

Wikification via Link Co-occurrence

Zhiyuan Cai Kaiqi Zhao
Shanghai Jiao Tong University
Shanghai, China

{luckyvega,kaiqi_zhao}@sjtu.edu.cn

Kenny Q. Zhu
Shanghai Jiao Tong University
Shanghai, China

kzhu@cs.sjtu.edu.cn

Haixun Wang
Google Research
Mountainview, CA, USA

haixun@google.com

ABSTRACT

Wikification, which stands for the process of linking terms in a plain text document to Wikipedia articles which represent the correct meanings of the terms, can be thought of as a generalized Word Sense Disambiguation problem. It disambiguates multi-word expressions (MWEs) in addition to single words. Existing Wikification techniques either model the context of a given term as well as the Wikipedia article as bags of words, or compute global constraints among Wikipedia concepts by the link graph or link distributions. The first method doesn't achieve good results because the MWEs can have very different meanings than its constituent words which themselves are ambiguous. The second method doesn't produce high accuracy because the link structure or link distribution is often biased or incomplete by themselves due to the fact that Wikipedia pages are often sparsely linked. In this paper, we present a simple but powerful framework of sense disambiguation using co-occurrences of Wikipedia links in the Wikipedia corpus. We propose an iterative method to enrich the sparsely-linked articles by adding more links and then use the resulting link co-occurrence matrix to disambiguate an input document by a sliding window algorithm. Our prototype system achieves 89.97% precision and 76.43% recall on average for three benchmark data sets and compares favorably against all four state-of-the-art wikification techniques.¹

Keywords

Wikification, phrase sense disambiguation, link co-occurrence, iterative algorithm

1. INTRODUCTION

Natural language is rife with ambiguities. A word or a phrase usually has multiple meanings depending on the context of its use. This has been one of the most significant problems in automatic understanding and processing of human text. Consider the following sentences.

¹Kenny Q. Zhu (corresponding author) is partially supported by NSFC grants 61100050 and 61033002.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to publish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

EXAMPLE 1. *The polar bear is a bear native largely within the Arctic Circle encompassing the Arctic Ocean, its surrounding seas and surrounding land masses.*

EXAMPLE 2. *The original band, Polar Bear, was formed in 1994 by Gary Lightbody who was a student at Dundee University in Scotland.*

In Example 1, “polar bear” refers to a large white carnivorous bear inhabiting the arctic region; in Example 2, “Polar Bear” is the former name of a British rock band. The task of labeling a word or a multi-word expressions (MWE) (collectively called as a term) in a plain text by their explicit meanings is known as phrase sense disambiguation (PSD) problem [6]. PSD generalizes the more well-known open problem, word sense disambiguation (WSD) [21], which seeks to identify the meaning or the sense of the words. The difference is that in WSD, the disambiguated unit is word, not phrase.

PSD is an important generalization because the meanings of MWEs can be independent of the constituent words so traditional WSD techniques cannot be directly applied on PSD problems. For example, the word “masses” often means normal people, while the term “Dundee” usually reminds us of the hilarious Australian comedy from the 80's, and neither of these meanings have anything to do with “land masses” or “Dundee University” above. In fact, some researchers have already indicated that PSD outperforms WSD in NLP tasks such as statistical machine translation [6]. Moreover, the number of MWEs is massive. For example, if we consider Wikipedia [2], which is the largest online encyclopedia today with over 4 million concepts (articles), 62.8% of the concept terms (titles of the articles) consist of two or more words. A random sample of 10,000 web pages suggests that 66.2% of the phrases in them are ambiguous, because they can be mapped to two or more concepts in Wikipedia.

A particularly important form of PSD is known as wikification [19]. Wikification is an automatic process of linking possible MWEs in a document to an article in Wikipedia. This process disambiguates the terms by labeling each term with a proper sense (or concept)² which is represented by a Wikipedia article. Existing wikification techniques focus on noun-phrases because Wikipedia does not have articles to support all senses of verbs or verb phrases. For this reason, in the rest of the paper, when we mention “terms” or MWEs, we mean noun phrases.

Previous work on wikification often models a Wikipedia concept by its article using a bag of words, models a target term by its context which is also a bag of words, and then compare the two bags. The bag-of-words model doesn't work well (e.g. on Example 1 and

²In this paper, we use the terms “term”, “MWE” and “phrase” interchangeably. We also use the terms “sense” and “concept” interchangeably.

2) because: 1) many words themselves are ambiguous and don't offer distinctive signals, resulting in the convoluted signals from a bag. (e.g., "circle", "bear" and "masses" in Example 1 all have multiple senses, and the words "surrounding", "original" and "formed" don't contribute much to the meaning of the sentences). 2) it ignores other MWEs in the context or article by only treating them as individual words, leading to misunderstanding of some of the MWEs (e.g. "land masses" in Example 1 and "Dundee University" in Example 2).

An improvement over the above bag-of-words approach uses the relatedness between two concepts to constraint the labeling of two neighboring terms simultaneously [26, 22, 16]. The relatedness can be computed from the article inter-link graph structure induced from the corpus[12]. While the link structure is an important source to extract relation among concepts, the above approach misses out a more *direct* and *accurate* source of information, which is the *co-occurrence* between Wikipedia concepts in the corpus. Such co-occurrence can be computed by the co-occurrence of links within an article, a paragraph, or a predefined window, since the links are the natural labels of surface terms to the concepts.

In this paper, we propose a simple and novel approach on the wikification problem by using co-occurrence between Wikipedia links, and show that it is more effective and practical than the existing approaches.

Unfortunately, Wikipedia articles can be *sparsely linked*. Some surface terms don't have a corresponding Wikipedia page to link because it's not created yet; many others are not linked because either they are too common or they have been linked previously in the article before. Either way, the author of the article didn't consider them to be "link-worthy". Sparsely linked Wikipedia does not reveal the true link distribution or link graph structure, thus harms the accuracy of the link co-occurrence information. Previous research on WSD using word co-occurrence [24] already indicated that the correct distribution of a term's surrounding environment is critical to the end-to-end disambiguation accuracy. The same argument applies to wikification by concept co-occurrence.

To mitigate this problem, in this paper, we propose an iterative enrichment algorithm that adds the *missing links* to Wikipedia pages. In a nutshell, this algorithm maintains a concept co-occurrence matrix of the Wikipedia snapshot in the current iteration as partial information, and use it to disambiguate unlinked terms for the next iteration until no more links can be added. Figure 1 shows a snapshot of the links in a sentence of a Wikipedia article "before" and "after" the iteration process. We have experimentally verified (in Section 4) that our iterative enrichment algorithm increases the number of links by five times, and the number of co-occurrence by three times, and the final co-occurrence information gives rise to significant improvement (up to 6% increase) in the accuracy of end-to-end wikification.

There are two other technical challenges in our wikification framework. First, parsing of plain text into phrases can be ambiguous itself and common NLP tools are not accurate enough in this chunking process. Considering the sentence in Example 3:

EXAMPLE 3. *Snow Patrol are an alternative rock band formed at the University of Dundee in 1994, though at this time as an indie rock band, the band is now based in Glasgow, Scotland.*

The phrase "University" and "Dundee" are treated as independent chunks in the NLP chunker while "University of Dundee" is a correct parse.

