Context-Aware In-Page Search

林昱豪 Yu-Hao Lin, 劉郁蘭 Yu-Lan Liu, 顏孜羲 Tzu-Xi Yen, 張俊盛 Jason S. Chang

國立清華大學資訊工程學系
Department of Computer Science
National Tsing Hua University
{ catonmars.lin, ikulan12, joseph.yen, jason.jschang}@gmail.com

Abstract

In this paper we introduce a method for searching appropriate articles from knowledge bases (e.g. Wikipedia) for a given query and its context. In our approach, this problem is transformed into a multi-class classification of candidate articles. The method involves automatically augmenting smaller knowledge bases using larger ones and learning to choose adequate articles based on hyperlink similarity between article and context. At run-time, keyphrases in given context are extracted and the sense ambiguity of query term is resolved by computing similarity of keyphrases between context and candidate articles. Evaluation shows that the method significantly outperforms the strong baseline of assigning most frequent articles to the query terms. Our method effectively determines adequate articles for given query-context pairs, suggesting the possibility of using our methods in context-aware search engines.

Keywords: entity linking, word sense disambiguation, Wikipedia, support vector machine, search engine

1 Introduction

Today we surf the Internet through search engines most of the time. With the explosive growth of web pages, the accuracy and relevancy of search results have become ever more important. Traditional search engines accept keywords, and return a page full of possible relevant results. Then users can click one of the results to visit the sites they are interested in. We call this type of search “keyword-search”. Today, almost all search engines are keyword-based.

However, various classes of results mixed in the search results. For example, when a user query the search engine with the keyword “apple”, the search results comprise of two major class, “Apple Inc.”, the computer company, and “apple”, a kind of fruit. With only one keyword, even state-of-the-art keyword-based search engines could not distinguish between different search intents. Unlike keyword search, context-aware search assume each query is

associated with a context.

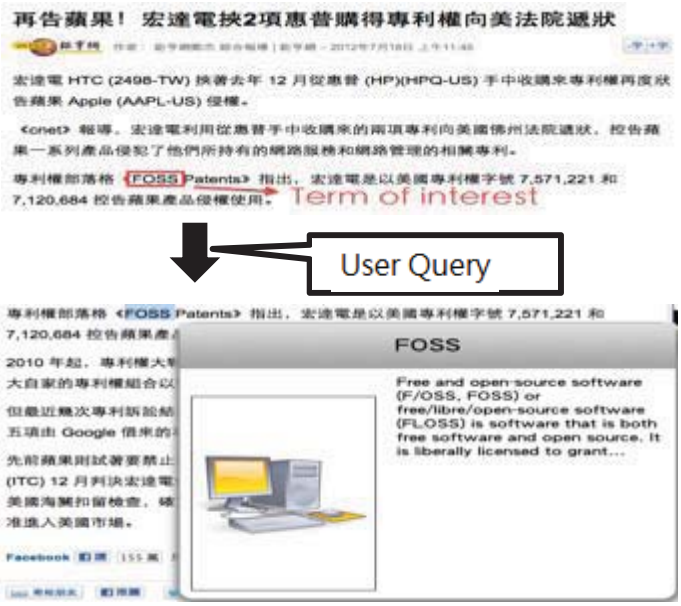

Figure 1. An example of context-aware search

John McCarthy (September 4, 1927 – October 24, 2011)[1][2][3][4][5][6] was an American computer scientist and cognitive scientist. He invented the term "artificial intelligence" (AI), developed the Lisp programming language family, significantly influenced the design of the ALGOL programming language, popularized timesharing, and was very influential in the early development of AI.

McCarthy received many accolades and honors, including the Turing Award for his contributions to the topic of AI, the United States National Medal of Science, and the Kyoto Prize.

Figure 2. The mention “John McCarthy” and its context.

In this paper, we present a prototypical system, In-Page Search, that automatically extract context information and use them to disambiguate ambiguous queries. Users could select the terms they are interested in, and then with a click of the mouse, the In-Page Search system shows a pop-up window with the most relevant results for the given context.(See Figure 1.) In-Page Search is similar to the “entity-linking problem”, which has long been an active research topic in IR and Database community. Entity-linking problem could be informally described as follows: given a knowledge base, in which every entry is an entity and its associated information. Given a mention and the context with the mention, determine the correct entity that the given mention really links to. For example, Figure 2 shows the mention “John McCarthy” and it’s context, in a knowledge base, there are more than 10 entities which may be linked to “John McCarthy”. The problem is determining the correct entity to link to. Intuitively, entity-linking could be considered a Named-Entity Disambiguation problem or more generally, a word sense disambiguation problem.

In our approach, we also exploit the cross-language features in multi-language knowledge bases. This method augments information in one language with other languages in the same

knowledge base to cope with the data sparseness problem which may be a problem for a language with less data. We discuss this multi-language model and the definitions of various link-based similarity measures in Chapter 3.

