categories from Wikipedia as the topical labels.

Figure 4.2 shows us a graphical representation of the labeled LDA for the multi-labeled document collection. Labeled LDA is a three level hierarchical

Bayesian model. β is the multinomial distribution over words for a Wikipedia category, which has a Dirichlet prior with hyperparameter η. Both the category set Λ as well as the topic prior α influence the topic mixture θ. These distributions can be used to generate documents in the form of a collection of words (w). D is the number of documents, N is the document length and K is the number of categories.

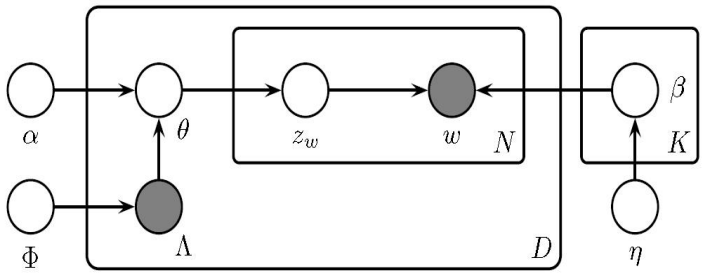


Figure 4.2: Graphical Model Representation of Labeled Latent Dirichlet Allocation

After the model is trained by Wikipedia data, the distributions of KB entry and the article over K categories are estimated by calculating the topic proportions θ. θ is given by an EM procedure that treats θ as a parameter with Z missing.

4.2.2 Context Similarity

We have mapped the contexts to a K-dimensional semantic space. Thus, we can calculate the context similarity by their distance in this space. To measure the context similarity in the K-dimensional topical space, we calculate the Cosine value as Eq. 4.2.

$$Similarity_{d,e} = \frac{\sum_{k=1}^{K} \theta_{d,k} \times \theta_{e,k}}{\sqrt{\sum_{k=1}^{K} (\theta_{d,k})^2} \times \sqrt{\sum_{k=1}^{K} (\theta_{e,k})^2}} \tag{4.2}$$

Where d means the document with the name mention and e means the KB entry. Such semantic similarity can be further combined with other term matching features for SVM ranker and classifier of entity linking.

4.2.3 Wikipedia Category Selection

Each article in Wikipedia is assigned several categories by the contributors as requested. However, from our observation some categories in Wikipedia may not be suitable to model the topics of a document. Thus, we shall consider selecting an appropriate subset from the Wikipedia categories to effectively model the contexts. We examined five possible category subsets:**all**, **all-admin**, **isa_all**, **isa_class**, and **isa_instance**.

Wikipedia contains 165,744 categories. This is the set **all**.

There are some meta-categories used for encyclopedia management. For example, "Wikipedia editing guidelines", which are unsuitable to describe the topics of a document. Thus, we remove the categories which contain any of the following strings: *wikipedia, wikiprojects, lists, mediawiki, template, user, portal, categories, articles and pages*. This leaves 127,325 categories (**all-admin**).

Besides, some categories such as "*River by Country*" and "*Geography by place*" in the **all-admin** set are still redundant, because all the categories in *is-a* relation can serve as the knowledge graph of the world. For example, the relation between the two topical categories "*Singapore River*" and "*River*" is an

is-a relation. These categories have covered topical information in the category "*River by Country*" which is not in any is-a relation. We thus only select the categories connected by *is-a* relation to **isa_all** subset.

Since the categories have been connected by unlabeled links in Wikipedia, we just need to identify those links representing *is-a* relation. We use the four methods as below proposed by Ponzetto and Strube (2007) to distinguish *is-a* and *not-is-a* relation links.

We first use a syntax-based method: assign *is-a* to the link between two categories if they share the same lexical head lemma (e.g. "*British Computer Scientists*" and "*Computer Scientists*").

Then, we use structural information from the category network: (1) for a category c, look for a Wikipedia article P with the same name, for example the article page "*Microsoft*" and the category "*Microsoft*". Take all $P's$ categories whose lexical heads are plural nouns $CP = \{cp_1, cp_2, ..., cp_n\}$. Take all super-categories of c, $SC = \{sc_1, sc_2, ..., sc_k\}$. If the head lemma of one of cp_i matches the head lemma of sc_j, label the relation between c and sc_j as *is-a*. For instance, the article "*Microsoft*" being categorized into "*Companies listed on Nasdaq*" indicates that "*Microsoft*" is a company. Then, assign *is-a* to the link between "*Microsoft*" and its super-category "*Computer and video game companies*". (2) assign *is-a* label to the link between two categories if a Wikipedia article is redundantly categorized under both of them. For example, "Internet" is categorized under both "*Computer networks*" and "*Computing*" and there is a link between "*Computer networks*" and "*Computing*". Then this link is assigned *is-a*.

Next, we consider lexical-syntactic patterns in a corpus. This method uses

two sets of patterns. One set is used to identify *is-a* relations (Caraballo, 1999; Hearst, 1992), for example "such NP_1 as NP_2", NP_1 and NP_2 are the values of categories and their subcategories respectively. The second set is used to identify *not-is-a* relations. For example "NP_1 has NP_2", where the link between NP_1 and NP_2 will be assigned *not-is-a*. These patterns are used with a corpus built from Wikipedia articles, and separately with the Tipster corpus (Harman and Liberman, 1993). The label is assigned by majority voting between the frequency counts for the two types of patterns.

Finally, we assign is-a labels to links based on transitive closures - all categories along an *is-a* chain are connected to each other by *is-a* links.

Another fact is that the categories defined by Wikipedia are entities, not all classes. For example, "*Microsoft*" is an instance of the class "*Computer and Video Game Companies*", and it appears both as an article page and as a category in Wikipedia. We would like to further examine the two different subsets: **isa_class**, and **isa_instance** in **isa_all** set for entity linking. To distinguish instance and class in **isa_all** set, we use a structure-based method (Zirn et al., 2008). The categories which have other sub-categories or Wikipedia articles connected to them by *is-a* relation are assigned class label. In our problem, the remaining categories are approximately regarded as instances.

4.3 Experiments and Discussions

In our study, we use *TAC-10* data set and adopt micro-averaged accuracy to evaluate our Entity Linker. For pre-processing, we perform sentence boundary detection derived from Stanford parser (Klein and Manning, 2003), named entity recognition using a SVM based system trained and tested on ACE 2005

with 92.5(P) 84.3(R) 88.2(F), and co-reference resolution using a SVM based resolver trained and tested on ACE 2005 with 79.5% (P), 66.7% (R) and 72.5% (F). In our implementation, we use the binary SVM^{Light} developed by Joachims (1999), and the classifier is trained with default parameters. The Stanford Topic Model Toolbox [2] is used for Labeled-LDA with default learning parameters.

Table 4.1 lists the performance of entity linking with overall accuracy (ALL) as well as accuracy on subsets (Nil, in KB, ORG,GPE and PER) of the data. In the first row, only base features described in Table 2.1 are used. This baseline system models the contexts with literal terms. The second row shows the accuracy combining base features with the hidden topical features learned by LDA in Section 4.1. The third to seventh rows report the results combining base features with topical knowledge (i.e. the context similarity is computed under the topic space of the five different subsets of Wikipedia categories described in Section 4.2.3).

Features	ALL	NIL	inKB	ORG	GPE	PER
Base Features	83.2	88.2	77.2	82.1	75.1	92.5
Base + Hidden Topics	84.5	81.4	87.1	92.7	82.7	78.1
Base + all	84.0	88.6	78.5	84.0	76.0	92.1
Base + all-admin	84.9	88.9	80.0	84.9	76.9	92.8
Base + isa_all	**85.9**	89.1	82.0	85.2	78.6	93.8
Base + isa_class	85.5	88.8	81.3	84.9	78.0	93.2
Base+isa_instance	83.9	88.9	77.8	82.9	76.6	92.1

Table 4.1: Results of Entity Linking for Topical Features

We see that all the six systems with topical features perform better than the baseline system, which models the context similarity as literal term matching. Especially, the **isa_all** and **isa_class** can achieve significantly better result

[2] http://nlp.stanford.edu/software/tmt/tmt-0.3/

than the baseline ($\rho < 0.05, \chi^2$test). These results prove that the semantic knowledge underlying the contexts has good disambiguation power for entity linking. Table 4.2 tells the reason of the improvements. Table 4.2 shows us four sample Wikipedia categories and top 15 highly probable words identified by the topic model for these categories. The topic model successfully assigns a high probability to the words "*researcher*" and "*professor*" in the category "*Members of the National Academy of Sciences*", and assign a high probability to the words "*nba*" "*basketball*" "*professional*" and "*season*" in the category "*American basketball players*". Such topical knowledge learned from Wikipedia data is helpful in the example of "*Michael Jordan*" mentioned at the beginning of this chapter. This shows that entity linking can benefit from the topical information underlying the words and overcome the shortcomings of literal matching

We further compare the performances of the five different category subsets. From the last five rows of Table 4.1, we can see that **isa_all** subset performs best among the five subsets for disambiguation. This should be because **isa_all** includes more categories than **isa_class** and **isa_instance**, and thus can capture more semantic information. However, although **all** and **all-admin** include even more categories, they introduce many categories which are unsuitable to model the topics of a news article or blog text, such as the two categories mentioned in Section 4.2.3, "*people by status*" which is not in an *is-a* relation and "*Wikipedia editing guidelines*" which is used for encyclopedia management.

Finally, we would like to compare our Wikipedia-LDA with LDA model. Comparing *Base + isa_all* with *Base + Hidden Topics* in Table 4.1. We can conclude that Wikipedia categories are better and more suitable to model our

American novels	American film actors	Members of the National Academy of Sciences	American basketball players
novel	role	prize	**nba**
book	actor	**researcher**	**basketball**
story	films	**professor**	points
paperback	appeared	science	rebounds
plot	television	nobel	games
print	hollywood	institute	draft
edition	california	theory	guard
isbn	roles	physics	overall
hardback	movie	received	coach
characters	acting	sciences	**professional**
published	married	medal	assists
man	death	chemistry	play
father	character	academy	**season**
love	starred	award	forward
written	actress	ph.d	ncaa

Table 4.2: Sample Wikipedia Categories and Corresponding Top 15 Words

KB topics, since these categories are manually defined for Wikipedia.

4.4 Conclusions

In this chapter, we explored using two innovative approaches for entity linking to address the sparseness issue of literal matching. First, we introduce LDA to entity linking, which can discover the semantic knowledge underlying the contexts. Since the number of topics learned by LDA have to be defined by human experience before the learning process, and the topics are hidden, we also proposed a Wikipedia-LDA to model the topics of texts, where we investigated the effectiveness of five subsets from Wikipedia categories to represent the underlying topics.

Chapter 5: LAZY LEARNING FOR ENTITY LINKING USING QUERY SPECIFIC INFORMATION

In the previous chapters we have discussed the training data and features for the supervised learning algorithm. In this chapter, I will discuss the learning framework. Figure 5.1 shows the learning framework of previous entity linking systems where they start with a training data set then train a linker, and finally use the learned linker to predict the query.

However, as there are infinite number of entity names, it is impossible to manually create the labeled data set for each name. The available labeled data for entity linking is only for a certain number of names. Thus, as shown in Figure 5.1, the query is a mention of the name *"AZ"*, but the names in training data set are *"Hoffman"*, *"Chad Johnson"* and so on. In other words, the existing approaches disambiguate a mention of the name (e.g.*"AZ"*) based on the distribution knowledge learned from the labeled mention-KB_entry pairs in the training set $\boldsymbol{M}$ related to other names (e.g.*"Hoffman"*,*"Chad Johnson"*, etc.).