Second, wikifying a new, unlinked document is a computationally intensive task because there's no support for any sense so all combinations of senses must be attempted. Suppose we have n

Modern portfolio theory(MPT) is a theory of [investment](#) which attempts to maximize portfolio expected [rate of return](#) for a given amount of portfolio risk, or equivalently minimize [financial risk](#) for a given level of expected return, by carefully choosing the proportions of various [assets](#).

[Modern portfolio theory](#)(MPT) is a [theory](#) of [investment](#) which attempts to maximize [portfolio](#) expected [rate of return](#) for a given [amount](#) of [portfolio risk](#), or equivalently minimize [financial risk](#) for a given [level](#) of [expected return](#), by carefully choosing the [proportions](#) of various [assets](#).

Modern portfolio theory	→	[Modern portfolio theory]		
MPT	→	[Modern portfolio theory]		
theory	→	[Theory]	portfolio	→ [Portfolio]
amount	→	[Quantity]	risk	→ [Risk]
level	→	[Level]	expected return	→ [Expected return]
proportion	→	[Proportionality]		

Figure 1: Snapshot of a Wikipedia Article "before" and "after" Iteration. (The source and destination of newly added links are indicated at the bottom)

The original band, Polar Bear, was formed in 1994 by Gary Lightbody who was a student at Dundee University in Scotland.

The original band, [Polar Bear](#)[[Snow Patrol](#)], was formed in 1994 by [Gary Lightbody](#)[[Gary Lightbody](#)] who was a [student](#)[[Student](#)] at [Dundee University](#)[[University of Dundee](#)] in [Scotland](#)[[Scotland](#)].

Figure 2: Wikification of Plain Text in Example 2 in the Web Demo. (Concepts are enclosed in square brackets)

phrases in a sentence, each has k senses in average, the number of combination is k^n ! Thus, we need a method that balances accuracy with complexity.

This paper makes the following contributions.

1. We propose an iterative algorithm that disambiguates unlinked MWEs in existing Wikipedia articles by linking them to other appropriate Wikipedia articles (concepts), and effectively enriches the link distribution;
2. We propose a sliding-window based method to wikify a given plain text document using the co-occurrence information obtained from enriched Wikipedia corpus;
3. We show comprehensive evaluation results that demonstrate the effectiveness of this novel approach with a web demo ³, as shown in Figure 2.

The rest of the paper is structured as follows. Section 2 presents the algorithms of this framework; Section 3 discusses some implementation details; Section 4 demonstrates the experimental results; Section 5 introduces some related work while Section 6 concludes the paper.

2. THE FRAMEWORK

The framework of wikification via link co-occurrence has a three-part architecture (See Figure 3): *preprocess of wiki data, co-occurrence matrix generation and document wikification*. The first part collects a term-sense mapping from Wikipedia data and parse Wikipedia articles to identify terms to be linked in the later parts. The second part is an iterative process which, in each iteration, computes the

³Wikification demo website: <http://202.120.38.145:4082/wvc/Guess.aspx>.

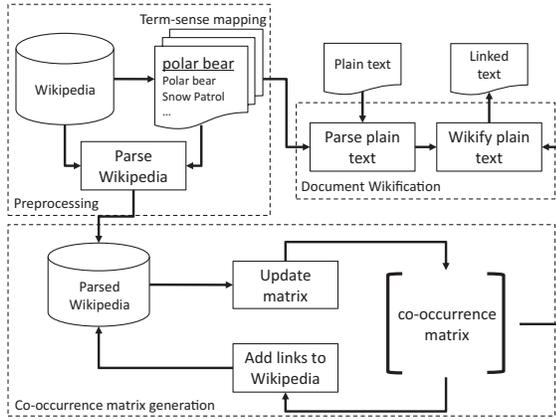


Figure 3: Architecture of Wikification via Link Co-occurrence

co-occurrence matrix from the current snapshot of the Wikipedia corpus, and then use this matrix to disambiguate (i.e., add links to) noun phrases that have not been linked yet in the corpus. The updated corpus will be used in the next iteration. This part can be thought of as an off-line process. The third part is an on-line process, which makes use of the final co-occurrence matrix and converts a plain text to a text with links pointing to relevant Wikipedia articles. Next, we describe each of the three parts in greater details.

2.1 Preprocessing

We start from the generation of the term-sense mapping from Wikipedia corpus. Then we introduce our parsing method on Wikipedia articles by making use of an NLP chunker.

2.1.1 Generate Term-Sense Mapping

To identify the sense of an unlinked term, we need to know the candidate senses of that term. Each term may map to more than one Wikipedia concepts which are the *senses*. The list of all Wikipedia concepts associated with an unlinked term is called the candidate sense list. In this paper, we use three sources in Wikipedia to build the term-sense mapping: Wikipedia article titles, redirect pages and disambiguation pages. Specifically, each Wikipedia concept (sense) is mapped to the title of this concept’s article, titles of redirect pages linked to this concept, and the title of disambiguation pages that contain this concept. In other works such as Cucerzan’s[8], anchor text of links in Wikipedia articles are also used as a source of mapping. Our observation is that surface forms in the anchor text and the links between the surface form and the Wikipedia article can both be noisy and unreliable. Anchor texts are not necessarily MWEs and the linked articles are sometimes not really the description of the anchor text but are rather just “related” information.

EXAMPLE 4. A further 6-7 million were [deported and exiled](#) [linkto: Population transfer in the Soviet Union] to remote areas of the USSR, and 4-5 million passed through “labour colonies”.

Example 4 shows a sentence from a Wikipedia article named “Gulag” [1]. The anchor text “deported and exiled” is not a term, and the concept “Population transfer in the Soviet Union” is not exactly a sense of “deported and exiled.” Due to this observation, we do not include links as a source for generating the term-sense mapping. Our candidate sense list for “polar bear” is, for example,

Polar bear, Polar Bear (American band), Snow Patrol, Polar Bear Pass, etc.

2.1.2 Parse Text into Noun Phrases

With the term-sense mapping generated, we can now parse texts to extract noun phrases. First, we take all the surface forms in the term-sense mapping to form a *term list*. Only those noun phrases in the term list will be considered later. In a Wikipedia article, we call terms which are already linked *linked terms*, while noun phrases which are waiting to be linked *unlinked terms*. Next we introduce the details of parsing.

Algorithm 1 Parsing a Chunk

```

1: function PARSECHUNK(Chunk)
2:   Unlinked  $\leftarrow \emptyset$ 
3:    $N \leftarrow \text{wordCount}(\text{Chunk}), k \leftarrow N$ 
4:   flag  $\leftarrow \text{False}$ 
5:   while  $k > 0$  and  $\text{!flag}$  do
6:     candidateTerm  $\leftarrow \text{Chunk}[N - k, N]$ 
7:     if candidateTerm is NP then
8:       if candidateTerm  $\in \text{TermList}$  then
9:         Add candidateTerm to Unlinked
10:       $L \leftarrow \text{ParseChunk}(\text{Chunk}[0, N - k - 1])$ 
11:       $\text{Unlinked} \leftarrow \text{Unlinked} \cup L$ 
12:      flag  $\leftarrow \text{True}$ 
13:       $k \leftarrow k - 1$ 
14:   return Unlinked

```

Parsing Wikipedia articles: The following is a short part of an Wikipedia article about a band.