At run-time, In-Page Search starts with a query together with its context page submitted by the user. The system then extracts context terms and transforms them into machine-readable features. Finally, the system uses a *SVM* model (Chang and Lin, 2011) trained on a knowledge base to determine which entity in the knowledge base should be linked to the current query, and output a summarized abstract of this entity to the user. The results could be further augmented for other purposes. For example, for the input links to a geographic entity, we could show the location using a map application.

The rest of this thesis is organized as follows. We review the related work in the following chapter. Then we describe our preprocessing and runtime algorithm in Chapter 3. We then report on the experimental setup and compare our results to various baselines in Chapter 4. Conclusions are provided in Chapter 5 along with the directions of future work.

2 Related Work

Search engines and related technology has long been an active research topic in information retrieval and natural language processing. Most modern search engines (e.g. *Google, Bing*, and *Yahoo!*) accept keyword or keyphrase as input. Today keyword search engines have excellent performance in terms of both results relevancy and response time. However, keyword search engines do not consider a query may come with a context, so they could not distinguish between different search intents. With the rise of the mobile web, some search engines have evolved to provide better user experience. One reprehensive example is the *Google Now* feature of mobile edition of *Google*. While accepting user's voice input, it extracts user's context information such as GPS location, user's schedule recorded on calendar application, and the contact information on user's cell phone. Thus, *Google Now* can analyze user's search intent and provide the most relevant information using these contexts.

Previously, much effort has been made in research on word sense disambiguation based on machine learning (Black, 1988; Hearst, 1991; Leacock, Towell, and Voorhees, 1993; Bruce and Wiebe, 1994). Yarowsky (Yarowsky, 1992) uses a Naïve Bayesian classifier trained on Roget's thesaurus to classify words with given context into its sense category. They use class-based salient words list provided by Roget's thesaurus as features and tuning weight by counting the frequencies of surrounding salient words in context. While achieving high accuracy, this research can be viewed as prototypical framework of most machine learning WSD systems. These approaches often rely on sense-labeled corpus. Although supervised machine learning WSD algorithms frequently gives high performance, however, sense-labeled

corpus is not always available. Compared to our approach, we use Wikipedia as our corpus, its cross-lingual nature enables us to augment smaller knowledge base with other languages.

An important branch of WSD is entity-linking. While WSD focuses on linking word to its correct sense given context, entity-linking systems focus on linking mentions of entities (often named-entities) to its correct entry in a given knowledge base. "Wikify" (Mihalcea and Csomai, 2007; Milne and Witten, 2008) is an example of entity-linking systems. These systems automatically augment user's input texts with hyperlinks to Wikipedia entries. For example, imagine Figure 2 with links removed, these systems will automatically detect them with anchors links to proper Wikipedia articles (e.g. John McCarthy in Figure 2 links to John McCarthy (computer scientist) in Wikipedia.). Mihalcea's system decomposes these task into two procedural: keyphrase extraction and word sense disambiguation. They achieve WSD by computing various linguistic features except the "Keyphraseness": how frequently one phrase in Wikipedia being hyperlinks.

Milne and Witten's system disambiguates mentions by incorporating more link-based measures. They apply normalized Google Distance (Cilibrasi and Vitanyi, 2007) to compute relatedness between two Wikipedia articles, and training machine learning models. Unlike Mihalcea's system, they first disambiguate possible candidates in input document, and then use information from this pass of disambiguation to aid keyphrase extraction. Their system has good performance both on Wikipedia articles and wild-life news pages.

Compared to our system, most entity-linking system developed their method on English, so they could not directly apply to languages that need segmentation pre-processing. To apply our method to CJK languages, we use a scheme similar in (Milne and Witten, 2008) to transform context page into vector of context entities. In addition, we extend traditional link-based measure to a cross-lingual augmented knowledge base. To the best of our knowledge, such technique hasn't been shown in previous systems.

3 Method

Understanding a user's search intent basing solely on query term (e.g., 蘋果) is a challenging task. Short query terms typically have more than one sense which leading to multiple entities in the knowledge base that could be linked to. To assign adequate entity for a given query, a promising method is to compute the similarity between a query's context and candidate entities' description, and returning the most similar entity (e.g. 蘋果公司 for 蘋果) in the context of a computer-related Chinese article.

3.1 Problem Statement

We focus on the essential step of determining user's search intent: choosing the appropriate

entity in the knowledge base for the given query. Once the entity has determined, the system returns information of this entity in various ways (e.g. text description, image, audio, video). In a *Wikipedia-like* knowledge base, we treat each document as an entity, its description page as the context, and hyperlinks in this page as query terms. With the hyperlinked nature of such a knowledge base, we train a classifier which estimate the similarity of link structure between each query term's context, and determine whether a query term and an entity (i.e. the article titles) should be linked together. Thus, the problem of context-aware search is transformed to an entity-linking problem. We now formally state the problem we are addressing by first giving a definition of *Wikipedia-like* knowledge base.