Figure 5.2 illustrates the locations of the labeled instances related to the three names *"AZ"*, *"Hoffman"* and *"Chad Johnson"* in a feature space (*Bag of Words*, *Named Entities* and *Edit Distance*, the popular features used in entity linking). We can see that the location of the hyperplane to separate positive and negative instances for different names vary widely. Moreover, the positive-

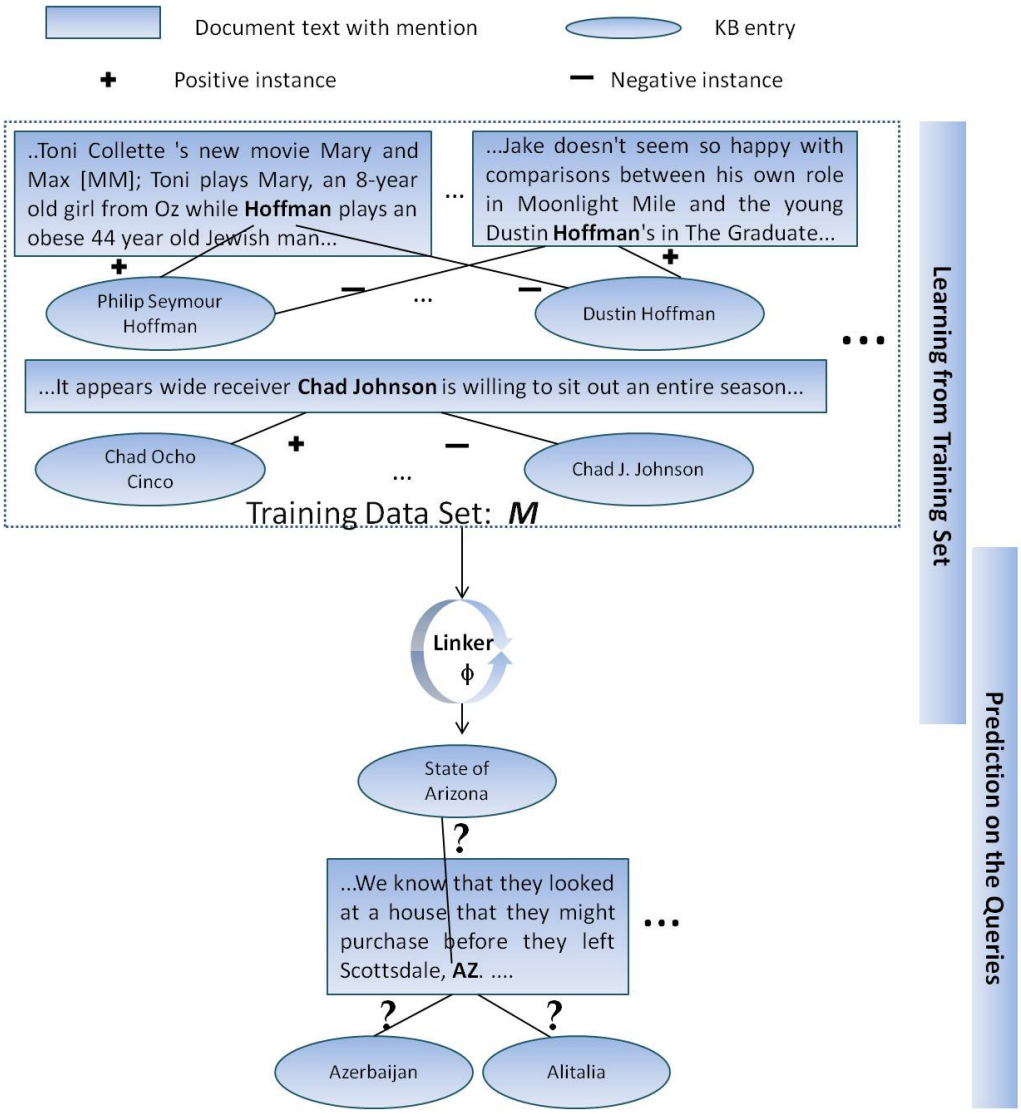


Figure 5.1: The System Architecture for Traditional Approaches. (*M* contains a certain number of names. *"Hoffman"* and *"Chad Johnson"* are two examples of them.)

negative instance ratio of each name is also very different from others. Thus, the entity linker generalized beyond labeled names (*"Hoffman"*, *"Chad Johnson"*, etc.) without considering the knowledge of the queried name (*"AZ"*) suffers from this distribution gap.

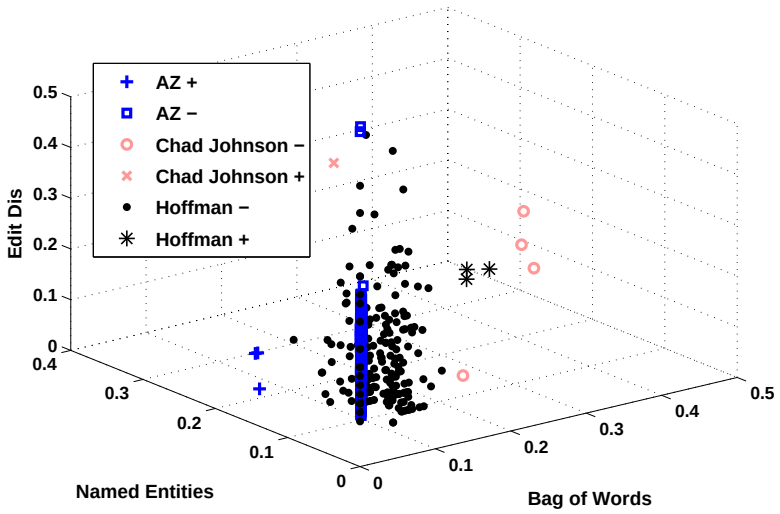


Figure 5.2: Instances Illustration in 3D Feature Space (Feature detail is in Table 2.1)

To narrow down the gap between the instance distributions related to labeled and queried names, this chapter proposes a lazy learning model, in which generalization on the labeled data is delayed until a query is made. This allows the training instances specific to queried name to be incorporated into the learning process. To obtain these training instances, our lazy learning model automatically labels some relevant instances for the queried name leveraging its unambiguous synonyms.

In addition to the new notion of benefiting from the auto-generated instances related with the queried name, our approach further benefits from the

manually labeled data related to other names. Specifically, the learned linker generalizes on the labeled data sets related to both queried and other names by exploiting the inherent predictive structure shared by these two data sets.

In the task of *Online Reputation Management* (Amigo et al., 2010; Spina et al., 2011) mentioned in Section 1.4, Amigo et al. (2010) concluded that it was not viable to train separate system for each of the companies, as the system must immediately react to any imaginable company name. Thus, in this benchmark, the set of company names in the training and test corpora are different. In contrast, the lazy learning approach proposed in this chapter demonstrates that it is feasible to train separate system for each company, and the system can immediately react to any company name without manually labeling new corpora.

We conduct evaluation on *TAC-10* data. Our experiments show that our proposed lazy learning model significantly improves entity linking over the traditional supervised learning framework.

5.1 Architecture of Lazy Learning

We formalize the disambiguation task as follows. We are given a query q (i.e. a document d_q with a mention m_q) and its associated KB candidates C_q=$\{c_1,..,c_N\}$ generated in Section 2.3, and our goal is to select the correct KB entry e from the set C_q. Specifically, let $\phi_q(q,c_i)$ be a score function reflecting the likelihood that the candidate c_i is the correct KB entry for q. Then, a disambiguation model is to solve the following optimization problem:

$$e = \arg\max_{c_i \in C_q} \phi_q(q, c_i) \tag{5.1}$$

To solve this optimization problem. As shown in Figure 5.3, our process of lazy learning model is as follows:

1: Automatically label an instance set A_q based on the query q and candidates C_q.

2: Generalize a function ϕ_q on the data set A_q related with the queried name and a manually labeled data set M related with other names.

3: Select the correct KB entry e from the candidates using the function ϕ_q.

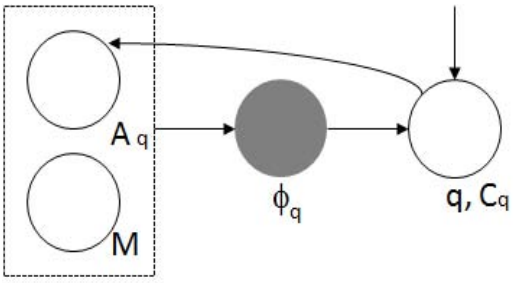


Figure 5.3: Graphical Representation of Lazy Learning

As shown in Figure 5.1, previous approaches generalize a universal linker ϕ for all of the queries on a labeled data set $\boldsymbol{M}$ related to irrelevant names (*"Hoffman"*, *"Chad Johnson"*, etc.), and they suffer from the distribution gap shown in Figure 5.2. In contrast, our lazy learning approach delays the model generalization until receiving the query. It can generalize a separate function ϕ_q for each query leveraging the distribution knowledge learned from the instances in A_q. As A_q is automatically labeled for the queried name, it can be used to narrow down the gap of the instance distributions related to different names

shown in Figure 5.2. Besides, ϕ_q also benefits from $\boldsymbol{M}$ in our model by mining its predictive information shared with A_q. Now, let us elaborate the method for generating A_q, and the generalization of ϕ_q, respectively.

5.2 Training Instances A_q for Queried Name

In this section, we propose to label the instances A_q for the queried name. Following the training data creation approach discussed in Section 3.1 which automatically generates training data for entity linking, we automatically label the instances related to the queried name.

Given a document d_q with a mention m_q and its associated KB candidates C_q, for example,

> $\boldsymbol{d_q}$ **($\boldsymbol{m_q}$="AZ")**: *...We know that they looked at a house that they might purchase before they left Scottsdale,* **AZ**. ...;
>
> $\boldsymbol{C_q}$: *{c_1: state of Arizona, c_2: Azerbaijan, ..., c_N: Alitalia}*,

automatically creating the labeled set A_q for the name *"AZ"* requires automatically linking some mentions of *"AZ"* in text with the KB candidates in C_q. Our approach performs this linking based on two facts: (a) the title of the KB entry is unambiguous (e.g. *"state of Arizona"*). (b) The name variations of KB entry derived from *"redirect pages"* of Wikipedia in Section 2.2.2 are unambiguous (e.g. *"The Grand Canyon State"*). Then, we can generate the unambiguous name variation list for each candidate in C_q (see Table 5.1).

Because the unambiguous name only refers to one KB entry, we can link unambiguous name appearing in a document with the correct KB entry directly without human labor. Thus, we search the documents with these unambiguous

c_1	*state of Arizona; The Grand Canyon State; US-AZ; 48th State; AZ (U.S. state); The Copper State; Arizona, United States; ...*
c_2	*Azerbaijan; Azerbaidzhan; Republic of Azerbaijan; Azerbaijan Republic; Azerbaijani independence; Azerbaijan (Republic); ...*
...	
c_N	*Alitalia; Alitalia Airlines; Alitalia airways; Alitalia.it; Alitalia S.p.A.; ...*

Table 5.1: Unambiguous Variations for the Candidates of *"AZ"*

name variations from a large document collection. Two examples of the retrieved documents are as below:

d_1 (m_1="The Grand Canyon State"): ... ***The Grand Canyon State*** *will get its shot to host the big game a year from now, ...*

d_2 (m_2="Azerbaijan Republic"): ... It is located 30 km east of Ardebil and on the borderline with ***Azerbaijan Republic****. ...*

We denote the labeled instance as a 4-tuple *(d,m,e,+1/-1)*, which means mention *m* in document *d* can/cannot be linked with KB entry *e*. Then, the two unambiguous examples above can be labeled as *(d_1, m_1, c_1, +1)* and *(d_2, m_2, c_2, +1)* automatically.