EXAMPLE 5. *Snow Patrol* are an [alternative rock](#) band formed at the [University of Dundee](#) in 1994, though at this time as an [indie rock](#) band, the band is now based in [Glasgow, Scotland](#).

To get the noun phrases from the articles, we first treat the article as plain text, i.e. remove all the links. The plain text of Example 5 contains the following noun phrases: “Snow Patrol”, “alternative rock band”, “University of Dundee”, “time”, “indie rock band”, “the band”, “Glasgow”, and “Scotland”. But not all of them can be found in Wikipedia. For example, you cannot find concepts “alternative rock band” or “indie rock band” in Wikipedia as of this writing. So instead, our candidates are those terms from the term list, e.g., “Snow Patrol”, “alternative rock”, “band”, “University of Dundee”, “time”, “indie rock”, “Glasgow” and “Scotland”.

We achieve the parsing task in two steps. In step 1, we parse the text into linguistic chunks to obtain noun phrases using an NLP chunker. The chunker can detect phrases from a sentence, including verb phrases, noun phrases, prepositional phrases and adverb phrases. In our framework, we only pick the noun phrases from the chunker. Notice that chunkers are not always correct, therefore we introduce a method to optimize our chunking result at the end of this section. In step 2, we detect unlinked terms from the resulting noun phrase chunks. To simplify the unlinked term detection, we adopt a simple strategy: remove words one by one from left to right, while the remaining part is a noun phrase and an unlinked term. The intuition here is that longer terms are more likely to be accurate. That is, we prefer to use “alternative rock” as unlinked term rather than “rock”. The details of the parsing strategy is shown in Algorithm 1. *wordCount* is a function for counting the number of words in the noun phrase chunk. *TermList* is the list of all terms in the term-sense mapping. *Chunk*[*i*, *j*] is the sub-string from word *i* to word *j*.

With the chunks, we then check if the chunks fit the original Wikipedia article which has linked terms. These linked terms may not align properly with the chunking results from the chunker and therefore cause conflicts. Below is the original text from Wikipedia along with the chunking results.

EXAMPLE 6. [Snow Patrol] are [an [alternative rock band](#)] formed at [the [University of Dundee](#)] in [1994], though at [this time] as [an [indie rock band](#)], [the band] is now based in [[Glasgow](#)], [[Scotland](#)].

The terms with underlines are linked terms, and phrases enclosed in square brackets are the chunks produced by a chunker. The three conflicts are listed as follows:

- [an [alternative rock band](#)]
- [an [indie rock band](#)]
- [the [University of Dundee](#)]

Our conflict resolution policy is that the original links in the Wikipedia article are always respected. In that words, links are natural chunks. Where there’s conflict, we break up an offending chunk produced by chunker into smaller chunks. For example the above segments can be re-chunked as:

- [an][[alternative rock](#)][band]
- [an][[indie rock](#)][band]
- [the][[University of Dundee](#)]

Optimization on chunks: As we mentioned earlier, the chunker can make mistakes. When a chunk is too wide, i.e., the correct term is properly contained in the chunk, Algorithm 1 can be applied to extract the correct term. When a chunk is too narrow, i.e., it is only a part of a term, we need a way to merge adjacent chunks together to form a term. For example, “the University of Dundee” was incorrectly chunked into “the University”, “of” and “Dundee” in Example 6. Had there been no hyperlink on “University of Dundee”, we need a way to reconstruct the term automatically.

We use the following regular expression pattern to capture the potential incorrect segmentations of a noun phrase:

$$(NP(PP|CC)?) + NP$$

where NP stands for noun phrase, PP for preposition and CC for conjunction. In this pattern, we allow prepositions and conjunctions to appear in the compound noun phrases. If the pattern matches an unlinked term, we combine phrases in the pattern to form a new chunk. “the University”, “of”, “Dundee” are thus combined to “the University of Dundee”.

Combining the optimization with the strategy described in parsing Wikipedia articles, and applying Algorithm 1, we can produce a more refined chunking for Example 6:

- [Snow Patrol] are an [[alternative rock](#)] [band] formed at the [[University of Dundee](#)] in [1994], though at this [time] as an [[indie rock](#)] [band], the [band] is now based in [[Glasgow](#)], [[Scotland](#)].

We map the unlinked terms in the parsing result to the corresponding candidate sense lists from the term-sense mapping to construct the final result of our parsing process which is used to generate the co-occurrence matrix next.

2.2 Enrich Co-occurrence Matrix and Articles

Co-occurrence among Wikipedia concepts provides the knowledge for disambiguating terms in a given document. The co-occurrence information of Wikipedia concepts is in theory a $K \times K$ square matrix where K is the total number of concepts in Wikipedia. Each element in the matrix represents the total co-occurrence frequency of the two concepts in any Wikipedia articles. Despite the large number of concepts that exist in Wikipedia, not every pair of them co-occur, and therefore in practice, the matrix is very sparse and manageable. Next, we present the algorithms that compute the co-occurrence matrix, which involves two phases: *matrix initialization* and *matrix enrichment*.

2.2.1 Matrix Initialization

In the initialization phase, we take as input the parsed Wikipedia articles which include both linked terms and unlinked terms, to calculate the co-occurrence frequency between the concepts of two linked term and the appearance frequency of each concept (i.e., the sum of all co-occurrence frequencies for that concept). We argue that computing the co-occurrence within the whole article is not only computationally demanding, but also counter-productive. Wikipedia articles are often much longer and richer than traditional dictionary definitions. Multiple topics may co-exist within the same article. As a result, co-occurrence of two concepts which are very far away from each other in the article might not be related at all! Therefore we only consider two concepts co-occur if they are less than W_c terms (either linked or unlinked) apart in the text. In addition, we consider a concept c to co-occur with all the concepts in its own definition page, considering that all the other concepts in the page contribute to the description of c . This actually incorporates the link structure in Wikipedia into the co-occurrence framework.

To illustrate the initialization process, let’s consider the parsing result of Example 6 again, with W_c equal to three. The first linked term is “alternative rock”. The only linked term within the window is “University of Dundee” which is 2 terms away. So the concept of these two linked terms are considered to co-occur. We then move on to the next linked term which is “University of Dundee”. This time the only linked term within the window is “alternative rock”. The process continues till the last linked term.

2.2.2 Matrix Enrichment

The initial co-occurrence matrix doesn’t have enough information because Wikipedia articles are often sparsely linked. We develop the algorithm that bootstraps from the initial matrix and iteratively adds links to the current Wikipedia articles and updates the matrix concurrently. The pseudo-code of this algorithm is shown in Algorithm 2 which consists of two procedures UPDATEARTICLES and UPDATEMATRIX.