A *Wikipedia-like* knowledge base is a collection of documents, each document should describe an unique concept with hyperlinks, *inter-wiki links* and *disambiguation pages* which list possible sense of an ambiguous term.

Problem Statement: We are given a set of Wikipedia-like knowledge bases $\boldsymbol{KB}=\{ \boldsymbol{kb}_1 ,\ldots, \boldsymbol{kb}_n | \ n \geq 1 \ \}$ (e.g., {Chinese Wikipedia, English Wikipedia}), a query term $\boldsymbol{q}$, a context document $\boldsymbol{c}$ of $\boldsymbol{q}$, and a knowledge base $\boldsymbol{kb}_j \in \boldsymbol{KB}$, where $\boldsymbol{q}$ should be searched. Our goal is to assign an adequate document $\boldsymbol{e}_i$, where $\boldsymbol{e}_i \in \boldsymbol{kb}_j = \{\boldsymbol{e}_1,\ldots, \boldsymbol{e}_j\}$ and $\boldsymbol{e}_1,\ldots, \boldsymbol{e}_j$ are candidate senses. For this, we compute the link structure similarity between each document pair ($\boldsymbol{c}$, $\boldsymbol{e}$), where $\boldsymbol{e}$ is in $\boldsymbol{kb}_j$, and then train a classifier to determine which ($\boldsymbol{c}$, $\boldsymbol{e}$) pair should be linked together.

3.2 Learning to Link with Wikipedia-like Databases

We attempt to resolve the sense ambiguity of a given query term by learning link structure characteristics from a collection of <Term, Entity> pairs in a Wikipedia-like knowledge base. Our learning process is shown in Figure 3.

```
(1)    Generate  Candidate Term-Entity Pairs From Knowledge Base (Section 3.2.1)
(2)    Augment Knowledge Bases by Inter-Wiki Links (Section 3.2.2)
(3)    Train Binary SVM Classification Model (Section 3.2.3)
```

Figure 3 Outline of the training process.

3.2.1 Generate Candidate Term-Entity Pairs From Knowledge Base

In the first stage of the learning process (Step (1) in Figure 3), we generate candidate <Term, Entity> pairs from ***KB***. Once the candidate pairs have been computed and stored, the *In-Page Search* system could use them to efficiently retrieve possible entities of a given query, instead of comparing every ***e*** in ***KB***. For example, given the query "蘋果", we retrieve { <"蘋果", "蘋果公司">, <"蘋果", "麥金塔電腦">, <"蘋果", "蘋果(水果)">, <"蘋果","蘋果日報">}, and then, these four entities will be disambiguated. We compute these pairs from ***KB*** using a hyperlink's anchor text and its destination entity. The rationale behind computing <Term, Entity> pairs using anchor texts is that anchor texts reflect how people mentioning

entities in written articles.

The input to this stage is a set of *Wikipedia-like* knowledge base ***KB***. With their hyperlinked nature, we could compute <Term, Entities> pairs easily. To provide broader coverage of query, we also take into account the redirect links and disambiguation pages.


蘋果公司

維基百科，自由的百科全书

蘋果股份有限公司（Apple Inc.，NASDAQ：AAPL，LSE：ACP），原稱蘋果電腦公司（Apple Computer, Inc.）總部位於美國加利福尼亞的庫比蒂諾，核心業務是電子科技產品，目前全球電腦市場佔有率為7.96%。蘋果的Apple II於1970年代助長了個人電腦革命，其後的Macintosh接力於1980年代持續發展。最知名的產品是其出品的Apple II、Macintosh電腦、iPod音樂播放器、iTunes商店、iPhone手機和iPad平板電腦等。它在高科技企業中以創新而聞名。

蘋果公司原稱蘋果電腦公司，於2007年1月9日舊金山的Macworld Expo上宣佈改為現名。

2010年9月23日，據CNN報導，蘋果公司曾短暫成為第二市值的上市公司，僅次於美國的石油業巨擘埃克森美孚，超過位於第三的中石油[1]。2011年8月9日周二經約股市盤中，蘋果公司市值曾短暫超越美國石油巨頭埃克森美孚，成為世界市值第一的上市公司[2]，次日周三收盤時蘋果公司再一次超過埃克森美孚，直到該周四埃克森美孚才重新奪回市值第一的名號[3]。

Figure 4. An input document from an Wikipedia-like knowledge base

Term	Entity
NASDAQ	那斯達克
LSE	倫敦證券交易所
CNN	有線電視新聞網

Term	Entity
中石油	中國石油
加利福尼亞	加利福尼亞州
蘋果電腦公司	蘋果公司

Table 1. Samples of <Term, Entity> pairs constructed from Figure 4.