As we need to label the instances related to the name *"AZ"*, we further replace the unambiguous names in the documents with their ambiguous synonyms *"AZ"*. Then d_1 and d_2 are converted to:

*d_1' (m_q="AZ"): ...**AZ** will get its shot to host the big game a year from now, ...*

*d_2' (m_q="AZ"): ...It is located 30 km east of Ardebil and on the borderline with **AZ**. ...*

Finally, the labeled data set A_q for the queried name *"AZ"* is generated, where A_q=*{(d_1', m_q, c_1, +1),(d_1', m_q, c_2, -1), ...,(d_1', m_q, c_N, -1),(d_2', m_q, c_1, -1),(d_2', m_q, c_2, +1), ...,(d_2', m_q, c_N, -1) ...}*.

5.3 Linear Function ϕ_q

In this section, we formulate the disambiguation function ϕ_q in Eq. 5.1 as follows,

$$\phi_q(q, c_i) = u^T \mathbf{X}_i \tag{5.2}$$

where the document d_q with a mention m_q and the candidate c_i in C_q are represented as a feature vector $\boldsymbol{X}_i \in \chi$, and u is a weight vector.

Estimate u on A_q. A popular method for finding u is *empirical risk minimization* with *least square regularization*. In this work, given a training set A_q=*{(d_i, m_q, e_i, Y_i)}*$_{i=1,...,n^{(q)}}$ ($Y \in$ *{+1,-1}*) related to the queried name m_q, firstly we constrct the feature vector $\boldsymbol{X}_i^q$ for the instance *(d_i, m_q, e_i)*. Then, A_q=*{($\boldsymbol{X}_i^q$, Y_i^q)}*$_{i=1,...,n^{(q)}}$, ($\boldsymbol{X} \in \chi$, $Y \in$ *{+1,-1}*). Finally, we aim to find the weigh vector u that minimizes the empirical loss on the training data,

$$\hat{u} = \arg\min_u \left(\frac{1}{n^{(q)}} \sum_{i=1}^{n^{(q)}} L(u^T \mathbf{X}_i^q, Y_i^q) + \lambda \|u\|^2 \right) \tag{5.3}$$

where L is a loss function. We use a modification of the Huber's robust loss function: $L(p, y) = (max(0, 1 - py))^2$, if $py \geq$ *-1*; and *-4py* otherwise. We fix the regularization parameter λ to 10^{-4}.

Feature Vector *X* for Instance *(q,c_i)*. The features we adopted to construct $\boldsymbol{X}_i$ from *(q,c_i)* include four groups, contextual features (**CF**), semantic features (**SeF**) [1], surface features (**SuF**) and generation source (**GS**) (see Table 2.1).

5.4 Incorporate *M* to *u* Estimation

A practical issue that arises in estimating u only on A_q is the paucity of labeled instances for some queries. This is because we automatically label the instances A_q leveraging its unambiguous synonyms (see Section 5.2). However, for some queried names, it is hard to find a sufficient number of unambiguous synonyms or the related documents containing these synonyms. On the other hand, the total number of available manually labeled instances *M* for other irrelevant names is relatively large. To illustrate the role of M in learning, consider the disambiguation of the two mentions "CPC" and "NY" in two documents. If the first mention "CPC" refers to entity "Communist Party of China" and the second mention "NY" refers to entity "the city of New York", they have similar surface features (e.g. feature "acronym matching" is true). Such surface features effective for linking to "Communist Party of China" may be also effective for disambiguating "NY", and vice versa.

However, with the gap in other aspects between the distributions of A_q and *M* shown in Figure 5.2, directly adding *M* to our training set will produce a lot of noise with respect to the queried name. Thus, instead of using all the distribution knowledge in *M*, we propose to only incorporate the shared knowledge with A_q from *M* into *u* estimation based on *structural learning*.

[1] It also includes the topic feature using Wikipedia categories as the topics described in Chapter 4

5.4.1 The Structural Learning Algorithm

Structural learning (Ando and Zhang, 2005a) is a multi-task learning algorithm that takes advantage of the low-dimensional predictive structure shared by multiple related problems. Let us assume that we have *K* prediction problems indexed by $l \in \{1, .., K\}$, each with $n^{(l)}$ instances ($\mathbf{X}_i^l$, Y_i^l). Each $\boldsymbol{X}_i^l$ is a feature vector of dimension *p*. Let Θ be an orthonormal $h \times p$ (*h* is a parameter) matrix, that captures the predictive structure shared by all the *K* problems. Then, we decompose the weight vector $\boldsymbol{u}_l$ for problem l into two parts: one part that models the distribution knowledge specific to each problem l and one part that models the common predictive structure,

$$\mathbf{u}_l = \mathbf{w}_l + \Theta^T \mathbf{v}_l \tag{5.4}$$

where $\mathbf{w}_l$ and $\mathbf{v}_l$ are weight vectors specific to each prediction problem l. Then, the parameters Θ, $\mathbf{w}_l$ and $\mathbf{v}_l$ can be learned by *joint empirical risk minimization*, i.e., by minimizing the joint empirical loss of the predictors for the *K* problems on the training instances as Eq. 5.5,

$$\underset{\Theta, \mathbf{w}_l, \mathbf{v}_l}{\arg\min} \sum_{l=1}^{K} \left(\frac{1}{n^{(l)}} \sum_{i=1}^{n^{(l)}} L\left(\left(w_l + \Theta^T v_l \right)^T \mathbf{X}_i^{(l)}, Y_i^{(l)} \right) + \lambda \|w_l\|^2 \right) \tag{5.5}$$

It shows that $\mathbf{w}_l$ and $\mathbf{v}_l$ are estimated on $n^{(l)}$ training instances of problem l. In contrast, Θ is estimated on all the training instances of the K problems. This is the key reason why structural learning is effective for learning the predictive structure shared by multiple prediction problems.

5.4.2 Alternating Structure Optimization

The Θ optimization problem in Eq. 5.5 can be approximately solved by the following alternating structure optimization procedure (Ando and Zhang, 2005b),

> 1: Learn K weight vectors u'_l for all the K problems on their corresponding instances independently using *empirical risk minimization* (similar with Eq. 5.3).
>
> 2: Let $U' = [u'_1,...u'_K]$ be the $p \times K$ matrix formed from the K weight vectors.
>
> 3: Perform Singular Value Decomposition on U':U'=$V_1 D V_2^T$. The first h column vectors of V_1 are stored as rows of $\hat{\Theta}$

5.4.3 Structural Learning for Entity Linking: Incorporate *M* to *u* Estimation

As previous entity linking systems do not consider the information of the queried name, they usually use all the instances in M without any difference to train the linker. However, in data set M, some instances related with some particular names may share more predictive information with the queried name than other instances. Thus, in this work, we group the instances in M based on the "name", and then learn the shared information from the "name" group instead of individual instance. As shown in Figure 5.1, the data set M for entity linking usually has a certain number of names (e.g.*"Hoffman"*,*"Chad Johnson"*, etc.), each with some labeled instances. Then, we treat each "name" and its associated instances in M as a prediction problem of *structural learning*. Besides, the

queried name (e.g. *"AZ"* in Figure 5.1) with auto-labeled instances A_q is our target prediction problem.

According to the applications of *structural learning* in other tasks, such as WSD (Ando, 2006), *structural learning* assumes that there exists a predictive structure shared by multiple related problems. In order to learn the predictive structure Θ shared by ***M*** and A_q, we need to (a) select relevant prediction problems (i.e. relevant names) from ***M***. That is, they should share a certain predictive structure with the target problem; (b) select useful features from the feature set shown in Table 2.1. The relevant prediction problems may only has shared structure with target problem over certain features. In our work, we use a set of experiments including feature split and data set ***M*** partitioning to perform these two selection processes. This empirical method for selection will be elaborated in Section 5.6.3.

Let us assume that we have selected relevant names from data set ***M***, which together with the queried name can be used as the K related prediction problems in *structural learning*. Applying *structural learning* to the K problems, we can obtain the shared structure $\hat{\Theta}$ by *alternating structure optimization*. Then, the weight vector ***u*** for the queried name in Eq. 5.2 can be approximately solved by the following procedure:

1: Learn $\hat{w}$ and $\hat{v}$ for the queried name by minimizing the empirical risk on data set A_q:

$$\arg\min_{w,v} \left(\frac{1}{n^{(q)}} \sum_{i=1}^{n^{(q)}} L\left((w + \hat{\Theta}^T v)\mathbf{X}_i^q, Y_i^q \right) + \lambda \|w\|^2 \right)$$

2: The estimated weight vector *u* for the queried name is:

$$\hat{u} = \hat{w} + \hat{\Theta}^T \hat{v}$$

The $\hat{\Theta}^T \hat{v}$ part is learned from the selected names in *M* and all the instances in A_q, and therefore it can model the shared predictive structure between *M* and A_q, and remove the noises in *M* as we expected. The $\hat{w}$ part is learned from the data set A_q, which can tackle the distribution gap between training and test data sets (see Figure 5.2) in the previous work only using *M*.

5.5 Predicting NIL Mentions

So far we have assumed that each mention has a correct KB entry; however, when we run over a large corpus, a significant number of entities will not appear in the KB. In this situation, the document d_q with mention m_q should be linked to *NIL*. Traditional approaches usually need an additional classification step to resolve this problem (Zheng et al., 2010; Lehmann et al., 2010). In contrast, our approach seamlessly takes into account the NIL prediction problem. As we define $Y \in \{+1,-1\}$ to denote whether the pair of the mention and KB entry can be linked together, the median 0 can be assigned to $\phi_q(q,NIL)$. Then Eq. 5.1 is extended to:

$$e = \underset{c_i \in C_q \cup NIL}{\arg\max}\ \phi_q(q, c_i) \tag{5.6}$$

5.6 Experiments and Discussions

5.6.1 Experimental Setup

In our study, we use *TAC-10*[2] KB and document collection to evaluate the lazy learning with query-specific information on entity linking, and adopt micro-averaged accuracy officially used in *TAC-10* evaluation for our experiments, i.e. the number of correct links (including *NIL*) divided by the total number of the mentions. The training set of *TAC-10* consists of 5,404 mentions. Among them, 3,404 mentions are used as the data set M in our approach and the remaining 2,000 mentions are used as development set in our experiments.

5.6.2 Statistics of Data Set A_q

To minimize the distribution gap between training data and queries discussed at the beginning of this chapter, we incorporate the distribution knowledge learned from A_q to the learning process. Thus, one of the key factors for the success of our lazy learning model is whether we can obtain A_q for the queries.

Therefore, firstly we investigate the amount of the labeled instances created for each query. When our model runs over the test data set which consists of 2,250 queries, we find that 359 queries are assigned empty candidate sets (i.e. $C_q = \emptyset$) by the process described in Section 5.2. For these queries, we can directly link them with *NIL* without disambiguation. Thus, we only need to create A_q for the remaining 1,891 queries.

Figure 5.4 compares the proportions of the queries in different A_q size ranges. It shows that we have successfully created non-empty A_q for 96%

[2]http://nlp.cs.qc.cuny.edu/kbp/2010/

of the 1,891 queries. This proves that our approach learning the distribution knowledge for the queried name from the automatically labeled instances A_q is feasible in practice. This also supports our assumption about the existence of the document with unambiguous synonyms in the document collection.

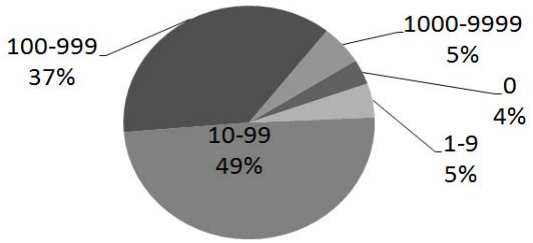


Figure 5.4: Proportions of the Queries Based on the Sizes of their Corresponding A_q

We also note that 49% of the queries have 10 to 99 labeled instances in A_q and 37% have 100 to 999 instances for each linker . In contrast, previous approaches usually trained their model on thousands of labeled instances. Thus, it suggests that we need more labeled instances for some queries and it is necessary to still leverage the manually labeled data set M in our learning process.