In Algorithm 2, S_u stands for the candidate sense list of an unlinked term t_u . The list of its linked neighbors’ senses are denoted as S_l . A neighbor here means a term that is fewer than W_c terms away. The co-occurrence frequency of two concepts (or senses) is denoted as $Co(c_i, c_j)$. The appearance frequency of a concept c is denoted as $Tf(c)$, which is equal to the times c occurs in Wikipedia corpus. UT is list containing all the updated (disambiguated) terms in the UPDATEARTICLES process. *Conditional concept score* (S_{CC}) is a score used to determine the sense of t_u based on conditional probability. Given $c_i \in S_l$, the conditional probability that a concept $c \in S_u$ is selected as the correct sense is defined as $P(c|c_1, c_2, \dots, c_n)$. According to Bayes’ theorem and

Algorithm 2 Enrich Co-occurrence Matrix

```

1: procedure ENRICHMATRIX
2:   InitMatrix()
3:   while UpdateArticles() > 0 do
4:     UpdateMatrix()

5: function UPDATEARTICLES
6:   updatedCount ← 0
7:   UT ← ∅
8:   for a in Wikipedia Corpus do
9:     for tu in a do
10:      Initialize Score[ sizeof(Su) ]
11:      for i ← 0, sizeof(Su) - 1 do
12:        Score[i] ← SCC(Su[i])
13:      Sort Score Descending
14:      if Score[1]/Score[0] ≤ τ then
15:        Assign Su[0] to tu
16:        updatedCount ← updatedCount + 1
17:        UT ← UT ∪ tu
18:   return updatedCount

19: procedure UPDATEMATRIX
20:   for tu in UT do
21:     for ci in Si do
22:       cu ← Concept of tu
23:       Update Co(cu, ci)
24:       Update Tf(cu)
  
```

assume independence of c_1, \dots, c_n , we have

$$P(c|c_1, c_2 \dots c_n) \propto P(c_1, c_2 \dots c_n|c) P(c) = P(c) \prod_{i=1}^n P(c_i|c) \quad (1)$$

In Equation (1), we can replace $P(c_i|c)$ with $\frac{Co(c, c_i)}{Tf(c)}$ and $P(c)$ is proportional to $Tf(c)$, therefore S_{CC} of a candidate concept c in S_u can be defined as:

$$S_{CC}(c) = Tf(c) \prod_{c_i \text{ in } S_i} \frac{Co(c, c_i)}{Tf(c)} \quad (2)$$

To prevent S_{CC} from being 0 when $Co(C, c_i)$ is 0, we add a smoothing term to Equation (2) and obtain:

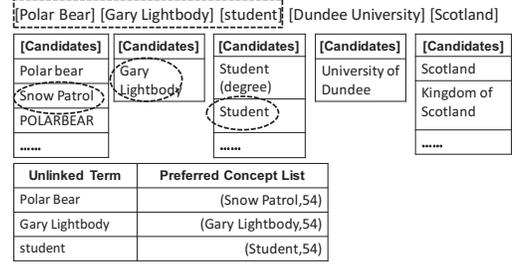
$$S_{CC}(c) = Tf(c) \prod_{c_i \text{ in } S_i} \frac{Co(c, c_i) + \frac{1}{sizeof(S_u)}}{Tf(c) + 1} \quad (3)$$

UPDATEARTICLES processes all unlinked terms in all articles and attempts to disambiguate and convert unlinked terms to linked terms. For each candidate concept of an unlinked term, we calculate S_{CC} of it then sort the concepts in the candidate list by this score. If there is only one concept in the candidate list and the score is non-zero, the term is disambiguated and linked to this concept. If there are more concepts in the candidate list and the ratio between the scores of the top two concepts is less than a threshold τ , the number one concept will be chosen to disambiguate the term.

In Example 6, the first “band” is an unlinked term. Term “band” has many candidate concepts such as *Belt (clothing)*, *Band (radio)*, *Band society*, and *Musical ensemble*. With window size equal to three, linked terms inside the window are “alternative rock” and “University of Dundee”, whose concepts are *Alternative rock* and *University of Dundee*. According to S_{CC} , *Musical ensemble* is

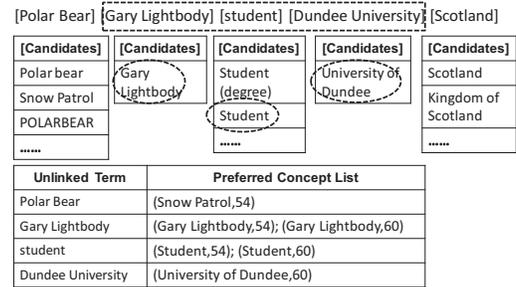
ranked number one, while *Band (radio)* is the number two concept. Since the ratio between *Band (radio)* and *Musical ensemble* is less than 0.5 (a τ value determined empirically in Section 3), *Musical ensemble* is picked to be the correct concept of “band” and “band” is inserted into UT .

In UPDATEMATRIX, for each term in UT , we update the co-occurrence matrix by the co-occurrence between this term’s concept and its neighboring linked terms’ concepts within the window of size W_c . This is very similar to *Matrix Initialization*. Once the co-occurrence matrix is updated, there’s more knowledge that enables the disambiguation of other unlinked terms in the next iteration. The iterative process continues until no more linked term can be linked and the final co-occurrence matrix is stored.



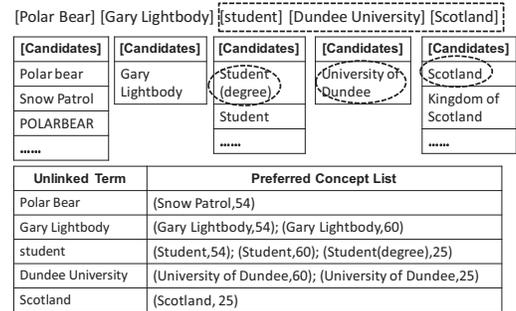
Unlinked Term	Preferred Concept List
Polar Bear	(Snow Patrol,54)
Gary Lightbody	(Gary Lightbody,54)
student	(Student,54)

(a) Step 1



Unlinked Term	Preferred Concept List
Polar Bear	(Snow Patrol,54)
Gary Lightbody	(Gary Lightbody,54); (Gary Lightbody,60)
student	(Student,54); (Student,60)
Dundee University	(University of Dundee,60)

(b) Step 2



Unlinked Term	Preferred Concept List
Polar Bear	(Snow Patrol,54)
Gary Lightbody	(Gary Lightbody,54); (Gary Lightbody,60)
student	(Student,54); (Student,60); (Student,25)
Dundee University	(University of Dundee,60); (University of Dundee,25)
Scotland	(Scotland,25)

(c) Step 3

Figure 4: Sliding Window Example

2.3 Wikify New Documents

To wikify an entire document, our general idea is to disambiguate several unlinked terms together within a local context by optimizing the likelihood of co-occurrence among a particular combination of senses for these terms. We could use the whole document as the context but that will be computationally infeasible if the document is large. Instead we make the size of the context parameterized and tunable. However, a local context can be misleading and can

introduce errors in the disambiguation. We solve this problem by creating a sliding window that allows us to aggregate the disambiguation results from neighboring contexts together and then make a pseudo-global decision about which sense each term ultimately should have.

