A Lazy Learning Model for Entity Linking Using Query-Specific Information

WeiZhang[1] *JianSu*[2]
ChewLimTan[1] *YunboCao*[3] *ChinYewLin*[3]
(1) National University of Singapore
(2) Institute for Infocomm Research, Singapore
(3) Microsoft Research Asia
{z-wei,tancl}@comp.nus.edu.sg, sujian@i2r.a-star.edu.sg,
{Yunbo.Cao,cyl}@microsoft.com

Abstract

Entity linking disambiguates a mention of an entity in text to a Knowledge Base (KB). Most previous studies disambiguate a mention of a name (e.g.*"AZ"*) based on the distribution knowledge learned from labeled instances, which are related to other names (e.g.*"Hoffman"*,*"Chad Johnson"*, etc.). The gaps among the distributions of the instances related to different names hinder the further improvement of the previous approaches. This paper proposes a lazy learning model, which allows us to improve the learning process with the distribution information specific to the queried name (e.g.*"AZ"*). To obtain this distribution information, we automatically label some relevant instances for the queried name leveraging its unambiguous synonyms. Besides, another advantage is that our approach still can benefit from the labeled data related to other names (e.g.*"Hoffman"*,*"Chad Johnson"*, etc.), because our model is trained on both the labeled data sets of queried and other names by mining their shared predictive structure.

Keywords: Entity Linking, Lazy Learning, Query-Specific Information.

Proceedings of COLING 2012: Technical Papers, pages 3089–3104,
COLING 2012, Mumbai, December 2012.

1 Introduction

Recently, more and more knowledge bases (KB) which contain rich knowledge about the world's entities such as Wikipedia [1], OpenCyc [2] and KIM [3] (Popov et al., 2004) have become available. These knowledge bases have been shown to form a valuable component for many natural language processing tasks such as knowledge base population (Ji and Grishman, 2011), text classification (Wang and Domeniconi, 2008), and cross-document coreference (Finin et al., 2009). However, to be able to utilize or enrich these KB resources, the applications usually require linking the mentions of entities in text to their corresponding entries in the knowledge bases, which is called entity linking task and has been proposed and studied in Text Analysis Conference (*TAC*) since 2009 (McNamee and Dang, 2009).

Given a mention of an entity in text and a KB, entity linking is to link the mention to its corresponding entry in KB. The major challenges of this task are name variation and name ambiguity. Name variation refers to the case that more than one name variation such as *alias, misspelling* and *acronym* refers to the same entity. For example, both *"48th State"* and *"The Grand Canyon State"* refer to *state of Arizona, U.S.*. Name ambiguity refers to the case that more than one entity shares the same name. For example, *"AZ"* may refer to *state of Arizona*, the Italian airline *Alitalia*, the country *Azerbaijan*, or other entries in KB that have the same name.

Most previous studies on entity linking used annotated data 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. (2011), all of the top systems from the participants in the shared task of *TAC-11*[4] used supervised learning approaches to solve this disambiguation problem.

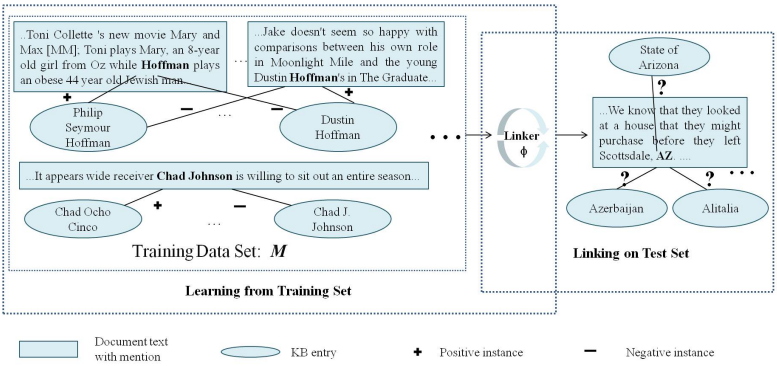

Figure 1: The System Architecture for Traditional Approaches. (*M* contains a certain number of names. *"Hoffman"* and *"Chad Johnson"* are two examples of them.)

However, as there are infinite number of entity names, it is impossible to manually create the labeled

[1] http://www.wikipedia.org/
[2] http://www.opencyc.org/
[3] http://www.ontotext.com/kim
[4] http://nlp.cs.qc.cuny.edu/kbp/2011/

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 1, the previous 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 ***M*** related to other names (e.g.*"Hoffman"*,*"Chad Johnson"*, etc.). Given the query, the previous systems are static at evaluation time and their training does not depend on the input query. However, Figure 2 illustrates the distribution of the labeled instances related to the three names in a feature space (*Bag of Words*, *Named Entities* and *Edit Distance*, the popular features used in previous work). We can see that the width and location of the gap to separate positive and negative instances for different names vary widely. Moreover, the positive-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 problem.