5.6.3 Exploring Θ Configuration

Because our lazy learning model generalizes on both the distribution knowledge learned from A_q and the predictive structure Θ shared by A_q and M, the effectiveness of such shared structure Θ is another key factor for the success of our lazy learning model. Thus, inspired by the work (Ando, 2006) for WSD, we design a set of experiments to investigate the configuration of Θ.

Consider the disambiguation of the two mentions *"CPC"* and *"NY"* in two

documents again. They have similar surface features (e.g. feature *"acronym matching"* is true). The surface features effective for linking to *"Communist Party of China"* may be also effective for disambiguating *"NY"* to "the city of New York", and vice versa. However, with respect to the semantic features, these two disambiguation problems may not have much in common. This is because *"Communist Party of China"* is likely related with the topic "politics", but *"the city of New York"* does not have such particular topic. That is, shared structure Θ between different names may depend on feature types, and in that case, seeking Θ for each of feature groups (*CF, SeF, SuF* and *GS* in Table 2.1) separately may be more effective. Hence, we experimented with both Θ configuration in Eq. 5.5 and Θ configuration, learning a Θ_j for each feature group *j* separately in Eq. 5.7.

$$\sum_{l=1}^{K}\left(\frac{1}{n^{(l)}}\sum_{i=1}^{n^{(l)}}L\left(w_l^T\mathbf{X}_i^{(l)}+\sum_{j\in F}v_l^{(j)\,T}\Theta_j\mathbf{X}_i^{(l,j)},Y_i^{(l)}\right)+\lambda\|w_l\|^2\right) \tag{5.7}$$

where F is a set of disjoint feature groups, and $\boldsymbol{X}^{(j)}$ (or $v^{(j)}$) is a portion of the feature vector $\boldsymbol{X}$ (or weight vector v) corresponding to feature group j, respectively.

The NE types of the instances in A_q and M are *PER, GPE* and *ORG*. Intuitively, the predictive structures of the names with the same NE type may be more similar than those of cross-NE-type names. Therefore, except for the feature split discussed above, we explore another two Θ configurations. One learns Θ from A_q and the whole M for each query. The other learns Θ from A_q and the subset of M, where the instances have the same NE type with the query.

Thus, we experiment on our development data set with the combinations of the two types of Θ configuration, i.e. configuration of feature split *F* and configuration for partitioning of data set *M*.

Figure 5.5 compares the performance using the various Θ configurations, and the results are in line with our expectation. *F={CF+SeF+SuF+GS}* treats the features of these four types as one group. It is equivalent to the Θ configuration without feature split in Eq. 5.5. Comparison of *F={CF, SeF, SuF, GS}* (learning Θ_j for these four feature groups separately by Eq. 5.7) and *F={CF+SeF+SuF+GS}* indicates that use of the feature split indeed improves disambiguation performance. We are also interested in whether all the feature groups are suitable for learning Θ_j. Thus, we further experimented with *F={SeF, SuF, GS}*, *F={CF, SuF, GS}*, *F={CF, SeF, GS}* and *F={CF, SeF, SuF}*. Figure 5.5 shows that these different subsets of feature groups do not improve the performance over using all the feature groups, and it proves that all the feature groups contribute to the learning of Θ. Besides, this figure also shows that learning Θ from A_q and the subset of *M* (i.e. instances have the same NE type with the query) usually performs better than learning it from A_q and the whole *M*. At last, as Θ has one parameter - its dimensionality h, the performance shown in this figure is the ceiling performance on the development set obtained at the best dimensionality (in {10, 50, 100,...}).

5.6.4 Evaluation Results for Lazy Learning

The experiments in this section evaluate our lazy learning model on the test data set of *TAC-10*. Our experiments used the best dimensionality $h = 150$ of Θ tuned on the development set in Section 5.6.3.

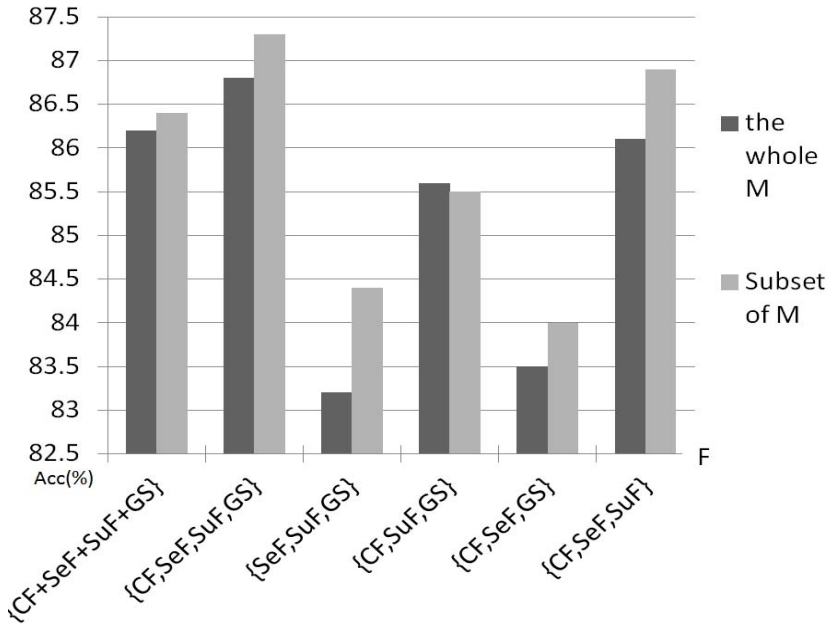


Figure 5.5: Accuracy on Development Set (As Θ has one parameter - its dimensionality h, the performance here is the ceiling performance obtained on the development set at the best dimensionality in {10, 50, 100,...})

Table 5.2 shows the performances of three baseline methods and our approach with overall accuracy as well as accuracy on five subsets of the test set.

	ALL	inKB	NIL	PER	ORG	GPE
M(Eq.3)	83.7	81.1	85.9	92.0	82.1	76.9
M(SVM)	84.0	78.5	88.6	92.1	84.0	76.0
M+A_q	84.5	81.4	87.1	92.7	82.7	78.1
A_q+Θ	86.6	84.5	88.3	94.8	85.2	79.7
A_q+Θ_j	**87.8**	85.5	90.0	96.1	86.3	80.9

Table 5.2: Micro-averaged Accuracy on Test Set

The second row (*M (Eq.3)*) used *empirical risk minimization* to estimate the weight vector u on the data set M (similar with Eq. 5.3). The third row (*M(SVM)*) used *SVM* classifier (Herbrich et al., 2000) to estimate the model on

M. These two methods are similar with most of the previous work for disambiguation, because all of them disambiguate a mention of a name based on the distribution knowledge learned from other labeled names. Row 5 (or 6) A_q+Θ *(or* Θ_j*)* shows the accuracy of our lazy learning model, which generalized the linker on both the distribution knowledge learned from A_q and the predictive structure Θ shared by A_q and *M*. Row 5 does not use feature split or data set *M* partitioning for learning Θ, but Row 6 uses them. Comparison of Row 6 and Row 2, 3 indicates our lazy learning model achieves significant improvements of 4.1% and 3.8%, respectively ($\rho < 0.05$, χ^2 statistical significance test). This significant improvement obtained by our approach is from solving the distribution gap (see Figure 5.2) of previous methods.

Besides, Row 4 (*M*+A_q) used *empirical risk minimization* to estimate *u* on the data set *M* and A_q directly. Comparing it with our lazy learning model, the idea to learn the shared predictive information Θ achieves significant ($\rho < 0.05$) gain. This is because, rather than directly using *M* with a lot of noise, we only incorporate the useful information in *M* shared with A_q to our learning process.

5.6.5 Comparison with State-of-the-Art Performance

We also compare our approach with the top systems in *TAC-10*. As shown in Figure 5.6, our lazy learning model achieves a 2% (or 5.9%) improvement over the best (or second best) system in *TAC-10*. The best system "lcc" used a state-of-the-art machine learning algorithm (i.e., logistic classifier) for disambiguation. However, same with other previous work, they only trained their model on data set *M* without considering the knowledge related to the queried

name. Comparing it with our approach, it proves that our lazy learning model has effectively tackled the distribution gap between training and test data set in the previous work and indeed improved the disambiguation systems.

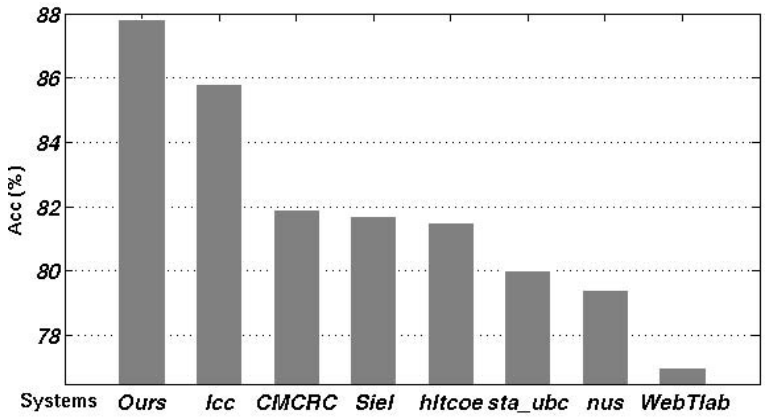


Figure 5.6: A Comparison with *TAC-10* Systems

We participated in *TAC-11*, and the submitted system (Zhang et al., 2011d) with the techniques: instance selection (see Chapter 3) and topic features (see Chapter 4) achieved the second best micro-averaged accuracy 86.3% among 21 teams. The best is 86.8% by Cucerzan (2011). We apply our method proposed in this chapter to our system at *TAC-11* (Zhang et al., 2011d), which achieves 87.6% with a 1.3% improvement.

5.7 Conclusions and Future Work

With the goal of achieving higher disambiguation performance, our focus in this chapter was to solve the distribution gap between training and test data sets in previous approaches. We have presented a lazy learning model, which can incorporate the distribution knowledge of the queried name to the learning process. To obtain this distribution knowledge, we proposed to automatically

label relevant instances A_q for the queried name. Besides, instead of using or combining labeled data set M directly to train the linker, we proposed to use the predictive structure Θ shared by M and A_q. Our experiment showed that the best configuration of Θ was to use feature split over all the feature groups and use data set M partitioning according to NE type. Finally, our experiments also proved that previous approaches for entity linking can be significantly improved.

In the future, to further improve the disambiguation performance, we would like to explore more methods to learn the knowledge from M and A_q.

Chapter 6: REAL-TIME ENTITY LINKING IN MICROBLOG

6.1 Introduction

Nowadays, microblog (e.g., Twitter) significantly influences the way we live. Millions of users post over 400 million status messages in Twitter daily and these textual messages on microblog contains various information for the entities in the world. Thus, in this chapter, we address the entity linking task under a more challenging scenario. Given a stream of messages from the microblog platform such as Twitter and an ambiguous name shared by some entities, we are required to link the name in the tweet to its corresponding KB entry in real time (the given name and its associated tweet are referred to as "query" in the remainder of this chapter).

Such an entity linking system can help other Twitter-based applications to find the right entity. For example, *Scandinavian Airlines System Group* is monitoring the feedback of passengers from Twitter. However, Twitter users usually use the ambiguous name "*SAS*" to represent *Scandinavian Airlines* in their status messages. Then it needs an entity linking system to filter out the noise: tweets with "*SAS*" but regarding to other entities sharing the same name "*SAS*" such as "*New Zealand Special Air Service*" and "*SAS Institute, Inc.*". Besides, entity linking system as the bridge between tweets and KB makes it

possible for them to borrow information from each other.