We illustrate this idea using Example 2 and show three steps in sliding a window of size three in Figure 4. We parse the plain text using the NLP chunker discussed in Algorithm 1 and get the following terms: “Polar Bear”, “Gary Lightbody”, “student”, “Dundee University” and “Scotland”. We create a candidate sense list for each term. Step 1 shows a window containing “Polar Bear”, “Gary Lightbody” and “student”. Given that each term has a few senses as candidates, there are many combinations of senses for these three terms, such as:

- Polar bear, Gary Lightbody, Student (degree)
- Snow Patrol, Gary Lightbody, Student
- POLARBEAR, Gary Lightbody, Student (degree)

Snow Patrol, *Gary Lightbody* and *Student* turn out to be the best sense combination since the sum of the *pair-wise* co-occurrence is the largest at 54. Note here that we are simulating the overall co-occurrence with pair-wise co-occurrence, which is a reasonable approximation under computation constraints. We call the maximum sum *sliding window score* (S_{SW}), and every term within this window is associated with this disambiguation result (in the form of sense- S_{SW} pair) in a data structure called *Preferred Concept List* (PCL).

In Step 2, the window is slid to the right by one term, and the best combination of senses is again computed while the PCL is updated with three more entries. This time S_{SW} is 60.

In Step 3, within the last window of this sentence. The best combination contains the *Student (degree)* sense, which is different from the result of “student” in the 2 previous steps.

After we have finished sliding the window from start to finish, we group the results for each term by senses and sum the S_{SW} by the group. In the case of “student”, *Student* sense has a combined S_{SW} of 114 while *Student (degree)* sense has a combined S_{SW} of 25. The first sense wins and becomes the final sense for the term “student”. Other terms are similarly disambiguated.

The detailed sliding window wikification algorithm is given in Algorithm 3. L_u stands for the list of unlinked terms. W_s stands for the size of the sliding window. By picking one concept for each unlinked terms inside the sliding window, we get a *Concept Combination* (CC). Picking different candidate concepts from unlinked terms gives different CC s, thus forms the *Concept Combination List* (CCL). Therefore, S_{SW} can be defined as

$$S_{SW} = \max_{CC \in CCL} \sum_{c_i, c_j \in CC; i \neq j} Co(c_i, c_j)$$

The CC associated with S_{SW} is called *Best Concept Combination* (BCC). $PCL_{L_u[i]}$ denotes the PCL of the i th term in L_u .

3. IMPLEMENTATION

In this section we first attempt to experimentally determine the value for three key parameters in our framework and then describe a baseline system for comparison with our framework.

3.1 Parameter Settings

The key parameters in our framework include τ which determines whether to disambiguate a given term in our matrix enrichment process, W_c , the co-occurrence window size in the iterations, and W_s , the sliding window size for wikifying new documents.

Algorithm 3 Sliding Window Method

```

1: procedure SLIDINGWINDOW( $L_u, W_s$ )
2:   for  $i \leftarrow 0, \text{sizeof}(L_u) - W_s$  do
3:      $CCL \leftarrow \text{getCombination}(L_u[i], W_s)$ 
4:     Calculate  $S_{SW}$  and  $BCC$ 
5:     for  $j \leftarrow 0, \text{sizeof}(BCC) - 1$  do
6:       Add  $\langle BCC[j], S_{SW} \rangle$  to  $PCL_{L_u[i+j]}$ 
7:   for  $i \leftarrow 0, \text{sizeof}(L_u) - 1$  do
8:     Initialize  $\text{hash}[\text{concept} \rightarrow \text{score}]$ 
9:     for  $j \leftarrow 0, \text{sizeof}(PCL_{L_u[i]}) - 1$  do
10:       $\langle \text{key}, \text{value} \rangle \leftarrow PCL_{L_u[i]}[j]$ 
11:       $\text{hash}[\text{key}] \leftarrow \text{hash}[\text{key}] + \text{value}$ 
12:       $\text{concept} \leftarrow \text{argmax}_{\text{key}}(\text{hash}[\text{key}])$ 
13:      if  $\text{hash}[\text{concept}] > 0$  then
14:        Assign  $\text{concept}$  to  $L_u[i]$ 

```

For threshold τ , we randomly pick 100 paragraphs from Wikipedia corpus. We use function *UpdateArticles* in Algorithm 2 to add links to these paragraphs using the matrixes generated by the enrichment process on different thresholds. We also manually add links to these 100 paragraphs, as ground truth labels. We compare the different linking results with the ground truth then calculate the precision and recall, which is shown in Table 1.

τ	Precision	Recall	τ	Precision	Recall
0.1	87.50%	40.94%	0.5	90.16%	64.50%
0.2	90.04%	63.66%	0.75	89.62%	66.01%
0.25	89.29%	63.74%	0.875	88.89%	68.57%

Table 1: Result on Different Thresholds (with co-occurrence window $W_c = 15$)

We can see that threshold 0.5 achieves the best precision and also reasonable recall. Since our matrix enrichment process is an iterative process, precision in each iteration is more important, we therefore choose 0.5 as threshold τ .

For W_c , we follow the same experiment described above, since these two parameters are both used in the matrix generation part. Instead of using different thresholds, we change the window size this time. Table 2 shows the linking precision and recall using different W_c . 5 and 15 both achieve precision higher than 90%. To optimize the recall, we set $W_c = 15$.

W_c	Precision	Recall	W_s	Precision	Recall
5	91.67%	45.20%	2	84.59%	52.74%
10	89.63%	60.84%	3	84.95%	83.47%
15	90.16%	64.50%	4	85.52%	88.68%
20	88.30%	67.05%	5	85.55%	89.60%

Table 2: Result on Different W_c and W_s (with threshold $\tau = 0.5$)

For W_s , we build another test data with 100 randomly picked paragraphs from web and wikify them using the sliding window algorithm. The matrix used in this experiment is generated by enriching 10,000 sample Wikipedia articles, with parameters $\tau = 0.5$ and $W_c = 15$. Table 2 compares the results on different W_s with the manually created ground truth. Both precision and recall increase with growing W_s . When W_s is larger than 5, the whole process takes too much time and is thus not practical. Consequently we set $W_s = 5$.

3.2 Baseline System

Besides the algorithm introduced in Section 2, for comparison purpose, we also implemented a baseline system which wikifies a document by the co-occurrence between Wikipedia concepts and plain words. This system can be thought of as a direct port of WSD from using WordNet to using Wikipedia, and it also uses a common bag-of-words approach. In this baseline system, the co-occurrence vector of each Wikipedia concept is constructed from words and frequencies in the article of this concept itself. With the vectors of all Wikipedia concepts, we can wikify a document by comparing the co-occurrence vectors with the context of each term in the document. Given a document, we parse it into terms in the same way as our wikification framework. Each term has a list of candidate Wikipedia concepts. We compute the cosine similarity between the vector of each candidate concept and the vector built from the input document. The concept whose vector has the best similarity with the document vector is chosen to disambiguate that term. We compare the result of this baseline system with our wikification framework in Section 4.

4. EVALUATION

In this section we evaluate the framework of wikification via link co-occurrence. We first explain how we prepare Wikipedia corpus samples and the test data. Then we discuss four experiments. The first experiment shows the effects of using corpora of different sizes; the second one evaluates the iterative algorithm which generates the co-occurrence matrix; the third one compares the end-to-end wikification results from our framework with the baseline as well as four state-of-the-art methods; while the last experiment measures the time performance of our system. The Wikipedia corpus we use was a complete dump [2] (27GB) published in May 2011 which contains 8.56 million surface forms and 3.28 million unique concepts (senses). All experiments in this section are conducted on a 64-bit workstation with 3.10 GHz quad-core Intel CPU and 16 GB memory running Windows Server 2008.