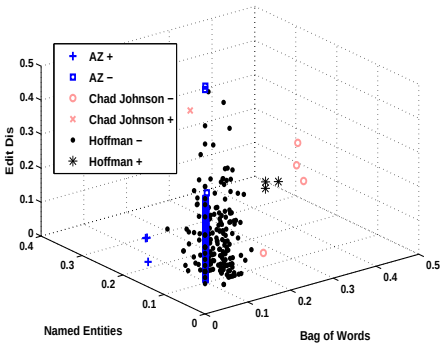

Figure 2: Instances Illustration in 3D Feature Space (Feature detail is in Table 2)

To narrow down the gap between the instance distributions related to labeled and queried names, this paper proposes a lazy learning model, in which generalization on the labeled data is delayed until a query is made. This allows the distribution information specific to queried name to be incorporated into the learning process. To obtain this distribution information, 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. We conduct evaluation on *TAC* data. Our experiments show that our proposed lazy learning model significantly improves entity linking over the state-of-the-art systems.

The remaining of this paper is organized as follows. In Section 2, we review the related work for entity linking. Section 3 elaborates the pre-processing stage to retrieve the possible KB entries

for a given mention. Section 4 presents our lazy learning for entity linking with query-specific information. Section 5 discusses a special case - NIL mentions . The experiments are shown in Section 6. Section 7 concludes the paper.

2 Related Work

As we have discussed, most of the previous entity linking work (Dredze et al., 2010; Lehmann et al., 2010; Zheng et al., 2010; Zhang et al., 2010; Ploch, 2011; Gottipati and Jiang, 2011; Han and Sun, 2011) fall into the traditional entity linking framework shown in Figure 1. Besides, the collaborative approach (Chen and Ji, 2011) tried to search similar queries as their query collaboration group by clustering texts. This differs from our method where we use the selective knowledge from unlabeled data.

In some other work, entity linking is also called *named entity disambiguation using Wikipedia* (Bunescu and Pasca, 2006; Cucerzan, 2007) or *Wikification* (Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ratinov et al., 2011). These two similar tasks link expressions in text to their referent Wikipedia pages. However, since Bunescu and Pasca (2006) used Wikipedia hyperlinks to train the SVM kernel, Cucerzan (2007) used Wikipedia collection and news stories as the development data, and all of the three *Wikification* work (Mihalcea and Csomai, 2007; Milne and Witten, 2008; Ratinov et al., 2011) generalized their ranker on the data generated from Wikipedia without considering the knowledge of the queried name, they also fall into the traditional entity linking framework and suffer from the distribution problem in the previous entity linking systems. Thus, we believe that our proposed lazy learning approach also can benefit these two tasks.

In WePS-3 [5], a task of *Online Reputation Management* was proposed (Amigo et al., 2010; Spina et al., 2011), which is the same with entity linking when KB only has one entry. Given a set of Twitter entries containing an ambiguous company name, and given the home page of the company, the task is to filter out Twitter entries that do not refer the company. 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. However, the lazy learning approach proposed in this paper 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.

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. This differs significantly from entity linking, which has a given entity list (i.e. the KB) to which we disambiguate the mentions.

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, the following pre-processing process is conducted to filter out irrelevant KB entries and select only a set of candidates that are potentially the correct match to the given query (a query consists of a name mention and its associated document text).

Because of name variation problem, it is ineffective to retrieve entity candidates by comparing the name strings of mention and KB entry. Thus, we need to use external world knowledge to build the name variation list for each KB entry. Since our experiment used a KB derived from Wikipedia, and

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

other KBs such as *KIM* and *OpenCyc* usually can be mapped to Wikipedia (Nguyen and Cao, 2008), we find the variations for the entries in KB by leveraging name variation sources in Wikipedia: *"titles of entity pages"*, *"disambiguation pages* [6]*"*, *"redirect pages* [7]*"* and *"anchor texts"*. Then, the candidates can be selected by comparing the name string of the mention with the name strings in the variation list of each KB entry. The KB entry with a name string which matches the name of the mention is considered as a candidate. In addition, as a pre-processing step, we prefer a candidate set with high recall. Thus, to increase the recall, we find more candidates by selecting the KB entry if its name string contains the name string of the mention (e.g. *"Cambridge, Massachusetts"* contains *"Cambridge"*).