The output of this stage is a collection of <Term, Entity> pairs of a certain knowledge base. Some <Term, Entity> pairs, automatically constructed, are shown in Table 1. Figure 5 shows the algorithm for computing <Term, Entity> pairs from a *Wikipedia*-like database.

```
procedure GenerateTermEntityPairs(kb)
(1)          docs = GetDocuments(kb)
(2)          list = emptyList
(3)          for each ei in docs
(4)               links = GetLinks(ei)
(5)               title = GetTitle(ei)
(6)               if ei is Disambiguation Page
(7)                    for each target in links
(8)                         list += <title,target>
(9)               else
(10)               for each <anchor,target> in links
(11)                    list += <anchor,target>
(12)         hist=Histogram(list)
(13)         return hist
```

Figure 5. Generating <Term, Entity> pairs.

In Step (1) of the algorithm we retrieve the list of all articles in ***kb***. Then we iterate through all articles. For each article, we first identify all hyperlinks and title of article (Steps (4), (5)). If this document is a *disambiguation page*, for each hyperlinks in this page, we add <document

title, link target> to a temp list. Otherwise we add <anchor text, link target> to the temp list (Steps (6)~(11)). Finally, we compute the histogram of the temp list, where every entry is a <Term, Entity> pair and its frequency (Steps (12)). An example of results is shown in Table 2.

Table 2. <Term, Entity> pairs of '蘋果'

Term	Entity	Frequency
蘋果	蘋果 (植物)	149
蘋果	蘋果公司	23
蘋果	蘋果 (電影)	1
蘋果	麥金塔電腦	1

3.2.2 Augmenting Knowledge Base using Inter-Wiki Links

In the second stage of the learning algorithm (Step (2) in Figure 3), we augment each *Wikipedia*-like knowledge base in ***KB*** using *inter-wiki* links. Consider *Chinese Wikipedia* and *English Wikipedia*, *language links* among them link two document describe the same entity together. For example, "*麥金塔電腦*" in *Chinese Wikipedia* and "*Macintosh*" in *English Wikipedia*. By linking one entity to its corresponding entity in other knowledge base, we could combine the knowledge to obtain a richer representation of information of each entity. For two imbalanced knowledge bases (e.g. *Chinese Wikipedia* and *English Wikipedia*), our algorithm could augment the one with less information using the one with more information.

In a *Wikipedia*-like knowledge base, each article can be viewed as a concept (i.e. entity). From hyperlinks in documents, we could build a directed graph of the entire knowledge base, in which nodes denote articles, the edge indicate an article mentions another via hyperlinks. Thus, out-going edges of a node point to other articles mentioned in the article represented by the node, while in-coming edges of a node indicate other articles mentioning the node. We call these two edges *out-links* and *in-links* respectively (See Figure 6.).

Figure 6. A link graph. Blue edges denote outlinks, green edges denote inlinks, orange edges denote both inlinks and outlinks.

The input of this stage is two *Wikipedia*-like knowledge bases (e.g. <*Chinese Wikipedia*, *English Wikipedia*>, we augment the first knowledge base using the second one. The output of this stage is an augmented knowledge base, in which each document is augmented.

```
procedure AugmentKB(kb_c, kb_e)
(1)          docs = GetDocuments(kb_c)
(2)          for each e_cn in docs
(3)              <olinks_cn,ilinks_cn> = <GetOLinks(e_cn),GetILinks(e_cn)>
(4)               if InterlinkOf(e_cn) exists:
(5)                   e_en=GetDocument(kb_e,InterlinkOf(e_cn))
(6)                   <olinks_en,ilinks_en> = <GetOLinks(e_en),GetILinks(e_en)>
(7)                   CombineLinks(olinks_cn,olinks_en)
(8)                   CombineLinks(ilinks_cn,ilinks_en)
```

```
procedure CombineLinks(link_cn,link_en)
(9)         for each lk_en in link_en:
(10)        if InterlinkOf(lk_en) exists:
(11)              lk_cn=translate(lk_en,InterlinkOf(lk_en))
(12)              link_cn+=lk_cn
(13)              link_en-=lk_en
(14)    AddToKB(<link_cn, link_en>)
```

Figure 7. The augmentation process.