4.1 Data Preparation

One of the important points to show in the evaluation is that our framework doesn't require all of the Wikipedia articles to be effective. But instead, samples of highly popular and information-rich articles are enough to provide the co-occurrences we need for wikification. Completely random samples don't guarantee information richness. Our approach is to first collect a set of information-rich articles, then apply random sampling on the set. We obtain the set of information-rich articles by taking the top $3k$ articles ordered by *PageRank*[5] within the Wikipedia network. *PageRank* ranks documents by their popularity. We argue that articles which are referenced by many others are supposed to have richer content.

To compute *PageRank* on the Wikipedia network, each Wikipedia article is treated as a web page, and links to the peers are treated as hyperlinks in web pages. As a bias towards longer articles, we set the initial *PageRank* to be proportional to the article length. Finally from the $3k$ informative articles, we randomly pick k articles as our sample corpus.

Next we describe our test data sets. Though there has been previous work on wikification, no standard test data sets are available yet. Bartunov et al.[3] ran a project called "WikifyMe" to create a standard test data for wikification. Unfortunately, the web site doesn't provide sufficient data for our test purposes. In our evaluation, we use three test data sets. The first two are publicly available data sets for wikification, reported by Dai et al.[9]. They are created by Cucerzan[8] and Kulkarni et al.[16], respectively. Cucerzan's test data comes from Wikipedia articles and news articles. Because

Home Depot[The Home Depot] CEO Nardelli[Robert Nardelli] quits.Home[The Home Depot] - improvement retailer's chief executive had been criticized over pay.ATLANTA - Bob Nardelli[Robert Nardelli] said he had no intention of stepping down. ... (omitted)"
On the Trail Nancy Pelosi[Nancy Pelosi]: End the war for our kids' sake In this essay, the Speaker of the House[Speaker (politics)] and mother of five calls for peace[Peace] Mark Wilson / Getty Images[Getty Images] file ... (omitted)"
US Navy[United States Navy]'s Operation Chief Jonathan Greenert[Jonathan W. Greenert] on Wednesday[Wednesday] said the Iranian Navy[Islamic Republic of Iran Navy] has been "relatively quiet" in their dealing with US[United States] ... (omitted)"

Figure 5: Snippets from The Three Test Data Set

Wikipedia articles may have been modified since Cucerzan finished his experiment, we only use the news articles. Kulkarni's data set is complete with test corpus and labeled ground truth. The third test set is our own creation which is extracted from 25 articles of New York Times and China Daily covering five topics: world, business, sports, entertainment and technology. We parsed these paragraphs into unlinked terms and then manually linked these terms to the appropriate Wikipedia articles as the ground truth labels.

In the rest of this section, the three test sets are referred to as Cucerzan's, Kulkarni's and Cai's, respectively. Examples from each of the three test set (along with our wikification results) are shown in Figure 5. The phrases disambiguated are marked with underlines, with the disambiguated senses surrounded by square brackets.

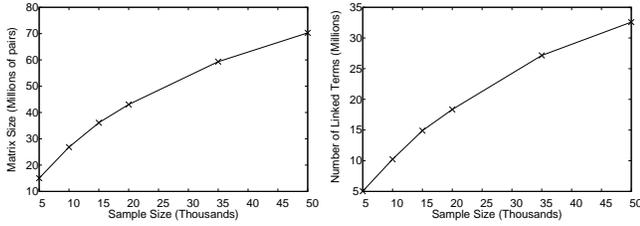
4.2 Effects of Wikipedia Corpus Sizes

In the first part of the evaluation, we would like to check the effect of using different corpus size. We run our experiment on five different samples. Figure 6a and Figure 6b show the final number of pairs of co-occurring concepts and linked terms after iteration on different samples. We can see that both numbers are sub-linear to the corpus sizes, indicating that increasing the sample size which increases the cost in time and space doesn't give us proportional gain. Our hypothesis is that, with a proper sampling strategy, we can obtain sufficient knowledge in a small corpus. To this end, we introduce *matrix coverage* as a measure to evaluate the quality of sampling. Matrix coverage measures the number of concepts appearing in the matrix. It's important because we can possibly disambiguate a term into a concept only if the concept exists in our matrix. Considering the fact that different concepts differ in their popularity and importance, we also calculate a weighted matrix coverage, i.e., we multiply the *PageRank* score by the frequency of a concept. Figure 7a and Figure 7b show the two versions of matrix coverage. The values are sub-linear to sample sizes, which supports our hypothesis.

To get a straightforward view of the effect on wikification using different corpus sizes, we conduct the next experiment. We take the final matrix produced by different corpus size with $W_c = 15$ and $\tau = 0.5$ to wikify the three test data set. Figure 9 shows the average precision, recall and F1-measure of our wikification method on them. Result in different corpus sizes are similar to each other, indicating that a small number of popular articles is good enough to bootstrap the matrix generation process, leading to a good wikification result.

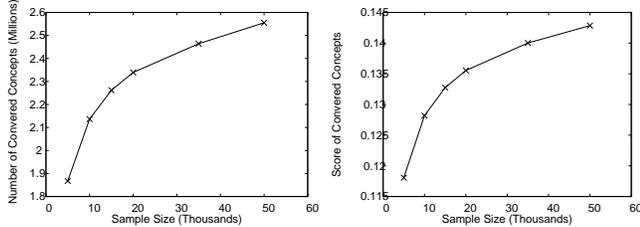
4.3 Iteration Results

In the second part of the evaluation, we verify the correctness of the new links added during the iterative enrichment algorithm.



(a) Final Matrix Size on Different Corpus Sizes (b) Final Number of Linked Terms on Different Corpus Sizes

Figure 6: Final Matrix Size and Number of Linked Terms



(a) Concepts Covered in the Matrix on Different Corpus Sizes (b) Score of Covered Concepts on Different Corpus Sizes

Figure 7: Final Matrix Coverage

The accuracy of the additional links is important because it affects the distribution of the final matrix, thus affects the end-to-end wikification result. We sample 20,000 articles from Wikipedia corpus and run our iterative process to adding links to Wikipedia articles. When the iteration completes, we manually check whether the added links are correct or not. The number of added links and the accuracy in each iteration is shown in Figure 8. We sample 100 articles from those 20,000 articles. For each article, we pick the first 50 added links in each iteration to form the data set, forming about 5,000 links to be examined in each iteration. From Figure 8 we can see that, our iterative process significantly increases the links in Wikipedia corpus while the accuracy in each iteration decreases slightly but stabilizes above 0.9.

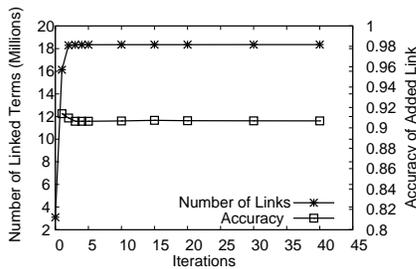


Figure 8: Number and Accuracy of New Links in Wikipedia

Next is to test the effectiveness of our iterative process in improving the end-to-end wikification result. We use the original matrix and final matrix to wikify the text in the three test data sets. Sample size here is 10,000. Table 3 shows the result of our wikification method on them, before and after the iterations. In each cell, the three numbers from top to bottom are precision, recall and F1-measure, respectively. We can see that, after iterations, which provides us more concept co-occurrence information by adding accurate links to Wikipedia corpus, the matrix achieves better precision and F1-measure in all the three test data set.