4 Lazy Learning for Linking

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 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:

$$e = \arg\max_{c_i \in C_q} \phi_q(q, c_i) \tag{1}$$

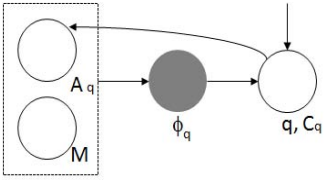

Figure 3: Lazy Learning Framework

In this section, we present our lazy learning model to solve this optimization problem. As shown in Figure 3, our process of the resolution 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.

As shown in Figure 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 problem shown in Figure 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

[6] http://en.wikipedia.org/wiki/Wikipedia:Disambiguation
[7] http://en.wikipedia.org/wiki/Wikipedia:Redirect

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 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.

4.1 Distribution Information A_q for Queried Name

In this section, we propose to obtain the distribution information for the queried name by automatically labeling some instances A_q for it. Following our work Zhang et al. (2010), which automatically generates training data for entity linking, we automatically label the instances related to the queried name based on its unambiguous synonyms.

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 3 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 1).

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 1: Unambiguous Variations for the Candidates of *"AZ"*

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 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) ...}*.

The data set A_q contains the query-specific information and such kind of information is not available in the training data M, such as the query-specific context features: words "*Airlines*" and "*State*" learned from A_q will be helpful for the disambiguation of "*AZ*"

4.2 Linear Function ϕ_q

In this section, we formulate the disambiguation function ϕ_q in Eq. 1 as follows,

$$\phi_q(q, c_i) = u^T \mathbf{X}_i \tag{2}$$

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 transfer the instance *(d_i, m_q, e_i)* to the feature vector $\boldsymbol{X}_i^q$. 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,

$$\hat{u} = \arg\min_u \left(\frac{1}{n^{(q)}} \sum_{i=1}^{n^{(q)}} L(u^T \mathbf{X}_i^q, Y_i^q) + \lambda \|u\|^2 \right) \tag{3}$$

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}.

Transfer *(q,c_i)* to Feature Vector $\boldsymbol{X}$. The features we adopted to construct $\boldsymbol{X}_i$ from *(q,c_i)* include 13 typical feature types used in *TAC* (Lehmann et al., 2010; Ji and Grishman, 2011), and the features are divided into four groups, contextual features (**CF**), semantic features (**SeF**), surface features (**SuF**) and generation source (**GS**) (see Table 2).

4.3 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 4.1). 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

Name	Description
*Contextual Features (**CF**)*	
Bag of Words	The cosine similarity (tf.idf weighting) between query document and text of the candidate.
Similarity Rank	The inverted cosine similarity rank of the candidate in the candidate set.
Co-occurring NEs	Number of the same named entities appearing in query document and the text of the candidate.
*Semantic Features (**SeF**)*	
NE type	True if NE type (i.e. Person, GPE, Organization) of the query and the candidate is consistent.
Topic Similarity	Topic similarity between query document and text of candidate obtained by LDA (Blei et al., 2003)
*Surface Features (**SuF**)*	
Surface Match	True if the query matches the title of the candidate
Substring Match	True if the title of the candidate begins with the query (e.g. *"Beijing China"* and *"Beijing"*)
Acronym	True if the query is an acronym for the title of the candidate (e.g. *"NY"* and *"New York"*)
Word Match	Number of the same words between the title of the candidate and the query
Word Miss	Number of the different words between the title of the candidate and the query
Edit Distance	Levenshtein distance between name strings of the query and the candidate's title
*Generation Source (**GS**)*	
Wikipedia Source	True for each Wikipedia source (i.e. "entity pages", "disambiguation pages", "redirect pages" and "anchor texts" (Section 3)) which generates the candidate
String Match	For the candidate not generated from Wikipedia source, true if it is generated from full match.

Table 2: Feature Set for Disambiguation.

of China" and the second mention "NY" refers to entity "the city of New York", they have similar surface features (e.g. feature "acronym" is true), certain 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 Section 1, 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*.

The Structural Learning Algorithm. *Structural learning* (Ando and Zhang, 2005b) 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 $(\mathbf{X}_i^l, \mathrm{Y}_i^l)$. 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,

$$\mathbf{u}_l = \mathbf{w}_l + \Theta^T \mathbf{v}_l \tag{4}$$

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,

$$\underset{\Theta,\mathbf{w}_l,\mathbf{v}_l}{\arg\min} \sum_{l=1}^{K} \left(\frac{1}{n^{(l)}} \sum_{i=1}^{n^{(l)}} L\left(\left(w_l + \Theta^T v_l \right)^T \mathbf{X}_i^{(l)}, Y_i^{(l)} \right) + \lambda \|w_l\|^2 \right) \tag{5}$$

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.