Compared with traditional entity linking task on long text, real-time entity linking in microblog is more challenging in at least the following three aspects.

First, microblog messages are much shorter consisting of only not more than 140 characters. The insufficient text in the messages makes it more difficult to properly characterize a common context between the short microblog message and KB entry. Besides, insufficient text also hinders applying the collective method (see Section 2.5) to entity linking in Twitter. The collective methods for entity linking in long text such as Cucerzan (2007), Ratinov et al. (2011) and Han et al. (2011), simultaneously disambiguated all the mentions in the same text by exploiting the interdependence among them. Note that a long text usually contains hundreds of mentions, for example, the data set used by Han et al. (2011) contains 161 name mentions per document on average. However, as a tweet usually contains only a small number of mentions [1], the collective methods leveraging on the interdependence among the mentions in the context would not perform well for entity linking in Twitter. Finally, the scarcity of text in the microblog messages also makes the tweets mentioning the same entity usually not share enough context and then it is difficult to cluster the tweets only based on their short text. Thus, the scarcity of text also pulls down the effectiveness of the query-level collaborative ranking method (Chen and Ji, 2011), as they searched collaborations for one query by clustering all the queries (i.e. tweets) containing the same mention.

To address the problem of insufficient text in Twitter, in this chapter we propose a bipartite graph based mapping function to enrich the context of the

[1] We run a named entity recognizer ($http://github.com/aritter/twitter\_nlp$) on the tweet collection used in our experiments. On average a tweet only contains 2.3 mentions.

microblog messages with the auxiliary long texts from web. The proposed method first selects some words as the bridge between tweets and auxiliary long text. The selected words should appear frequently in both tweets and auxiliary long texts, and also have the power to distinguish different entities. Then, based on the auxiliary long text, we construct a term co-occurrence matrix for the selected words and all other words. As the k largest eigenvectors of the term co-occurrence matrix are the continuous solution of the cluster membership indicators of the word set in the k-means clustering method, we propose to measure the word similarity in an space of the k principal components hidden in the the word set. Finally, a mapping function is constructed for tweet enrichment based on the learned word similarity.

Second, real-time entity linking in Twitter requires a quick response time, as it needs to monitor the tweets that keep coming at a fast pace. However, the important technique for traditional entity linking systems that gathers information from external sources on-the-fly can not meet this requirement. For example, as mentioned above, the collaborative ranking method (Chen and Ji, 2011) tried to obtain external information from query collaborations by clustering the text collection. Then their feature vector for the query also includes the cluster-level features, such as maximum, minimum, average tfidf/entity similarities between the KB entry and the texts in query collaboration cluster. Our lazy learning described in Chapter 5 automatically labeled some instances which contains the same mention with the query. As all these methods obtain external information based on the query, they only can generalize their model after the query is made, which causes the response time becomes slow. In this chapter, we use the tweets also containing the name being monitored as the external

knowledge for the query. Instead of generalizing the model on-the-fly, we propose a bipartite-graph-based model to cluster the external tweets and then we store the cluster information into a multinomial model off-line.

Third, traditional entity linking approaches used manually annotated data to learn a classifier or ranker (Dredze et al., 2010; Lehmann et al., 2010; Zheng et al., 2010; Zhang et al., 2010; Chen and Ji, 2011; Ploch, 2011; Ratinov et al., 2011). However, in the Twitter platform, different users post their tweets using diverse lexicons and styles which make it very difficult for the supervised learning approach trained on a small amount of training instance perform well on the new coming tweets. Besides, new words or phrases are introduced in Twitter daily, which quickly invalidates static linking models trained on a certain manually labeled set. Thus, manually labeling training data is very expensive for entity linking in Twitter. In Chapter 3, we proposed to automatically label training instances for entity linking by leveraging unambiguous names in the text. The basic idea is to take a news article with an unambiguous mention referring to an entity e_1 in KB and replace it with its variation which may refer to e_1, e_2 or others. However, unlike news article, tweets usually only contains the ambiguous name of the entities. For example, the new article for "*Apple Inc.*" usually uses the full name "*Apple Inc.*" in the first paragraph, and uses "*Apple*" for short name in the following paragraphs, but in tweets usually there is only short name "*apple*". This property of Twitter disables the method in Chapter 3 for automatically labeling training instances.

In this chapter, we present our unsupervised learning framework (USLF) based on three bipartite graphs for entity linking in Twitter to address the new challenges above. Given the name which we are required to monitor on Twitter,

our USLF first collects a set of tweets containing the given name and then enrich their context by our mapping function mentioned above. During the unsupervised learning stage, nodes (i.e. tweets, words and KB entries) of the bipartite graphs are partitioned into clusters. Cluster labels $C = \{1, 2, ...K\}$ are assigned to tweets and words, and also each cluster k are given probabilities $p(C = k|e_j)$ for each KB entry e_j. During the supervised learning stage, with the clusters determined we learn a multinomial model for each cluster where the cluster information is represented as a posterior probability of a tweet t_i given one cluster k, $P(t_i|C = k)$. Finally, during on-line prediction, for a new query with tweet t, the correct entity is selected for the query by a Bayes model with $P(t|C = k)$, $p(C = k|e_j)$ and the prior probability of the entities $P(e_j)$. $P(e_j)$ can be estimated by the popularity of the KB entities in the world.

The contribution of this chapter is the proposed USLF for real-time entity linking in Twitter. The advantages of our model can be summarized as follows,

(1) A tweet enrichment function is embedded in our model based on the word similarity, which is calculated in the k principal component space of the word set with the help of auxiliary long text.

(2) Our model not only uses the information in the query, but also benefits from the external information: the tweets which also contains the name being monitored. Meanwhile our model also can guarantee a fast response time. This is because we generalize the multinomial model $P(t_i|C = k)$, and calculates probabilities of each cluster $p(C = k|e_j)$ for KB entries based on these tweets off-line. This significantly differs from previous query-level collaborative ranking method for traditional entity linking. As their model needs to cluster texts together with the query and then extract cluster features for the

query, their approach benefits from the collaboration texts on-the-fly.

(3) Our USLF method leverages context information of query and KB entities, popularity knowledge of KB entities and clustering result on an additional tweet set for disambiguation. By modeling these sources as probabilistic distributions, our method has a statistical foundation, which differs from previous *ad-hoc* approaches (e.g., the way to represent the cluster-level information for the query in previous collaborative ranking method).

(4) The multinomial model $P(t_i|C = k)$ and probabilities of each cluster $p(C = k|e_j)$ for KB entries in USLF are learned from the unsupervised clustering result. Thus, USLF does not need human labor to annotate training data and then can easily update our model with the fresh data containing new words or phrases in the message stream.

6.2 Unsupervised Learning Framework

6.2.1 Bipartite Graphs for Entity Linking in Tweets

In this section, we model the problem of entity linking in microblog as three weighted bipartite graphs. Let us denote a graph by $G(V, E)$, where V is the vertex set and E is the edge set. The graph $G(V, E)$ is bipartite with two vertex classes X and Y if $V = X \cup Y$ with $X \cap Y = \emptyset$ and each edge in E has one endpoint in X and one endpoint in Y . A weighted bipartite graph is denoted as $G(X, Y, WT)$ with $WT = wt_{ij}$, where $wt_{ij} > 0$ denotes the weight of the edge between vertex i and j and $w_{ij} = 0$ denotes there is no edge between vertices i and j. As shown in Figure 6.1, the three weighted bipartite graphs in our model are named as tweet representation, context enrichment and entity

representation. In the graphs, $V_T = \{t_i\}$ represents the set of tweets with the same name but they may refer to different entities, $V_W = \{w_i\}$ represents a set of words in Tweets, $V_A = \{a_i\}$ represents a set of words in auxiliary long text and $V_E = \{e_i\}$ represents the set entities to which we link the mentions of tweets. Note that the tweets in V_T are not the queries, and we collect these tweets also containing the monitored name as external collaborators. Also note that the entities in V_E share a same name which we are monitoring in Twitter.

In the graph of tweet representation $G(V_T, V_W, WT')$, the weighted edge $WT' = \{wt'_{i,j}\}$ denotes the number of times word w_j appears in tweet t_i. The second bipartite graph $G(V_W, V_A, WT'')$ is used for context enrichment, where $WT'' = \{wt''_{i,j}\}$ denotes the number of auxiliary long text where both w_i and a_j appear. In the graph entity representation $G(V_A, V_E, WT''')$, $WT''' = \{wt'''_{i,j}\}$ denotes the number of times word a_i appears in the text of the entity.

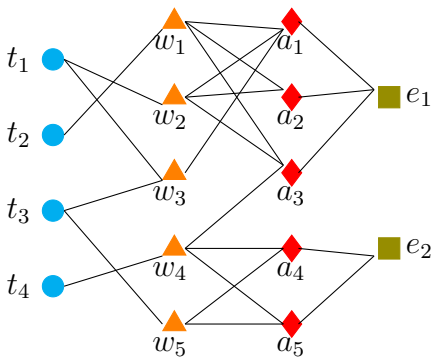


Figure 6.1: Bipartite graphs of tweets, words, auxiliary words and entities

Based on these three bipartite graphs, we first give an overview of our USLF framework in Algorithm 6.1.

Intuitively, there are two stages during off-line learning. In the unsuper-

Algorithm 6.1 Real-time Entity Linking in Twitter

– ***Off-line Learning***

- ***unsupervised learning***

1: Construct the tweet enrichment graph $G(V_W, V_A, WT'')$

2: Learn the word similarity matrix $S_{i,j}$ as Eq. 6.2 based on $G(V_W, V_A, WT'')$

3: Given the tweet representation graph $G(V_T, V_W, WT')$, define the mapping function for tweet enrichment as $WT'S$

4: Simplify the graphs in Figure 6.1 to two bipartite graphs $G(V_T, V_A, WT'S)$ and $G(V_A, V_E, WT''')$ by tweet enrichment $WT'S$

5: Partition V_A and V_T into K clusters $C = \{1, 2, ..., K\}$ based on $G(V_T, V_A, WT'S)$ and $G(V_A, V_E, WT''')$

6: Compute the prior probability $P(e_j)$ for entity e_j by the popularity of the entity in the world.

- ***supervised learning from clustering result***

7: From the clustering result, learn the probabilities of each cluster $p(C = k|e_j)$ for KB entries as Eq. 6.4

8: Learn a multinomial model $multinomial(p_k)$ as Eq. 6.5 for each cluster k based on the clustering result

– ***On-line Inference for a New Tweet*** t_i

9: Enrich tweet by t_iS

10: Calculate the posterior probabilities $P(t_i|C = k)$

11: Return the correct entity by Eq. 6.9

vised learning stage, we compute the prior probability $P(e_j)$ for each entity, learn the tweet enrichment function and also cluster the tweets, words and entities. Then, supervised by the clustering result, the probabilities of each cluster $p(C = k|e_j)$ for KB entries and the multinomial model $P(t_i|C = k)$ are learned in the supervised learning stage. In the on-line linking, we compute the probability distribution of a new tweet over predefined clusters acquired in the learning stage by Bayes rule. Entity is selected as answer according to three criteria: $P(e_j)$, $P(t_i|C = k)$ and $p(C = k|e_j)$.

6.2.2 The Bipartite Graph of Context Enrichment

Seeds Selection. Tweets (50.6% of the Tweets in our experiments) contain URLs, we therefore crawl the content of the referenced URLs to form the set of auxiliary long texts. This can also be achieved by sending the monitored name as queries to a search engine to retrieve a set of most relevant results. With the auxiliary long texts, we then need to identify some words used as the bridge between the tweets and auxiliary long texts. The seeds of word should occur frequently in both tweets and long text. Furthermore, the seeds should also have the property which can distinguish different entities. Based on the two criteria, we present two strategies for selecting the seeds.