Figure 7. shows the knowledge base augmenting process. In Step (1) of the algorithm, we retrieve the list of all articles in $\boldsymbol{kb_c}$. For each article, we first examine whether it has an *inter-link* points to its corresponding entity in $\boldsymbol{kb_e}$. If the result is negative, we leave the current article unchanged without augmentation. In Step (5), we identify the corresponding article in $\boldsymbol{kb_e}$ by looking at the target, $\boldsymbol{e_{en}}$ of *inter-wiki* link of $\boldsymbol{e_{cn}}$. Then, we retrieve all *out-links* and *in-links* of $\boldsymbol{e_{en}}$ and carry out the *CombineLinks* procedure with both kinds of links (Step (6), (7), (8)). In the *CombineLinks* procedure, we iterate through all links in $\boldsymbol{link_{en}}$, and then determine if the link (i.e. $\boldsymbol{lk_{en}}$) has an *inter-link* (Step (10)). If such an *inter-link* exists, we "translate" the link by replacing $\boldsymbol{lk_{en}}$ with $\boldsymbol{lk_{cn}}$, a hyperlink point to destination of the *inter-link* and has anchor text of destination title. Finally we add the translated link to the original set of link (i.e. $\boldsymbol{link_{cn}}$), and store them in database. Note that the $\boldsymbol{link_{en}}$ is also stored in $\boldsymbol{kb_c}$ (Step (14)). We do that to support cross-lingual entity-linking. Once the augmentation has been done, each article in $\boldsymbol{kb_c}$ has two link sets from each knowledge base. For articles with *inter-links*, the performance of entity-linking could be improved from the augmentation algorithm.

3.2.3 Training the Binary SVM Model

In the third and final stage of the learning process, we train a Link Similarity Model based on the link graph of *Wikipedia*-like knowledge base articles. To determine which entity to be linked given query term $\boldsymbol{q}$, we compute link graph similarity between context $\boldsymbol{c}$ of $\boldsymbol{q}$ and candidate entities' articles, and transform them to feature vectors to train a binary SVM classifier. In the rest of this section, we first explain the Link Similarity Model, which is used to estimate the similarity between two entities, and show how we incorporate the Link Similarity Model with SVM.

Consider link graphs in Figure 6. We compute similarity between two link graphs which

has vertices v_a, v_b as central node respectively using following equations:

$$Sim(v_a, v_b) = \frac{|E_a \cap E_b|}{\min(\|E_a\|, \|E_b\|)} \qquad (1)$$

In Eq. (1) $\boldsymbol{E_a}$, $\boldsymbol{E_b}$ denote the edges of $\boldsymbol{v}_a$, $\boldsymbol{v}_b$ respectively. The interpretation of Eq. (1) is that we compute the number of edges in common with both vertices respectively, and normalize it using edges of smaller graph constructed from v_a and v_b. In order to make range of Eq. (1) lies in [0, 1], we choose to normalize by smaller graph. Thus, bigger value means bigger similarity between two vertices.

Given training data, we use Eq. (1) to compute features from training data and use them to train a binary SVM classifier. The procedure is shown in Figure 8.

```
procedure GenerateSVMInput(kb)
(1)     <Terms, Articles>= RandomTermArticles(kb)
(2)     for each <term, article> in <Terms, Articles>
(3)         candidates = GetTermEntity(term)
(4)         for each <term, entity> in candidates
(5)              <lp, olinkSim, ilinkSim> =  extractFeatures(article, entity)
(6)              if entity==TargetOf(term)
(7)                   AddToOutput(<1, lp, olinkSim, ilink>)
(8)              else
(9)                   AddToOutput(<0, lp, olinkSim, ilink>)
```

Figure 8. Training SVM Classifier.

In Step (1) we retrieve a list of <Term, Article> pairs in which Term is an anchor text of randomly chosen hyperlink in Article, a randomly chosen article from ***kb***. We treat Terms as query terms, and Articles as their contexts. Using <Term, Entity> pairs computed in 3.2.1, we can get candidates <Term, Entity> pairs (Step (3)). Then we iterate through them (Step (4)). In Step (5), for each <Term, Entity> pairs, we extract three features from them:

lp: The link probability defined as P(Entity|Term), which could be easily computed since we have stored the histograms in 3.2.1.

olinkSim: The link similarity considering only *outlinks*, i.e. $\boldsymbol{Sim_l}$(article, entity).

ilinkSim: Likewise, the link similarity by considering only *inlinks*.

In the computation of link similarity, notice that since the knowledge base has been augmented in 3.2.2, each articles has two link sets. We utilize a set of constant coefficient $<\boldsymbol{\alpha_1}, \boldsymbol{\alpha_2}, \boldsymbol{\alpha_3}>$ to interpolate between similarity computed from $<\boldsymbol{link_{en}}, \boldsymbol{link_{cn}}, \boldsymbol{link_{cn0}}>$, where $\boldsymbol{link_{cn0}}$ is the unaugmented link set of $\boldsymbol{kb_{cn}}$. Finally we examine whether the target of term's hyperlink equals entity, if the result is positive, we add the current feature vector to the input of SVM with positive example, otherwise with negative example (Steps (6)~(9)).