Alternating Structure Optimization. The Θ optimization problem in Eq. 5 can be approximately solved by the following alternating structure optimization procedure (Ando and Zhang, 2005a),

> 1: Learn K weight vectors u_l' for all the K problems on their corresponding instances independently using *empirical risk minimization* (similar with Eq. 3).
>
> 2: Let $U' = [u'_1, \ldots u'_K]$ 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 $\hat{\Theta}$

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 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 1) with auto-labeled instances A_q is our target prediction problem, which is the problem we are aiming to solve.

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. The relevant prediction problems may only has shared structure with target problem over certain features. In this paper, 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 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 $\boldsymbol{u}$ for the queried name in Eq. 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 problem (see Figure 2) in the previous work only using M.

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. 1 is extended to:

$$e = \arg\max_{c_i \in C_q \cup NIL} \phi_q(q, c_i) \tag{6}$$

6 Experiments and Discussions

6.1 Experimental Setup

In our study, we use *TAC-10*[8] KB and document collection to evaluate our approach for entity linking. The KB is derived from Wikipedia, which contains 818,741 different entries and the document collection contains 1.7 million documents from newswire and blog text. Each KB entry consists of the Wikipedia Infobox [9] and the corresponding Wikipedia page text.

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

[9] http://en.wikipedia.org/wiki/Template:Infobox

The test set of *TAC-10* has 2,250 mentions across three named entity (**NE**) types: Person (**PER**), Geo-Political Entity (**GPE**) and Organization (**ORG**). The documents containing these mentions are from the document collection above. 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.

We 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.

6.2 Statistics of Data Set A_q

To minimize the distribution gap discussed in Section 1, 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, we find that 359 queries are assigned empty candidate sets (i.e. $C_q = \emptyset$) by the process described in Section 3. 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 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% 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.

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.

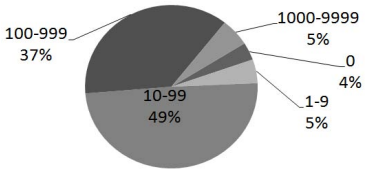

Figure 4: Proportions of the Queries Based on the Sizes of their Corresponding A_q

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"* 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) separately may be more effective. Hence, we experimented with both Θ configuration in Eq. 5 and Θ configuration, learning a Θ_j for each feature group j separately in Eq. 7.

$$\sum_{l=1}^{K}\left(\frac{1}{n^{(l)}}\sum_{i=1}^{n^{(l)}}L\left(w_l^T\mathbf{X}_i^{(l)}+\sum_{j\in F}v_l^{(j)\,T}\Theta_j\mathbf{X}_i^{(l,j)},Y_i^{(l)}\right)+\lambda\|w_l\|^2\right) \tag{7}$$

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 experimented 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 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. Comparison of *F={CF, SeF, SuF, GS}* (learning Θ_j for these four feature groups separately by Eq. 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 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,...}).

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 6.3.

Table 3 shows the performances of three baseline methods and our approach with overall accuracy as well as accuracy on five subsets of the 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. 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

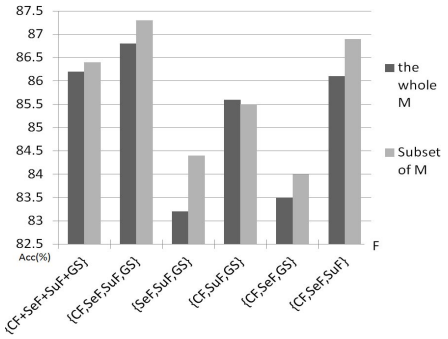

Figure 5: Accuracy on Development Set (As Θ has one parameter - its dimensionality h, the performance here is the ceiling performance obtained on the development set at the best dimensionality in {10, 50, 100,...})

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 problem (see Section 1) 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.

	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 3: Micro-averaged Accuracy on Test Set

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 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

problem in the previous work and indeed improved the disambiguation systems. On the *TAC-11* data set, we obtain the similar result. We apply our method in this paper to our system in *TAC-11* (Zhang et al., 2011), which achieves 87.6% with a 1.3% improvement.