The first strategy is to select seeds based on their frequency in both tweets and long texts and the document frequency among texts of entities. Specifically, given the number l of words to be selected, we chose words with a $tf.idf$ value larger than β. β is set to be the largest number such that we can get at least l such words.

The second strategy uses mutual information (MI) to measure the depen-

dence between words and text collections (i.e. tweets, long texts and the texts of entities). Let W, T, L and $\{e_j\}$ denote words, tweets, long texts and the texts of entities, respectively. Then, we would like to select the words w_i with low mutual information $I(w_i, \{T, L\})$ and high mutual information $I(w_i, \{e_j\})$. Thus, the seeds selection criterion using mutual information is as Eq. 6.1

$$\begin{aligned} & I(w_i, \{e_j\}) - I(w_i, \{T, L\}) \\ & = \sum_{e_j} P(w_i, e_j) log(\frac{P(w_i, e_j)}{P(w_i)p(e_j)}) \\ & - \sum_{y=TorL} P(w_i, y) log(\frac{P(w_i, y)}{P(w_i)p(y)}) \end{aligned} \tag{6.1}$$

Tweets Enrichment. The tweets enrichment approach proposed in this section is based on the bipartite graph $G(V_W, V_A, WT'')$ in Figure 6.1. As we have selected some proper seeds as the bridge between tweets and auxiliary long text, $WT'' = \{wt''_{i,j}\}$ can be defined as follows, if w_i is in the seed set, $wt''_{i,j}$ is the number of auxiliary long text where both w_i and a_j appear, otherwise $wt''_{i,j}$ =0. Our goal is to learn a mapping function so that we can enrich the short text of tweets using the auxiliary words in V_A.

In the spectral graph theory (Ding and He, 2004), it has been proved that the k largest eigenvectors of a term-document co-occurrence matrix are the continuous solution of the cluster membership indicators of the data in the k-means clustering method. This implies that our mapping function constructed from the k largest eigenvectors can cluster the words and then we can calculate the word similarity in the new space. Assume that the weight matrix WT'' in graph $G(V_W, V_A, WT'')$ is in a $\mathbb{R}^{l\times(m-l)}$ space. Then we can form an affinity matrix

A for the bipartite graph $G(V_W, V_A, WT'')$: $A = \begin{bmatrix} 0 & WT'' \\ WT''^T & 0 \end{bmatrix} \in \mathbb{R}^{m \times m}$.
The k principal components can be obtained by the following procedures,

1. Form a diagonal matrix D, where $D_{ii} = \sum_j A_{ij}$, and construct the matrix $D^{-1/2}AD^{-1/2}$.

2. Find the k largest eigenvectors of $D^{-1/2}AD^{-1/2}$, referred to as $u_1, u_2, ..., u_k$, and form the matrix $U = [u_1 u_2 ... u_k] \in \mathbb{R}^{m \times k}$.

Then we form a matrix S as Eq 6.2, which is the word similarity between the word set V_W and the auxiliary word set V_A based on the k principal components of the word set. If the word w_i is in the *seed set*, the similarity will be calculated in the space of the k principal components.

$$
\begin{aligned}
S_{ij} &= sim(w_i, a_j) \\
&= \begin{cases} WT''_{[i,:]} U_{[1:m-l,:]} (U^T_{[1:l,:]} WT''_{[:,j]}) & w_i \in seed\ \ set \\ 1 \quad w_i = a_j \wedge w_i \notin seed\ \ set \\ 0 \quad w_i \neq a_j \wedge w_i \notin seed\ \ set \end{cases}
\end{aligned} \tag{6.2}
$$

Then, a tweet can be enriched by TS and we can transfer bipartite graphs shown in Figure 6.1 to the bipartite graphs in Figure 6.2 by applying TS to the tweets in V_T. The new graphs are named as tweets enrichment representation $G(V_T, V_A, WT)$, and entity representation $G(V_A, V_E, WT''')$. The new weight matrix WT in $G(V_T, V_A, WT)$ can be obtained by $WT'S$, where WT' refers to the weight matrix in $G(V_T, V_W, WT')$ in Figure 6.1 and S refers to the similarity matrix in Eq. 6.2.

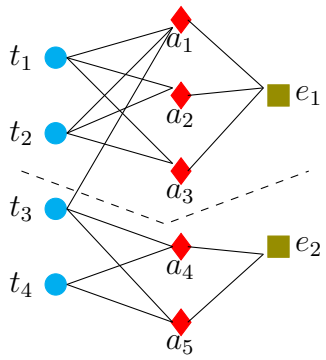


Figure 6.2: Bipartite graphs of tweets, auxiliary words and entities

6.2.3 Off-Line Learning

In this section, we present our learning process for entity linking in Twitter based on the bipartite graphs: tweets enrichment representation $G(V_T, V_A, WT)$ and entity representation $G(V_A, V_E, WT''')$ in Figure 6.2. These weighted graphs can be written as,

$$M = \begin{bmatrix} 0 & WT & 0 \\ WT^T & 0 & WT''' \\ 0 & WT'''^T & 0 \end{bmatrix}.$$

We apply our proposed framework USLF to the matrix M, during the offline learning stage, all the nodes in Figure 6.2 are first partitioned into clusters by applying an unsupervised clustering method to M such as spectral clustering (Ding and He, 2004), cluster labels $C \in \{1, 2, .., K\}$ are assigned to tweets and words as their cluster indicator, and the probabilities of each cluster $p(C = k|e_j)$ given KB entry e_j are calculated. Then, a multinomial model is built to store the relation between tweet clusters and words.

Estimate the Probabilities $p(C = k|e_j)$. With bipartite graph $G(V_A, V_E, WT''')$ constructed and the clusters $C = \{1, 2, ...K\}$ for words V_A determined, we estimate the probability $p(C = k|e_j)$ of cluster k given each KB entry as follows.

First, we define the closeness between a word cluster $C = k$ and an entity e_i based on two criteria: (1) the clusters are more close to the entity e_j if they have a higher connectivity with e_j than other clusters; (2) the clusters are more close to the entity e_j if they also have a lower connectivity with other entities. Thus, our closeness score for entity e_j and cluster $C = k$ is defined as Eq. 6.3.

$$Closeness(e_j, C = k) = \frac{\sum_i wt'''^2_{i,j}}{\sqrt{\sum_i \sum_{j'=1}^{n} wt'''^2_{i,j'}}} \tag{6.3}$$

where, i satisfies that $a_i \in V_A$ is clustered to the cluster $C = k$. n is the number of entities in V_E.

Then,

$$p(C = k|e_j) = \frac{Closeness(e_j, C = k)}{\sum_{k'=1}^{K} Closeness(e_j, C = k')} \tag{6.4}$$

Multinomial Model Learning. With bipartite graph $G(V_T, V_A, WT)$ constructed and the clusters $C = \{1, 2, ...K\}$ for words V_A determined, we learn a multinomial model to represent the relation between the word clusters and tweets as follows.

An enriched tweet vector can be represented as the vector $t_i S = WT_{i,:} = [a_1, a_2, ...a_q]$, where q is the number of words in V_A. The distribution of the tweet within each cluster k can be estimated by learning a multinomial model. Let the cluster labels be $C = \{1, 2, .., K\}$, then

$$P(t_i|C=k) = Mulnomial(p_{1,k}, p_{2,k}, ...p_{q,k}) \tag{6.5}$$

where $p_{m,k} \in \{p_{1,k}, p_{2,k}, ...p_{q,k}\}$ is probability of cluster $C = k$ generating word m (word $m \in V_A$).

Parameter Estimation. There are $q \times K$ parameters in the mixture multinomial model. With the clusters determined, we apply EM algorithm to estimate the multinomial parameters $p_m \in \{p_1^{(t)}, p_2^{(t)}, ...p_q^{(t)}\}$.

The E-step estimates the posterior probability $P(t_i|C = k)$:

$$P(t_i|C=k) = Mulnomial(p_1^{(t)}, p_2^{(t)}, ...p_q^{(t)}) \tag{6.6}$$

The M-step uses $P(t_i|C = k)$ to update the parameters of multinomial distribution based on ***Laplace smoothing*** with a smoothing parameter α,

$$\hat{p}_m^{t+1} = \frac{sft.cnt(w_m) + \alpha}{\sum_{j=1}^{q} sft.cnt(w_j) + \alpha q} \ \ (0 < \alpha \leq 1) \tag{6.7}$$

where,

$$sft.cnt(w_j) = \sum_{t_i \in C=k} p(t_i) count(w_{i,j});$$

$$count(w_{i,j}) = WT_{i,j};$$

$$p(t_i) = \sum_{k=1}^{K} P(t_i|C=k);$$

Estimate the Probability of KB Entries $p(e_j)$. We estimate $p(e_j)$ based on

Wikipedia Hyper-links. Let $IL(e_j)$ denote the number of incoming links in Wikipedia referring to the Wikipedia page of e_j. Then,

$$p(e_j) = \frac{IL(e_j)}{\sum_{j'=1}^{n} IL(e_{j'})} \tag{6.8}$$

where n is the number of entities in V_E.

6.2.4 On-Line Inference for New Tweets

Given a new tweet t_i containing the monitored mention, the corresponding entity is determined by $L(t_i) = \arg\max_{e_j} P(e_j|t_i)$. Apply the Bayes rule to it as follows,

$$\begin{aligned} P(e_j|t_i) &= \frac{P(e_j)P(t_i|e_j)}{P(t_i)} \\ &\propto P(e_j)\sum_{k=1}^{K}[P(C=k|e_j)P(t_i|C=k)] \end{aligned} \tag{6.9}$$

where the three factors can be calculated by Eq. 6.8, Eq. 6.4 and Eq. 6.5, respectively.

6.3 Experiments and Discussions

6.3.1 Experiment Setup

In our study, we employed the tweet collection introduced in the task of online reputation management at Web People Search (WePS-3) (Amigo et al., 2010). To compare with the state-of-the-art systems, the standard training and test set

of this task are used as our development set and test set, which are referred to as WePS3-D and WePS3-T respectively. WePS3-D and WePS3-T contain 52 and 48 home pages of companies respectively. The tweets collection in WePS3-D and WePS3-T is retrieved from Twitter API by 100 search keywords. The keywords such as "apple", "oracle" and "sony" are the short names of the 100 companies. The number of retrieved tweets per search keyword is variable: between 385 and 500 tweets. Based on each company homepage, the tweets retrieved by that company's short name are labeled as "related" or "non related". For example, based on the home page "*http://www.apple.com*", two tweets retrieved by keyword 'apple' are labeled as follows.

- you can install 3rd-party apps that haven't been approved by Apple. -**related**
- okay maybe i shouldn't have made that apple crumble. -**non related**

In our algorithm, we use three additional data sets to construct the bipartite graphs, a tweet collection $V_T = \{t_i\}$, auxiliary long text and an entity list $V_E = \{e_i\}$. The tweet collection V_T can be retrieved from Twitter API using company's short name (e.g. 'apple') as search keyword. In our experiments, when we are predicting an ambiguous name (e.g. 'apple') in a tweet, the remaining tweets in WePS3-D/WePS3-T also containing 'apple' serve as the tweet collection V_T. The auxiliary long text are formed from the two sources: (1) there are 50.6% tweets in V_T containing URLs, we therefore crawl the content of the referenced URLs to obtain auxiliary long texts (2) As mentioned above, the short names of companies are used as the search keywords to collect the tweets in WePS3-D/WePS3-T. Then, we also submit the short name to

Google and crawl the top 200 Google results as the auxiliary long texts. At last, we use the articles of Wikipedia to form the entity list V_E. We also employ the short names to retrieve Wikipedia articles by Wikipedia disambiguation page. The disambiguation page titled as "short name (disambiguation)" contains a list of entities sharing such short name, For example, "Apple (disambiguation)" contains a list of entities V_E sharing the same short name "Apple". Note that the homepage of the company in WePS3-D/WePS3-T can be mapped to one Wikipedia article in V_E by URL matching, as the URL of the company always appears in its corresponding Wikipedia article. Table 6.1 shows the sizes of the three data sets for bipartite graph construction.

