Unsupervised Online Wikification: A Practical Approach

Mayank Kejriwal and Tomas McCandless

University of Texas at Austin
{kejriwal,tomas}@cs.utexas.edu

Abstract. Over the last decade, Wikipedia has emerged as the *de facto* global knowledge base owing to its exponential growth in popularity. At the same time, there are many unstructured documents on the Web that can be *enriched* by recognizing *entities* within the document and linking them to specific pages in Wikipedia. Recent approaches have tackled this problem by exploiting global and local feature engineering methods on Wikipedia *dumps*. These approaches are supervised, in that they require training on manually provided data before they can disambiguate links to unknown entities. While useful in many contexts, such methods lack two features. The first is the ability to perform instantaneous and therefore, *online*, wikification in articles that are published and consumed in real-time. An example of a use-case would be a breaking news article. As a new article is written about a breaking news event, it would be possible to augment the text via links to specific pages in wikipedia. The second is *automation*, since online wikification is likely to be useful in an unsupervised setting due to the evolving nature of the Web. The problem is exacerbated by regular updates and additions to Wikipedia; hence, a dump cannot always be relied upon for either of the two goals stated. In this paper, we present a solution to both these problems by making judicious use of the Wikipedia API and favoring precision over recall. To control complexity and simplify implementation, we use simple, but discriminative features. Experimental results on a test dataset used in a prior offline approach demonstrate that, in conjunction with the Wikipedia API, the best features can achieve over 80 percent accuracy on the online wikification task.

Keywords: Wikification, Entity Linking

1 Introduction

Wikification is the task of identifying entity *mentions* in text and linking those mentions to referent Wikipedia pages. The motivations for solving such a problem effectively are numerous. Wikification has been shown to be useful in many natural language processing tasks, including text classification [4] and numerous other tasks [6]. Although many wikification works differ in the document collection which they are attempting to Wikify and also the expressions in those documents, they share the common problem denoted *Disambiguation to Wikipedia*

(D2W) [10]. As an example, consider the simple sentence, *I am attending Wimbledon this summer*. Assume that a named entity recognition (NER) [9] system has managed to detect *Wimbledon* and *summer* as relevant entities. These entities would then have to be correctly linked to the Wikipedia pages referenced by URIs `http://en.wikipedia.org/wiki/The_Championships,_Wimbledon` and `http://en.wikipedia.org/wiki/Summer`. A closer look at this simple example shows two potential problems, assuming an off-the-shelf NER system works without error. The first is the decision of *what* to disambiguate. One could make the (subjective) argument that the entities *I* and *summer* do not need to be linked, and might, in fact, distract a reader from the overall meaning of the text. Certainly, one can argue that the user experience would be less than pleasant if every single entity is linked to some Wikipedia page. The second problem is that of semantic ambiguity. Wimbledon here clearly refers to the prestigious tennis tournament, but Wimbledon is also a place in England. The problem is exacerbated when we consider that for a given entity, there are many candidate Wikipedia links. Another challenging aspect of the problem is that the ambiguity is bidirectional. In other words, multiple surface forms can refer to the same entity in the knowledge base, and a single surface form can potentially refer to multiple entities in the knowledge base. For example, George Bush, the entity who was president in 2003, can be referred to by "George Bush", "George W. Bush", "Bush", or other surface forms. Similarly, the surface form "Kennedy" can refer to any number of entities from the Kennedy family, including the president who was assassinated. Furthermore, when performing NED [2], the knowledge base is typically assumed to be incomplete, meaning there are entities that exist in the world but do not have corresponding records in the knowledge base. This is one challenging difference between NED and WSD – when performing Word Sense Disambiguation, it is often assumed that the dictionary containing all the word senses is complete. Correctly disambiguating the entity in terms of the link is a challenging problem, even in the supervised setting.

In this paper, we present a framework for unsupervised entity linking using the Wikipedia API and basic but representative features. We are able to narrow the gap between the supervised and unsupervised frameworks to less than 10 percent, and show improvement of over 5 percent over baseline features, using a tri-gram model and simple enhancements to those baseline features.