Data Set		Cucerzan's	Kulkarni's	Cai's
Original Matrix	P	83.57%	85.11%	85.75%
	R	71.06%	68.21%	76.79%
	F	76.81%	75.73%	81.02%
Matrix after Iterations	P	85.41%	89.24%	95.25%
	R	71.26%	71.05%	86.99%
	F	77.69%	79.11%	90.93%

Table 3: Result Before and After Enrichment (P/R/F)

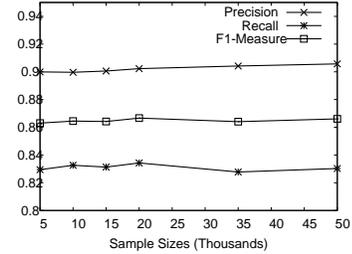


Figure 9: Results on Different Data Set (P/R/F)

4.4 End-to-End Wikification Results

In the third part of the evaluation, we compare our result with 4 state-of-the-art methods, namely, Cucerzan's[8], Kulkarni's[16], WikiMachine[26] and Ratinov[22] as well as the baseline algorithm in Table 4. Our system (Co-occur.) outperforms all the competing methods on F1-measure and has better precisions on Kulkarni's and Cai's data.

Method		Data Set		
		Cucerzan's	Kulkarni's	Cai's
Cucerzan	P	69.40%	54.84%	35.96%
	R	57.49%	45.79%	33.73%
	F	62.88%	49.91%	34.81%
Kulkarni	P	63.75%	63.76%	44.35%
	R	49.08%	62.43%	43.91%
	F	55.46%	63.09%	44.13%
WikiMachine	P	78.39%	78.97%	70.17%
	R	52.30%	57.41%	59.88%
	F	62.74%	66.48%	64.62%
Ratinov	P	87.15%	84.50%	90.12%
	R	53.07%	34.41%	29.74%
	F	65.97%	48.91%	52.59%
Baseline	P	72.02%	66.14%	52.59%
	R	62.67%	63.66%	51.79%
	F	67.02%	64.88%	52.19%
Co-occur.	P	85.41%	89.24%	95.25%
	R	71.26%	71.05%	86.99%
	F	77.69%	79.11%	90.93%

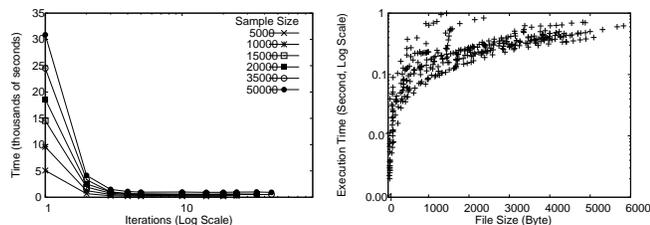
Table 4: Overall Comparison Against Peers (P/R/F)

4.5 Performance

Finally, we measure the system performance of both the matrix generation process and the end-to-end wikification process. The execution times of each iteration with different sample sizes are shown in Figure 10a. Figure 10a shows that the iterative method converges quickly after a few iterations, indicating that it is possible to use a stopping criterion to terminate the process after a few iterations and yet produce a reasonable co-occurrence matrix.

We also collect data to see how those two parameters, threshold τ and co-occurrence window size W_c , affect the total number of iterations in matrix generation process. It's interesting that τ doesn't affect the number of iterations much but W_c does. Our explanation is that in a small window, an unlinked term has fewer linked terms to help disambiguate it. It is possible that there is even no linked term in a small window. Thus, the links added in each iteration is fewer when using small W_c , resulting in more iterations. It's true that τ also affects the speed of adding link, but our experiment shows that it's not as obvious as W_c .

To evaluate the online wikification performance, we use the 20 articles in Cucerzan's test data set. We produce article segments of various sizes by chopping the articles in the data set into paragraphs before merging the consecutive ones together. The correlation between file size and time cost is showed in Figure 10b. Note that the y-axis is on logarithmic scale so the scatter plot clearly indicates that our wikification time is roughly linear to the input document size. In addition, all of the articles, including the longer ones with over 1000 words, can be effectively wikified under 1 second.



(a) Execution Time of Each Iteration (b) Wikification Time of Different File Sizes

Figure 10: Performance

5. RELATED WORK

In the following, we first present the state of the affairs in WSD research, and then put our work into perspective by discussing various approaches in wikification before briefly introducing several additional pieces of work related to wikification.

5.1 Traditional WSD

Traditional methods for WSD are either dictionary-based or machine learning methods. Co-occurrence information between words is also used in some WSD work [13, 17, 7, 24, 10, 27]. Among them, Guthrie [13], Fernandez-Amoros [10] and Véronis [27] introduced unsupervised methods. Guthrie [13] proposed a two-level WSD on subject (domain) level and sense level (within a subject). In each disambiguation level, they chose the subject/sense with the highest similarity between the context and the description of the subject/sense in a dictionary. Fernandez-Amoros [10] used co-occurrence to compute a relative matrix using mutual information (MI) measure. They used the MI between the target word and words of its context as the weight of each context words, and select the sense whose WordNet definition is most similar to the context by the bag-of-words model. Véronis [27] clustered words into hubs on a co-occurrence graph mined from a large corpus. The hubs that contain a target word define its different senses. The disambiguation of a word is done by computing the similarity between the context of this word and its various hubs by bag-of-words again. All of the above methods disambiguate *words* only whereas in this paper, we proposed a method for the more general problem of *phrase* sense disambiguation, and instead of relying on bags-of-words approach, our method takes advantage of the link co-occurrences of Wikipedia concepts.

5.2 Wikification Algorithms

There are generally two broad approaches to the wikification problem [22]: *local algorithms* which labels the terms in a document one by one using the local context of each term only; and the *global algorithms* which use global information of the sense configuration in the whole sentence to improve the previous local methods. Next we discuss these two approaches separately.

Most local algorithms use bag-of-words similarity between the context of the target term and the context of each candidate sense to identify the correct sense. Cucerzan [8] applied vector space model on linking *named entities* to Wikipedia concepts. Each Wikipedia concept is modeled as a vector. The concept with the most similar vector with the document vector is chosen to link a term. Since document vector is constructed by merging the vectors of all terms' candidate concepts, important signals may be diluted and weakened. Furthermore, this work only focuses on disambiguation of named entities. Ferragina and Scaella [12] applied similar framework, but instead of calculating vector similarity, they calculated a relatedness score between concepts. Since this disambiguation process is very time-consuming for an online system, they restricted the input documents to be short texts. Milne and Witten applied machine learning methods on wikification [20]. Using part of the Wikipedia corpus as the training set, they combined three features together, which were commonness, relatedness and quality of text, to train a classifier that can distinguish correct concept from irrelevant ones for a given term. Fernando and Stevenson limited their wikification work on cultural heritage [11]. Their system is based on Milne and Witten's work, but uses category information in Wikipedia to filter irrelevant concepts both in the training set and the result. They also used the link structure in Wikipedia to help find more proper links. Skaggs [23] and Boston [4] proposed topic models for wikification. Skaggs used an LDA-based topic model while Boston applied a simple generative model which selects the sense with the highest conditional probability given the context of the term.