	V_T	auxiliary long text	V_E
# of texts	461	382	28

Table 6.1: Sizes of text collections (average value over the 100 short names)

During on-line prediction, our model links the short name in a new Tweet to its corresponding entity in V_E. Note that for each short name, only the entity mapped to company's homepage is annotated as "related" or "non related". We then convert our system output to "related" if the new tweet is linked to that annotated entity, or "non related" if it is linked to other entities. Based on the ground truth, system output can be grouped into four categories: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Following online reputation management task (Amigo et al., 2010), we evaluate the performance of our entity linking system in Twitter by three metrics as Eq. 6.10: accuracy (Acc), F-measure of the "related" class (F^+) and F-measure of the "non related" class (F^-). As supplemental metrics, the two kinds of F-

measure consider the distribution of "related" and "non related" tweets within the correct outputs. That is, for a high ambiguous company name, even only a few related tweets appeared in the corpus, the decisions taken in these cases are crucial.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precison^{+} = \frac{TP}{TP + FP}; Recall^{+} = \frac{TP}{TP + FN}$$

$$F^{+} = \frac{2 * Precison^{+} * Recall^{+}}{Precison^{+} + Recall^{+}} \tag{6.10}$$

$$Precison^{-} = \frac{TN}{TN + FN}; Recall^{-} = \frac{TN}{TN + FP}$$

$$F^{-} = \frac{2 * Precison^{-} * Recall^{-}}{Precison^{-} + Recall^{-}}$$

6.3.2 Experiment Results

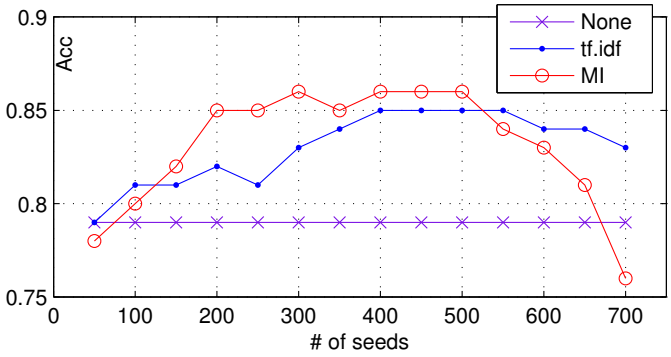
To address the problem of context scarcity in a single tweet, a context enrichment function is proposed in our model. This enrichment function is defined based on a matrix of word similarity, which is calculated over the k principal components of the word set with the help of auxiliary long text. As shown in Section 6.2.2, we propose two methods based on tf.idf and MI to select top l

words as bridge between tweets and auxiliary long text. Thus, we designed a set of experiments on development set WePS-D to evaluate the effectiveness of the enrichment function under tf.idf-based or MI-based seed selection with different l values (i.e. 50, 100, 150 ...). USLF without enrichment function is used as the baseline system.

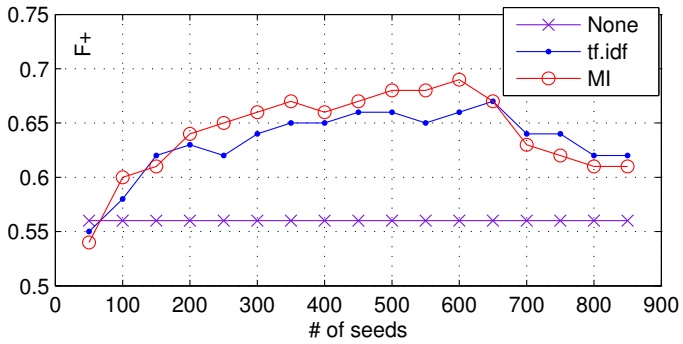
Figure 6.3 provides the results (Acc, F^+ and F^-) of these experiments, where "None" is the baseline system, and X-axis represents the values of l. We can find that both "tf.idf" and "MI" significantly outperforms the baseline system under all the three metrics (Acc, F^+ and F^-) within a large range of l value from 200 to 600. The improvements prove that the proposed context enrichment function can effectively address the problem of context scarcity in tweets by leveraging auxiliary long texts.

By checking the performance of "MI" and "tf.idf" at l values which are greater than 600, we can find that the performances over all the three metrics decrease as the increase of l value. This downward trend indicates that not all the words can serve as the bridge between tweets and long text. It also proves the effectiveness of our proposed seed selection methods that aim to find the words with a high frequecy in both tweets and long text, and the property of disignuishing entities.

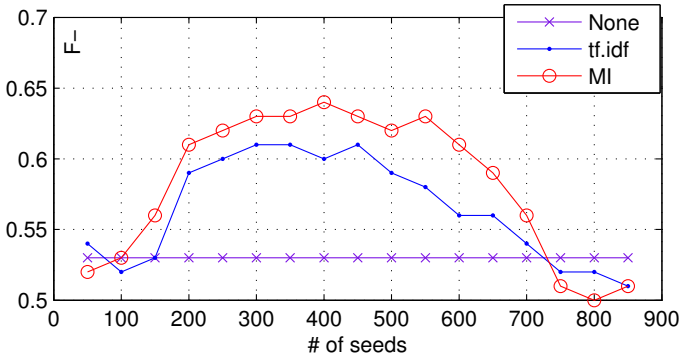
By comparing "MI" with "tf.idf" in Figure 6.3, I also see that "MI" is able to find a more proper set of seeds than "tf.idf" at most of the values of l. For example, by checking Acc at l=200, 250 or 300, the difference between "MI" and "tf.idf" is statistical significant ($\rho < 0.05$, t-test). This tells us that mutual information is a better choice for seed selection, and thus we use MI-based seed selection for the following experiments on data set WePS-T.



(a)



(b)



(c)

Figure 6.3: (a)Acc, (b)F^{+} and (c)F^{-} on data set WePS-D

	Acc	$Precison^+$	$Recall^+$	F^+	$Precison^-$	$Recall^-$	F^-
USLF	**0.87**	0.77	0.73	**0.69**	0.89	0.53	**0.59**
WePS rank-1	0.83	0.71	0.74	0.63	0.84	0.52	0.56
WePS rank-2	0.75	0.75	0.54	0.49	0.74	0.60	0.57
WePS rank-3	0.73	0.74	0.62	0.51	0.74	0.49	0.47

Table 6.2: Acc, F^+ and F^- on data set WePS-T

In Table 6.2, we compare our USLF with the state-of-the-art entity linking systems in Twitter on WePS-T. The three baselines are the top 3 systems in WePS-3 task. WePS rank-1 used an SVM classifier, which employed a set of features including keywords and "profile". Both WePS rank-2 and rank-3 use a naive Bayes classifier with a set of feature such as "is the query an acronym" and "does Wikipedia have disambiguation page for the query", but difference between the two systems is only at some parameter setting. By checking Acc, F^+ and F^- in Table 6.2, we can find that our approach significantly outperforms all the three systems over all the metrics. This proves that our bipartite-graph-based model is more suitable for entity linking in Twitter, as it seamlessly combine the context information of query and entities, popularity knowledge of entities and clustering result on an additional tweet set. Especially, popularity knowledge of entities, information of clustering result on an additional tweet set and context enrichment are not covered in the baseline systems. Besides, all the baseline systems use a set of manually labeled data set to train their model. In contrast, our model is guided by the unsupervised clustering result, which does not need human labor to annotate training data and also can easily update our model to adapt to the new changes in Twitter.

6.4 Conclusions and Future Works

In this chapter, we present an unsupervised learning framework (USLF) based on three bipartite graphs for entity linking in Twitter. First, a tweet enrichment function is embedded in our model based on the word similarity, which is calculated in the k principal component space of the word set with the help of auxiliary long text. Second, our model not only uses the information in the query, but also benefits from the external information: the tweets which also contains name being monitored. Meanwhile our model also can guarantee a fast response time. This is because we generalize the multinomial model $P(t_i|C = k)$, and calculates probabilities of each cluster $p(C = k|e_j)$ give a KB entry based on these tweets off-line. Third, our USLF method leverages context information of query and entities, popularity knowledge of entities and clustering result on an additional tweet set for disambiguation. By modeling these sources as probabilistic distributions, our method has a statistical foundation. Finally, the multinomial model $P(t_i|C = k)$ and probabilities of each cluster $p(C = k|e_j)$ for KB entries in USLF are learned from the unsupervised clustering result. Thus, USLF does not need human labor to annotate training data. This chapter shows that our USLF has a more quick response time than previous work theoretically. In the future, we would like to design a set of experiments to justify it. Besides, in this chapter, we conduct experiments to compare our approach with the state-of-the-art entity linking in Twitter. In the future, we will also design experiments to compare our approach with the approaches in traditional entity linking. We also would like to incorporate user information of Tweets to our model in the future as Cassidy et al. (Cassidy et al., 2012) did.

Chapter 7: CONCLUSIONS

In this thesis, we have systematically conducted a literature survey on entity linking. The survey starts on the definitions, benchmarks and related problems for entity linking. Then, we summarized the existing work on entity linking and presented their pros and cons.

Most of state-of-the-art entity linking systems use annotated data to learn a classifier or ranker by supervised learning algorithms. Chapter 3 proposed to automatically label a large scale training corpus for supervised learning algorithms, where we label the ambiguous mentions leveraging on their unambiguous synonyms. We also proposed an instance selection strategy to select an informative, representative and diverse subset from the auto-generated dataset. During the iterative selection process, the batch sizes at each iteration change according to the variance of classifier's confidence or accuracy between batches in sequence.

Chapter 4 introduced topic models to entity linking, which can discover the underlying topics in the context of mention and KB entries. we proposed a Wikipedia-LDA to model the topics of texts, where we investigated the effectiveness of five subsets from Wikipedia categories to represent the underlying topics.

Chapter 5 presented a lazy learning model, which can incorporate the query-specific information to the learning process. To obtain such information, we

proposed to automatically label relevant instances for the queried name. Besides, instead of using or combining labeled data set related with other names directly to train the linker, we proposed to use the predictive structure shared by the two data sets which are related with queried name and other names respectively.

Finally, this thesis addressed entity linking task under a more challenging scenario, where we linked the mentions in microblog to a KB in real time. We proposed an unsupervised learning framework (USLF) which is based on three bipartite graphs to address the new challenges in microblog. First, in our USLF, a tweet enrichment function is embedded based on the word similarity, which is calculated in the k principal component space of the word set with the help of auxiliary long text. Second, our model not only uses the information in the query, but also benefits from the external information: the tweets which also contains the name being monitored. Meanwhile our model also can guarantee a fast response time. Third, our USLF method uses a Bayes method to model the context information of query and entities, popularity knowledge of entities and clustering result on an additional tweet set for disambiguation.

Bibliography

E. Amigo, J. Artiles, J. Gonzalo, D. Spina, B. Liu, and A. Corujo 2010. *WePS3 Evaluation Campaign: Overview of the On-line Reputation Management Task.* CLEF (Notebook Papers/LABs/Workshops) 2010.

R. K. Ando. 2006. *Applying Alternating Structure Optimization to Word Sense Disambiguation.* In Proc. of the Conference on Natural Language Learning (CoNLL). 2006

R. K. Ando and T. Zhang. 2005a. *A high-performance semi-supervised learning method for text chunking.* In Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL). 2005.

R. K. Ando and T. Zhang. 2005b. *A framework for learning predictive structures from multiple tasks and unlabeled data.* Journal of Machine Learning Research, 6(Nov):1817–1853.