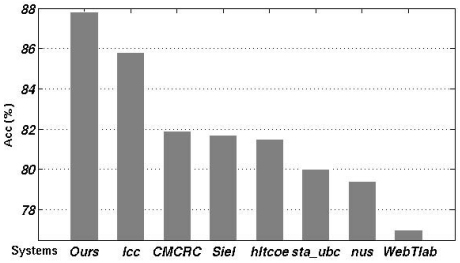

Figure 6: A Comparison with *TAC-10* Systems

6.6 Response Time

Our lazy learning delays the generalization on the labeled data until receiving the query. Hence, comparing with previous work, it increases the response time of the system for each query. However, many of the entity linking applications such as *knowledge base population* do not require real-time user interaction, and therefore they are time-insensitive applications. For those potential time-sensitive applications, we can calculate u_i' (i=1,...,K) in the optimization procedure of Eq. 5 before receiving the query and extract the features from the candidates in parallel. In our experiment, when using 8 *CPUs* (*1.86GHz*) for the multi-thread configuration and using Lemur/Indri[10] to index and search document for generating A_q, our approach can disambiguate the mentions in an article as fast as the baseline method (*M (Eq.3)*) using a single *CPU*.

7 Conclusions and Future Work

With the goal of achieving higher disambiguation performance, our focus was to solve the distribution problem 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.

Acknowledgments

This work is partially supported by Microsoft Research Asia eHealth Theme Program.

[10] http://www.lemurproject.org/indri.php

References

Amigo, E., Artiles, J., Gonzalo, J., Spina, D., Liu, B., and Corujo, A. (2010). Weps3 evaluation campaign: Overview of the on-line reputation management task. In *CLEF (Notebook Papers/LABs/Workshops) 2010*.

Ando, R. K. (2006). Applying alternating structure optimization to word sense disambiguation. In *Conference on Natural Language Learning (CoNLL)*.

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

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

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

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

Blei, D., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. In *Journal of Machine Learning Research 3:993-1022, 2003*.

Bunescu, R. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *the 11th Conference of the European Chapter of the Association for Computational Linguistics*.

Chen, Z. and Ji, H. (2011). Collaborative ranking: A case study on entity linking. In *the Conference on Empirical Methods in Natural Language Processing*.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *the Conference on Empirical Methods in Natural Language Processing*.

Dredze, M., McNamee, P., Rao, D., Gerber, A., and Finin, T. (2010). Entity disambiguation for knowledge base population. In *23rd International Conference on Computational Linguistics*.

Finin, T., Syed, Z., Mayfield, J., McNamee, P., and Piatko, C. (2009). Using wikitology for cross-document entity coreference resolution. In *AAAI Conference on Artificial Intelligence*.

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

Han, X. and Sun, L. (2011). A generative entity-mention model for linking entities with knowledge base. In *the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers (pp. 115-132)*.

Ji, H. and Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *the 49th Annual Meeting of the Association for Computational Linguistics*.

Ji, H., Grishman, R., and Dang, H. T. (2011). An overview of the tac2011 knowledge base population track. In *Text Analytics Conference*.

Lehmann, J., Monahan, S., Nezda, L., Jung, A., and Shi, Y. (2010). Lcc approaches to knowledge base population at tac 2010. In *Text Analysis Conference 2010 Workshop*.

McNamee, P. and Dang, H. (2009). Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference*.

Mihalcea, R. and Csomai, A. (2007). Wikify!: linking documents to encyclopedic knowledge. In *the sixteenth ACM conference on Conference on information and knowledge management*.

Milne, D. and Witten, I. H. (2008). Learning to link with wikipedia. In *the ACM Conference on Information and Knowledge Management*.

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

Ploch, D. (2011). Exploring entity relations for named entity disambiguation. In *49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., and Kirilov, A. (2004). Kim - a semantic platform for information extraction and retrieval. In *Journal of Natural Language Engineering*.

Ratinov, L., Roth, D., Downey, D., and Anderson, M. (2011). Local and global algorithms for disambiguation to wikipedia. In *the 49th Annual Meeting of the Association for Computational Linguistics*.

Spina, D., Amigo, E., and Gonzalo, J. (2011). Filter keywords and majority class strategies for company name disambiguation in twitter. In *CLEF*.

Wang, P. and Domeniconi, C. (2008). Building semantic kernels for text classification using wikipedia. In *14th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Zhang, W., Su, J., Chen, B., Wang, W., Toh, Z., Sim, Y., Cao, Y., Lin, C. Y., and Tan, C. L. (2011). I2r-nus-msra at tac 2011: Entity linking. In *Text Analysis Conference*.

Zhang, W., Su, J., Tan, C., and Wang, W. (2010). Entity linking leveraging automatically generated annotation. In *23rd International Conference on Computational Linguistics*.

Zheng, Z., Li, F., Huang, M., and Zhu, X. (2010). Learning to link entities with knowledge base. In *Annual Conference of the North American Chapter of the ACL*.