The rest of the paper is as follows. Section 2 describes some related work in this area, while Section 3 lays out the problem formulation. Section 4 describes the framework in some detail followed by the experimental results in Section 5. Section 6 concludes the paper.

2 Related Work

Wikification has received increasing research attention since the paper by Bunescu and Pasca in 2006 [1]. They used an SVM kernel and the context around the ambiguous mention for disambiguation. However, a different model had to be trained for each mention. This limited the solution to only those ambiguous men-

tions on which an SVM model had already been trained. The algorithm couldn't be applied to mentions that had never been seen before (and trained on) in the training data. Mihalcae and Csomai published an influential work in 2007 that used Word Sense Disambiguation (WSD) [11] to perform entity disambiguation [7]. Two techniques used by their proposed system, *Wikify!*, included computing lexical overlap between candidate Wikipedia pages and the context of the ambiguous mention, and training a Naive Bayes classifier using ground truth from Wikipedia *itself*. It is commonly observed that a Wikipedia page itself contains hyperlinks to other Wikipedia pages. Thus, Wikipedia pages themselves provide valuable training data, which was exploited by the supervised system in that paper.

Since 2007, increasingly sophisticated methods have been proposed to solve the problem, at the price of increasing complexity[8],[2],[10],[5]. Specifically, such methods espoused moving from *local* methods to *global* methods. Examples of local methods are the works by Bunescu and Pasca and the Wikify! system. Such methods disambiguate each entity mention *in isolation*. The disambiguation of the p^{th} entity mention is therefore independent of the disambiguation of the $(p+1)^{th}$ entity mention. On the other hand, global methods propose to increase accuracy by performing *collective* disambiguation. That is, the disambiguation of an entity is now dependent on the disambiguation of other entities in the text. This makes sense if the document has *thematic coherence*, which is a characteristic of many real-world documents.

Ratinov et al. formulate the problem precisely in their work on D2W [10], and show that existing systems adhere to their formulation. Furthermore, they show that an optimal solution to the global problem is NP hard. The differences in existing systems proposed were those involving the features they employed and also, the approximation technique they used. One important point to note is that all systems assume supervision. That is, the approximation algorithm (typically framed as a machine learning classification problem) assumes a training set, which in the spirit of the *Wikify!* system, is often derived from Wikipedia itself. Although our methods and features are considerably simpler, their novelty lies in their unsupervised application and instantaneous run-time. The wikification is achieved on-the-fly. That is, it is online and uses the most recent version of Wikipedia, as opposed to previous work that relied on training data from manually downloaded dumps of a previous date.

3 Problem Formulation

3.1 Problem definition

Consider an unstructured document (equivalently denoted *raw text*). We assume that a set of p *mentions* $M = \{m_1, m_2 \dots m_p\}$ can be extracted from each such document, where each *mention* is defined as a fragment of text that occurs somewhere in the body of the text. Assume a simplified (but still accurate) representation of the Wikipedia knowledge base as a set of T *links* $L = \{t_1, t_2 \dots t_T\}$. For the purposes of this section, we assume T is a constant. In reality, T is a

function of underlying variables like *time* and *language*. Since we assume the English language in the rest of this paper, language variability is not an issue. *Temporal* variation of wikipedia was provided as a motivation for doing online wikification but it is reasonable to assume (for the sake of formalism) that T is a constant. This does not really affect the problem formulation being described. There are two tasks that now need to be undertaken, *ranking* and *linking*. Ranking assumes that some underlying system (for example, the Wikipedia API but also offline systems previously proposed in the literature) is able to generate a set of candidate links for a given mention m (we drop the subscript for ease of exposition). Let this candidate set be $L'_m \subseteq L$. Intuitively, such a candidate set represents candidates for disambiguating the mention. The goal of ranking is to turn this set into an ordered data structure (like a list), based on some underlying algorithm that could rely on (for example) feature selection or some other machine learning technique. We explicitly assume that each mention has *at most* one correct candidate disambiguation. A *k-correct* ranking would then place the correct candidate disambiguation somewhere in the top k, if such a candidate disambiguation exists.
Linking is a binary classification problem that takes as input a mention and a *single* candidate disambiguation (perhaps output by a ranker, or provided by an oracle) and, with the help of contextual features, decides whether the candidate disambiguation is the correct one. Although linking might seem easier than ranking, it was empirically shown to be quite difficult in practice, even in the supervised setting. The reason is that contextual features can be very misleading with respect to *nulls*. That is, it is difficult to tell when *not* to link a candidate. In this paper, we are exclusively concerned with the *unsupervised* linking problem. As described subsequently, we can do away with ranking by making conservative decisions with respect to the Wikipedia API output. The problem then is to make a yes/no decision on whether to link a mention to a candidate link.