J. Artiles, J. Gonzalo and S. Sekine. 2007. *The semeval-2007 web evaluation: Establishing a benchmark for the web people search task.* In Proceeding of the Fourth International Workshop on Semantic Evaluations (SemEval-2007).

A. Bagga and B. Baldwin. 1998. *Entity-based cross-document coreferencing using the vector space model.* In Proceedings of joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL), 79-85. 1998

D. Blei, A. Y. Ng, and M. I. Jordan. 2003. *Latent Dirichlet Allocation.* Journal of Machine Learning Research 3:993-1022, 2003.

H. Brighton and C. Mellish. 2002. *Advances in Instance Selection for Instance-Based Learning Algorithms.* Data Mining and Knowledge Discovery. 6,153-172. 2002

Brinker. 2003. *Incorporating Diversity in Active Learning with Support Vector Machines.* In Proceeding of International Conference on Mechine Learning. ICML 2003

R. Bunescu and M. Pasca. 2006. *Using Encyclopedic Knowledge for Named Entity Disambiguation.* In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, 2006

P. Bysani, K. Reddy, V. Reddy, S. Kovelamudi, P. Pingali and V. Varma. 2010. *IIIT Hyderabad in Guided Summarization and Knowledge Base Population.* In Proceedings of Text Analysis Conference. 2010

S. A. Caraballo. 1999. *Automatic construction of a hypernym-labeled noun hierarchy from text.* In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, College Park, Md., 20-26 June. 1999.

T. Cassidy, Z. Chen, J. Artiles, H. Ji, H. Deng, L. Ratinov, J. Zheng, J. Han and D. Roth. 2011. *CUNY-UIUC-SRI TAC-KBP2011 Entity Linking System Description.* Proceedings of Text Analysis Conference (TAC2011).

Taylor Cassidy, Heng Ji, Hongzhao Huang, Lev-Arie Ratinov, Arkaitz Zubiaga. 2012. *Expanding Microblog Context to Enhance Disambiguation to Wikipedia.* Proc. 24th International Conference on Computational Linguistics (COLING2012).

A. X. Chang, V. I. Spitkovsky, E. Agirre, C. D. Manning 2011. *Stanford-UBC Entity Linking at TAC-KBP, Again.* In Proceedings of Text Analysis Conference. 2011

S. Cucerzan 2007. *Large-Scale Named Entity Disambiguation Based on Wikipedia Data.* In Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2007

S. Cucerzan 2011. *TAC Entity Linking by Performing Full-document Entity Extraction and Disambiguation.* In Proceedings of Text Analysis Conference. 2011

Z. Chen and H. Ji 2011. *Collaborative Ranking: A Case Study on Entity Linking* In Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2011

H. Dai, R. Tsai and W. Hsu1. 2011. *Entity Disambiguation Using a Markov-Logic Network.* In Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 846-855, Chiang Mai, Thailand, November 8-13, 2011. c 2011 AFNLP.

C. Ding and X. He 2004. *K-means clustering via principal component analysis* International Conference on Machine Learning, 2004.

M. Dredze, P. McNamee, D. Rao, A. Gerber and T. Finin 2010. *Entity Disambiguation for Knowledge Base Population.* 23rd International Conference on Computational Linguistics, 2010, China.

P. Edmonds and S. Cotton. 2001. *Senseval-2 overview* In Proceedings of SENSEVAL-2, 1-6.

N. Fernandez, J. A. Fisteus, L. Sanchez, E. Martin 2010. *WebTLab: A cooccurrence-based approach to KBP 2010 Entity-Linking task* In Proceedings of Text Analysis Conference. 2010.

T. Finin, Z. Syed, J. Mayfield, P. McNamee and C. Piatko. 2009. *Using Wikitology for Cross-Document Entity Coreference Resolution.* In Proceedings of AAAI Conference on Artificial Intelligence, AAAI 2009.

S. Gottipati and J. Jiang 2011. *Linking entities to a knowledge base with query expansion.* In Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2011

X. Han and J. Zhao 2009. *Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge.* In Proceedings of the 18th ACM conference on Information and knowledge management (2009).

X. Han and L. Sun 2011a. *A Generative Entity-Mention Model for Linking Entities with Knowledge Base.* In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL). 2011.

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

D. Harman and M. Liberman. 1993. *TIPSTER Complete. LDC93T3A, Philadelphia, Penn.* Linguistic Data Consortium , 1993.

M. A. Hearst. 1992. *Automatic acquisition of hyponyms from large text corpora.* In Proceedings of the 15th International Conference on Computational Linguistics, Nantes, France, 23-28 August 1992.

R. Herbrich, T. Graepel and K. Obermayer. 2000. *Large Margin Rank Boundaries for Ordinal Regression.* Advances in Large Margin Classifiers (pp. 115-132). 2000.

H. Ji, R. Grishman, H. T. Dang, K. Griffitt and J. Ellis 2010. *Overview of the TAC 2010 Knowledge Base Population Track.* In Proceedings of Text Analysis Conference 2010 (TAC 10).2010

H. Ji and R. Grishman 2011a. *Knowledge Base Population: Successful Approaches and Challenges.* In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL). 2011.

H. Ji, R. Grishman and H. T. Dang 2011b. *An Overview of the TAC2011 Knowledge Base Population Track.* In Proceedings of Text Analytics Conference 2011.

T. Joachims. 1999. *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999.

Zornitsa Kozareva and Sujith Ravi 2011. *Name Ambiguity Resolution Using A Generative Model.* Proc. EMNLP2011 Workshop on Unsupervised Learning in NLP

A. Kilgarriff and J. Rosenzweig. 2000. *Framework and results for English Senseval.* Computers and Humanities, Special Issue on SENSEVAL, 15-48.

D. Klein and C. D. Manning. 2003. *Fast Exact Inference with a Factored Model for Natural Language Parsing.* In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10.

Y. K. Lee and H. T. Ng. 2002. *An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation.* In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 41-48.

J. Lehmann, S. Monahan, L. Nezda, A. Jung and Y. Shi. 2010. *LCC Approaches to Knowledge Base Population at TAC 2010.* In Proceedings of Text Analysis Conference 2010 Workshop.

H. Liu and H. Motoda. 2002. *On Issues of Instance Selection.* Data Mining and Knowledge Discovery, 6, 115-130. 2002

P. McNamee and H. Dang. 2009. *Overview of the TAC 2009 Knowledge Base Population Track.* In Proceedings of Text Analysis Conference 2009.

R. Mihalcea and A. Csomai 2007. *Wikify!: linking documents to encyclopedic knowledge.* In Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM â07, pages 233-242, New York, NY, USA. ACM. 2007.

R. Mihalcea and D. I. Moldovan 1999. *An automatic method for generating sense tagged corpora.* In Proceedings of AAAI Conference on Artificial Intelligence (AAAI). 1999

D. Milne and I. H. Witten. 2008. *Learning to link with Wikipedia* In Proceedings of the ACM Conference on Information and Knowledge Management, 2008.

S. Monahan, J. Lehmann, T. Nyberg, J. Plymale, and A. Jung 2011. *Cross-Lingual Cross-Document Coreference with Entity Linking* In Proceedings of Text Analysis Conference. 2011

H. T. Nguyen, T. H. Cao. 2008. *Named entity disambiguation on an ontology enriched by Wikipedia.* Research, Innovation and Vision for the Future, 2008. RIVF 2008.

D. Ploch 2011. *Exploring Entity Relations for Named Entity Disambiguation.* In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL). 2011.

S. P. Ponzetto and M. Strube. 2007. *Deriving a Large Scale Taxonomy from Wikipedia.* In Proceedings of the 22nd National Conference on Artificial Intelligence, Vancouver, B.C., 22-26 July, 2007, pp. 1440-1447.

B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov and A. Kirilov. 2004. *KIM - a Semantic Platform for Information Extraction and Retrieval.* In Journal of Natural Language Engineering, Vol. 10, Issue 3-4, Sep 2004, pp. 375-392

W. Radford, B. Hachey, J. Nothman, M Honnibal and J. R. Curran 2010. *Document-level Entity Linking: CMCRC at TAC 2010.* Proceedings of Text Analysis Conference. 2010.

D. Ramage, D. Hall, R. Nallapati and C. D. Manning. 2009. *Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora.* In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, 2009.

L. Ratinov, D. Roth, D. Downey and M. Anderson 2011. *Local and Global Algorithms for Disambiguation to Wikipedia.* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, 2011

D. Schohn and D. Cohn. 2000. *Less is more: active learning with support vector machines.* In Proceedings of International Workshop on Adaptive Text Extraction and Mining. 2000.

D Shen, J. Zhang, J. Su, G. Zhou and C. Tan 2004. *Multi-Criteria-based Active Learning for Named Entity Recognition.* In Proceeding of Annual Meeting of the Association for Computational Linguistics (ACL). 2004.

D. Spina, E. Amigo and J. Gonzalo 2011. *Filter Keywords and Majority Class Strategies for Company Name Disambiguation in Twitter.* In Proceeding of CLEF 2011, pp. 50-61

H. Srinivasan, J. Chen and R. Srihari 2009. *Cross document person name disambiguation using entity profiles.* In Proceeding of Text Analysis Conference. 2009

V. Vapnik. 1995. *he Nature of Statistical Leaning Theory.* Springer-Verlag, New York. 1995.

V. Varma, V. Bharat, S. Kovelamudi, P. Bysani, S. GSK, K. Kumar N, K. Reddy, K. Kumar, N. Maganti 2009. *IIIT Hyderabad at TAC 2009.* In Proceeding of Text Analysis Conference. 2009

A. Vlachos. 2008. *A Stopping Criterion for Active Learning.* Computer Speech and Language.22(3):295-312. 2008.

P. Wang and C. Domeniconi 2008. *Building Semantic Kernels for Text Classification using Wikipedia.* In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008

T. Zhang, K. Liu and J. Zhao 2011a. *NLPR TAC Entity Linking System at TAC2011.* In Proceedings of Text Analysis Conference. 2011

W. Zhang, J. Su, C. Tan and W. Wang. 2010. *Entity Linking Leveraging Automatically Generated Annotation.* In Proceedings of 23rd International Conference on Computational Linguistics. 2010

W. Zhang, Y. Sim, J. Su and C. Tan. 2011b. *Entity Linking with Effective Acronym Expansion, Instance Selection and Topic Modeling.* In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI. 2011.

W. Zhang, J. Su and C. Tan. 2011c. *A Wikipedia-LDA Model for Entity Linking with Batch Size Changing Instance Selection.* In Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 562-570, Chiang Mai, Thailand, November 8-13, 2011.

W. Zhang, J. Su, B. Chen, W. Wang, Z. Q. Toh, Y. C. Sim, Y. B. Cao, C. Y. Lin and C. L. Tan. 2011d. *I2R-NUS-MSRA at TAC 2011: Entity Linking.* In Proceedings of Text Analysis Conference. Nov 14-15, 2011. Maryland. US.

W. Zhang, J. Su, C. Tan, Y. Cao and C. Lin. 2012. *A Lazy Learning Model for Entity Linking Using Query-Specific Information.* In Proceedings of 24rd International Conference on Computational Linguistics. 2012

Y. Zhao, W. He, Z. Liu and M. Sun 2011. *THUNLP at TAC KBP 2011 in Entity Linking.* In Proceedings of Text Analysis Conference

Z. Zheng, F. Li, M. Huang, X. Zhu 2010. *Learning to Link Entities with Knowledge Base.* In Proceedings of Annual Conference of the North American Chapter of the ACL. 2010

C. Zirn, V. Nastase and M. Strube. 2008. *Distinguishing Between Instances and Classes in the Wikipedia Taxonomy.* In Proceedings of the 5th European Semantic Web Conference, Tenerife, Spain, 1-5 June 2008.