3.3 Run-Time Entity Linking

Once the SVM model is constructed, we are ready to classify or disambiguate query terms to corresponding entities in ***KB***. We associate adequate entities with given query terms and context using the procedures in Figure 9.

```
procedure ClassifyTerm(q, context, kb)
(1)     ctxEntity = transformContext(context, kb)
(2)     candidates = GetCandidateEntities(q)
(3)     for each entity in candidates
(4)          feature = <LinkProb(entity), olinkSim(entity,
        ctxEntity), ilinkSim(entity, ctxEntity)>
(5)          if SVMPredict(feature) is positive
(6)               AddToResultCandidate(entity)
(7)          else
(8)               continue
(9)     if ResultCandidate is empty
(10)          return "No entity could be linked"
(11)    else
(12)          return MaxLinkProb(ResultCandidate)

Procedure TransformContext(context, kb)
(1)     terms = LongestPossibleMostFrequentMatch(context, kb)
(2)     for each terms in terms:
(3)          entity = GetEntity(term)
(4)          CombineToCtxEntity(<olinks(entity), ilinks(entity)>)
(5)     return ctxEntity
```

Figure 9. Classification algorithm at run-time.

In Step (1) of ClassifyTerm procedure, we transform given context into an entity containing *out-link* set and *in-link* set, thus the link similarity measure could be applied. In TransformContext procedure, we first split the context into N-grams, and then do a longest possible match with the <Terms, Entity> pairs of ***kb*** computed in 3.2.1. For every N-gram there may be more than one matching <Term, Entity> pairs, we choose the one with highest frequency. Then we iterate through the matched terms (Step (2)), and then retrieve the corresponding entity (Step(3)), finally in Step (4) we make a union on the entity's link sets with the output, ctxEntity's link set, which is initialize as empty set.

We now return to the ClassifyTerm procedure. Once we get the transformed context entity, in Step (2) we retrieve the candidates <Term, Entity> pairs where "Term" equals the query term ***q***. For each entities in the candidate list, we compute feature vectors, where the first element is the link probability of current entity, the second and third elements are computed using eq. (1) with entity and context entity as input (Step (4)). After that we run the SVM model trained in 3.2.3 to predict the results, if it is positive, we add this entity to the result candidates list, otherwise we continue the iteration. After the end of the iteration, we select the one with highest link probability as the result entity to be linked (Steps (9)~(12)).

4 Experimental Setting

The proposed *Link Similarity Model* and knowledge base augmentation method was designed to resolve the sense ambiguity of given query terms and to leverage broader information from larger knowledge base. As such, our models will be trained on query terms and their target entities. In this thesis we treat hyperlinks and their destination in *Wikipedia* as query terms and target entities. Using such data, we compiled datasets from Chinese Wikipedia for training and evaluation. In this chapter, we first present the training and test data for the evaluation (Section 4.1). Then, Section 4.2 lists the methods we use in comparison. Section 4.3 introduces the evaluation metrics. Finally, we report the settings of the parameters in Section 4.4.

4.1 Data Set

In this thesis we focus on linking Chinese query terms to articles in Chinese *Wikipedia*. We used the Chinese *Wikipedia XML* file dumped at 20120503 as our main knowledge base. For the augmentation algorithm, we used 20120502 version of English Wikipedia to augment Chinese Wikipedia. Some statistics are shown in Table 3. Currently English *Wikipedia* is far more larger than Chinese *Wikipedia*, no matter in numbers of articles, numbers of language-links or average sense ambiguity. Notice that the sense ambiguity is lower in Chinese. To better investigate our algorithms, we compiled a collection of <hyperlink, article> pairs from Chinese *Wikipedia* with two criteria:

1. The sense ambiguity of hyperlink's anchor text (i.e. query terms) should not be too low or high. Lower ambiguity leads to easier datasets for our classifier, while extremely high value makes running time exponential longer, which is unacceptable for a real-time system. We set this value to lie in [2,7] in our experiment.

2. The contexts (i.e. articles) where each hyperlink appeared should not be too lengthy. Our *Link Similarity Model* uses hyperlinks information in context. In Wikipedia some special pages such as Lists pages, which lists instances of entities, contain extremely many hyperlinks that introduce too much noise to our model. In our implementation we make a threshold on number of hyperlinks per article to lower than 50.

Table 3. Statistics of Wikipedia

	Chinese Wikipedia	English Wikipedia
Number of articles	482,095	4,485,110
Percentage of language links	67%	9%
Average sense ambiguity	3.1	6.7

Using these criteria we randomly chosen 501 distinct <hyperlink, article> pairs from Chinese Wikipedia as our training data, and another distinct 2965 <hyperlink, article> pairs as testing data.

4.2 Methods Compared

The proposed method starts with a query term and its textual context, and determines a suitable entity (i.e. article) for the query term in Chinese *Wikipedia*. The output of our system is the linked article from Chinese *Wikipedia*.