Several recent attempts were made to combine both local and global information in wikification. Global information is the relation or constraints between Wikipedia concepts in the whole sentence, it usually comes from the context similarity between two Wikipedia articles. The context can either be the content of the article or the surrounding words of links which point to the article. Kulkarni et al. [16] proposed a method that considers compatibility and relatedness in disambiguating a term. Compatibility is measured based on similarity features between the context of the term and the concept's context. Relatedness between two concepts is computed by the category structure in Wikipedia and the cross references between the two articles. The above two factors are combined to form an optimization model, which is approximately solved by greedy hill-climbing. Tonelli [26] developed Wiki Machine based on an SVM model by combining local and global kernels. Local kernels are built from non-contiguous n-grams and part-of-speech tags while global kernels use bag-of-words and latent semantic to capture the topical information. Ratinov [22] did a similar work to Tonelli, using both local and global information to train an SVM model. Different from Tonelli's work the local information they used is context similarity, while the global information is the Wikipedia article/concept similarity. They also trained a linear SVM classifier to decide whether a term should be tagged or not. This classifier is trained based on the link distribution of Wikipedia corpus, attempting to avoid labeling general terms which are usually not labeled by people in Wikipedia.

Strictly speaking, the link co-occurrence approach adopted in this paper can be categorized as a global approach because the link co-occurrence matrix that we obtain from the iterative algorithm is

indeed global information among the concepts, while at the same time during wikification time we disambiguate several neighboring terms together in a sliding window which forms a local context. Our method differs from all existing methods in: 1) it doesn't use a bag-of-words (or bag-of-terms) model at all; 2) it relies on co-occurrences between Wikipedia links or concepts rather than words or surface terms; and 3) wikification algorithms which depends on the Wikipedia link structures or the link distribution face the problem of link sparsity in Wikipedia, our iterative link enrichment process solves this problem and can be used as a complementary pre-processing step for many of these algorithms.

5.3 Other Work Related to Wikification

There are some other work related to Wikification. Strube and Ponzetto [25] proposed path based, information content based and text overlap based measures on the semantic relatedness between two Wikipedia articles/concepts. These relatedness features can be used to train some statistical models [16, 22]. Different from them, our approach is based on co-occurrence, which is a more natural relation between concepts. To help users better understand the complex medical terminology, He et al. [14] tried to generate links from narrative radiology reports to Wikipedia automatically. He's work didn't share the same goal with ours since their main concern is to resolve the ambiguity of anchor medical terms other than semantic ambiguity. Lui et al. [18] applied wikification on the generation of hypertext for web based learning. Given a set of documents, after wikification on them, they tried to replace the destination of the link with a document from the set which is semantically related to the original destination(Wikipedia article). Jadidinejad et al. [15] implemented the wikification approach by Milne et al. [20] and applied it on structured query generation. Miao and Li proposed sentence wikification for query oriented summarization, but only applied an exact match strategy. The proposal in this paper complements much of the above work as it improves the accuracy of wikification, a key component in these systems.

6. CONCLUSION

Phrase Sense Disambiguation is an important problem in natural language processing. In this paper, we show the possibility to use Wikipedia as a reliable and comprehensive source to bootstrap a process that disambiguates unlinked noun phrases in Wikipedia articles and then use the obtained co-occurrence information to disambiguate noun phrases in new documents, a process known as wikification. Our evaluation shows that the co-occurrence based wikification can achieve high accuracy (about 82.58% on F1) efficiently (over 1000 words per second) using just 10,000 popular Wikipedia articles and moderate computation resources.

7. REFERENCES

- [1] Gulag: Wikipedia article. <http://www.wikipedia.org/Gulag>.
- [2] Wikipedia. <http://www.wikipedia.org>.
- [3] S. Bartunov, A. Boldakov, and D. Turdakov. Wikifyme: Creating testbed for wikifiers. In *Proceedings of the Spring Researchers Colloquium on Database and Information Systems, Moscow, Russia*, volume 2009, 2011.
- [4] C. Boston, S. Carberry, and H. Fang. Wikimantic: disambiguation for short queries. In *NLDB'12*, pages 140–151, 2012.
- [5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7*, pages 107–117. Elsevier Science Publishers B. V., 1998.
- [6] M. Carpuat and D. Wu. How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. *Proceedings of TMI*, pages 43–52, 2007.
- [7] Y.-J. Chung, S.-J. Kang, K.-H. Moon, and J.-H. Lee. Word sense disambiguation using neural networks with concept co-occurrence information. In *NLPRS*, pages 715–722, 2001.
- [8] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, volume 2007, pages 708–716, 2007.
- [9] H. Dai, C. Wu, R. Tsai, and W. Hsu. From entity recognition to entity linking: A survey of advanced entity linking techniques.
- [10] D. Fernández-Amorós, R. H. Gil, J. A. C. Somolinos, and C. C. Somolinos. Automatic word sense disambiguation using cooccurrence and hierarchical information. In *NLDB*, pages 60–67, 2010.
- [11] S. Fernando and M. Stevenson. Adapting wikification to cultural heritage. *EACL 2012*, page 101, 2012.
- [12] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, pages 1625–1628. ACM, 2010.
- [13] J. A. Guthrie, L. Guthrie, Y. Wilks, and H. Aidinejad. Subject-dependent co-occurrence and word sense disambiguation. In *ACL*, pages 146–152, 1991.
- [14] J. He, M. de Rijke, M. Sevenster, R. C. van Ommering, and Y. Qian. Generating links to background knowledge: a case study using narrative radiology reports. In *CIKM*, pages 1867–1876, 2011.
- [15] A. Jadidinejad and F. Mahmoudi. Query wikification: Mining structured queries from unstructured information needs using wikipedia-based semantic analysis. In *Working Notes for the CLEF 2009 Workshop*, 2009.
- [16] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *SIGKDD*, pages 457–466. ACM, 2009.
- [17] H. Li and N. Abe. Word clustering and disambiguation based on co-occurrence data. In *COLING '98*, pages 749–755, 1998.
- [18] A. Lui, V. Ng, E. Tsang, and A. Ho. Generation of hypertext for web-based learning based on wikification. *Enhancing Learning Through Technology. Education Unplugged: Mobile Technologies and Web 2.0*, pages 280–290, 2011.
- [19] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242, 2007.
- [20] D. Milne and I. Witten. Learning to link with wikipedia. In *CIKM*, pages 509–518. ACM, 2008.
- [21] R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), 2009.
- [22] L.-A. Ratnov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL*, pages 1375–1384, 2011.
- [23] B. Skaggs. *Topic Modeling for Wikipedia Link Disambiguation*. PhD thesis, 2011.
- [24] C. Stokoe, M. P. Oakes, and J. Tait. Word sense disambiguation in information retrieval revisited. In *SIGIR '03*, pages 159–166, 2003.
- [25] M. Strube and S. P. Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, pages 1419–1424, 2006.
- [26] S. Tonelli, C. Giuliano, and K. Tymoshenko. Wikipedia-based WSD for multilingual frame annotation.

Artificial Intelligence, 2012.

- [27] J. Véronis. Hyperlex: Lexical cartography for information retrieval. *Computer Speech and Language*, Special Issue on Word Sense Disambiguation, 2004.