In this thesis, we proposed a method for augmenting the smaller *Wikipedia*-like knowledge base (**CN**) using larger knowledge base (**EN**). In addition, we propose a model for computing link structure similarity between two hyperlinked articles, and then use it to train a *SVM* classifier, in which we use *out-links* (**OL**) and *in-links* (**IL**) as features. Further, the link probability (**LP**) is used as a feature to balance the system performance between rare and common entities. To inspect the effectiveness of the augmentation method and these modules in more detail, the baseline and the combinations of the three main modules, **OL**, **IL**, and **LP**, evaluated in our experiments are described as follows:

- **LP**: We train the *SVM* model using only link probability, and we use this model as baseline.
- **OL+IL+LP (CN)**: The full model trained using *out-links*, *in-links*, and link probability without augmentation.
- **OL+IL+LP (CN+EN)**: The most complete version of proposed system, using all features and augmentation process.
- **-LP (CN+EN)**: The full model with augmentation minus the link probability feature.
- **-OL (CN+EN)**: The full model with augmentation minus the *out-links* feature.
- **-IL (CN+EN):** The full model with augmentation minus the *in-links* feature.

4.3 Evaluation Results

In this section, we report the evaluation results of the experiments on the methodology described in the previous chapter. Table 4. shows the results evaluated on the testing data consist of 2965 <*query term*, *context*>.

Table 4. The evaluation results of different systems

System	*Classifier accuracy*	*Entity accuracy*
LP (Baseline)	95.87	90.54
OL+IL+LP(CN)	97.49	92.81
OL+IL+LP(CN+EN)	**97.61**	**93.02**
-LP (CN+EN)	90.38	71.38
-OL (CN+EN)	97.46	92.69
-IL (CN+EN)	95.94	88.81

As we can see, the full model (i.e. OL+IL+LP (CN+EN)) outperformed the strong baseline LP either on classifier accuracy or entity accuracy, which indicates that our

classification strategy can effectively return the most compatible entity to a given query term. As identified in previous related research (Milhacea et al., 2007; Milne et al., 2008), the baseline LP is extremely effective for determining suitable English *Wikipedia* articles for ambiguous query terms, in our experiment performed using Chinese *Wikipedia*, this is also the case.

Comparing the two full models (i.e. OL+IL+LP), the results on CN and CN+EN indicate that our augmentation process provides a small performance improvement. Although the augmentation process does not greatly improve the performance, we perform 10-fold cross validation on another test set consisting of 3001 <*hyperlink*, *article*> pairs and found that the performance gain is statistically significant.

In general, there is no significant difference between average number of *in-links* and *out-links*, so the number of links does not explain this phenomena. We suggest that in Wikipedia, *in-links* reflect topics that mention an entity, while *out-links* reflect context terms of a certain entity. Since topics are more stable than context term, the performance influenced by *in-links* are stronger.

In sum, our model achieved impressive performance for linking query terms to articles in Chinese *Wikipedia.* The augmentation process further significantly improve performance.

5 Conclusion and Future Works

Many avenues exist for future research and improvement of our system. For example, more features used in training the classification models could be added to boost system performance. To improve our system, language features such as collocations, *N*-gram counts, or part-of-speech could be added. Additionally, an interesting direction to explore is to apply our model to cross-language entity-linking. To support cross-language entity-linking, we could also augment the <*Term*, *Entity*> pairs described in 3.2.1 using similar augmentation process. Once the augmentation has been done, we could cross-link a term to other knowledge base. For example, "*蘋果*" in *Chinese Wikipedia* may be linked to "*Big Apple*", the nickname of New York city, in *English Wikipedia*.

In summary, we have introduced a method for linking a <*query term*, *context*> pair to an appropriate article in Chinese *Wikipedia*. Our goal is to improve user experience so that the underlying search system could distinguish between different search intents based on the context. The method involves possible candidates construction, knowledge base augmentation via inter-links, computation of various link similarity measures, and multi-class classification using binary *SVM* classifier. We have implemented and thoroughly evaluated the method as applied to linking query terms to Chinese *Wikipedia* articles. In our evaluation, we have shown

that the augmentation process slightly improved system performance. In addition, our full model significantly outperforms the strong baseline in terms of *entity accuracy*.

References

[1] Agirre, E., and Rigau, G. (1996). Word Sense Disambiguation using Conceptual Density. *16th Conference on Computational Linguistics*, (pp. 16-22). Copenhagen.

[2] Banerjee, S., and Pedersen, T. (2002). An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet. *the Third International Conference on Intelligent Text Processing and Computational Linguistics.* Mexico City.

[3] Black, E. W. (1988). An Experiment in Computational Discrimination of English Word Senses. *IBM Journal of Research and Development* , 185-194.

[4] Bruce, R., and Wiebe, J. (1994). Word-Sense Disambiguation Using Decomposable Models. *32nd Annual Meeting of the Association for Computational Linguistics* (pp. 139-146). Las Cruces: Association for Computational Linguistics.

[5] Carpaut, M., and Wu, D. (2007). Improving Statistical Machine Translation using Word Sense Disambiguation. *2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 61-72). Prague: Association for Computational Linguistics.

[6] Chan, Y. S., Ng, H. T., and Chiang, D. (2007). Word Sense Disambiguation Improves Statistical Machine Translation. *the Association for Computational Linguistics (ACL)*, (pp. 33-40).

[7] Chang, J. S., Lin, T., You, G.-N., Chuang, T. C., and Hsieh, C.-T. (2003). Building a Chinese WordNet via Class-based Translation Model. *Computational Linguistics and Chinese Language Processing* , 61-76.

[8] Chang CC and Lin CJ. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

[9] Cilibrasi RL and Vitanyi PMB. 2007. The google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on* 19(3):370-83.

[10] Diab, M., and Resnik, P. (2002). An Unsupervised Method for Word Sense Tagging using Parallel Corpora. *the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, (pp. 255-262). Philadelphia.

[11] Gale, W. A., Church, K. W., and Yarowsky, D. (1992). Using Bilingual Materials to Develop Word Sense Disambiguation Methods. *the International Conference on Theoretical and Methodological Issues in Machine Translation*, (pp. 101-112).

[12] Galley, M., and McKeown, K. (2003). ImprovingWord Sense Disambiguation in Lexical Chaining. *18th International Joint Conference on Artificial Intelligence (IJCAI 2003).* Acapulco.

[13] Hamp, B., and Feldweg, H. (1997). GermaNet - a Lexical-Semantic Net for German. ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications, (pp. 9-15). Madrid.

[14] Hearst, M. A. (1991). Noun Homograph Disambiguation using Local Context in Large Corpora. *7th Annual Conference of the University of Waterloo Centre for the New OED*

and Text Research, (pp. 1-15).

[15] Hsieh, C.-T. (2000). Semi-Automatic Construction of Chinese WordNet - Using Class-based Translation Model.

[16] Huang, C.-C., Tseng, C.-H., Kao, K. H., and Chang, J. S. (2008). A Thesaurus-based Semantic Classification of English Collocations. *ROCLING 2008*, (pp. 38-52). Taipei.

[17] Huang, C.-R., Chang, R.-Y., and Lee, H.-P. (2004). Sinica BOW (Bilingual Ontological Wordnet): Integration of Bilingual WordNet and SUMO. *4th International Conference on Language Resources and Evaluation (LREC2004)*, (pp. 1553-1556). Lisbon.

[18] Leacock, C., Towell, G., and Voorhees, E. (1993). Corpus-based Statistical Sense Resolution. *ARPA Human Language Technology Workshop*, (pp. 260-265).

[19] Lesk, M. (1986). Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. *5th Annual International Conference on Systems Documentation* (pp. 24-26). Toronto: Association for Computing Machinery.

[20] Longman Group. (1992). *Longman English-Chinese Dictionary of Contemporary English.* Hong Kong: Longman Group (Far East) Ltd.

[21] Mihalcea, R., and Moldovan, D. I. (1999). A Method for Word Sense Disambiguation of Unrestricted Text. *the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics* (pp. 152-158). College Park: Association for Computational Linguistics.

[22] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography* , pp. 235-244.

[23] Medelyan O., Witten I. H. and Milne D. 2008. Topic indexing with wikipedia. Proceedings of the AAAI WikiAI workshop, AAAI Press. 19 p.

[24] Mihalcea R. and Csomai A. 2007. Wikify!: Linking documents to encyclopedic knowledge. Proceedings of the sixteenth ACM conference on conference on information and knowledge management. 233 p.

[25] Milne D. 2007. Computing semantic relatedness using wikipedia link structure. Proceedings of the new zealand computer science research student conference.

[26] Milne D. and Witten I. H. 2008. Learning to link with wikipedia. Proceedings of the 17th ACM conference on information and knowledge management, ACM. 509 p.

[27] Pasca, M., and Harabagiu, S. M. (2001). The Informative Role of WordNet in Open-Domain Question Answering. *NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions, and Customizations*, (pp. 138-143). Pittsburgh.

[28] Towell, G., and Voorhees, E. M. (1998). Disambiguating Highly Ambiguous Words. *Computational Linguistics* , 125-145.

[29] Voorhees, E. M., and Tice, D. M. (1999). The TREC-8 Question Answering Track Evaluation. *TREC-8*, (pp. 84-106).

[30] Vossen, P. (1998). Introduction to EuroWordNet. *Computers and the Humanities* , 73-89.

[31] Wible, D., and Kuo, C.-H. (2001). A Syntax-Lexical Semantics Interface Analysis of Collocation Errors. *Pacific Second Language Research Forum.*