3.2 Features

In the previous section, we did not mention how the ranking and linking operate. In the most recent state-of-the-art systems, they make use of both global and local features. Both ranking and linking can make use of global features. For ranking, assume that we have sets of candidate links for all mentions $m \in M$, L'_m. The ranking of each list (say L'_{m_1}) can either occur independently of each other or be dependent on each other. We do not state the formal optimization problem here, but as Ratinov et al. showed, the global optimization is NP-hard. For linking, an analogous procedure applies. We can either link each mention to the candidate as an isolated task, or perform collective linking.

4 System

The overall system schematic is shown in Figure 1. The only input to the system is the unstructured document or the raw text. An off-the-shelf Named Entity

Recognizer (NER) first extracts the set of mentions from the raw text. These mentions are sent on to the Wikipedia API. In this paper, we use the Stanford NER [3], which was also the NER used by the authors of the GLOW system [10]. The NER is available freely[1]. As we describe in the Dataset section, the Stanford NER had already been applied on the test data and the mentions were extracted and included as part of the test suite made available with the GLOW project. We attach the test data with the submission of this project. Note that we never explicitly had to use the Stanford NER ourselves. In the subsequent sections, we describe the remaining sub-modules that were used in the framework.

4.1 Wikipedia API

Given a list of entity mentions that is output by the NER submodule, we rely on a wikipedia API [2] to extract the raw text of the corresponding wikipedia page, rather than scraping wikipedia pages ourselves. Some named entities are ambiguous in the sense that they could refer to one of many different wikipedia articles. For example, the named entity "labour party" could refer to "Social Democratic and Labour Party", "Socialist Labor Party of America", or any number of other labour parties from many different countries. For our particular application we are more focused on precision than recall, so when this happens we choose not to perform any linking. For each named entity we also extract the outgoing links from the corresponding wikipedia page. Our interface to the wikipedia API is implemented in python.

4.2 Feature Generator

Features form the backbone of our system. A literature survey of existing wikifiers show that the primary difference between them lies in the complexity and expressivity of their features. Online wikification demands features that are simple yet representative. Towards this end, we devised three simple features to be used in conjunction with any n-gram model (including a bag of words model). The first of these is TF-IDF. The IDF is computed over the set of Wikipedia pages returned for the mentions in a raw text. Thus, a different IDF (an 'on the fly' IDF) would be computed for each raw text document. We also experiment with TF and stop-words, that is, removing common stop words before computing the normalized TF vectors. Finally, we combined stop words and IDF with TF. The baseline feature is simply normalized TF without any IDF computation or preprocessing. Note that the feature generator outputs a feature vector for each candidate Wikipedia page corresponding to a mention in the raw text and also outputs a feature vector for the raw text itself.

[1] `http://nlp.stanford.edu/software/CRF-NER.shtml`
[2] `http://www.mediawiki.org/wiki/API:Main_page`

4.3 Top N Candidates

The goal of this module is to take the Wikipedia feature vectors extracted by the feature generator and the raw text feature vector and an additional input N (which is experimentally determined, as subsequently described). Each Wikipedia feature vector is scored by taking its dot product with the raw text feature vector. The top N Wikipedia vectors are marked as *correct* and the corresponding mentions are linked to these Wikipedia pages. We experimentally show how accuracy varies with N in subsequent sections.

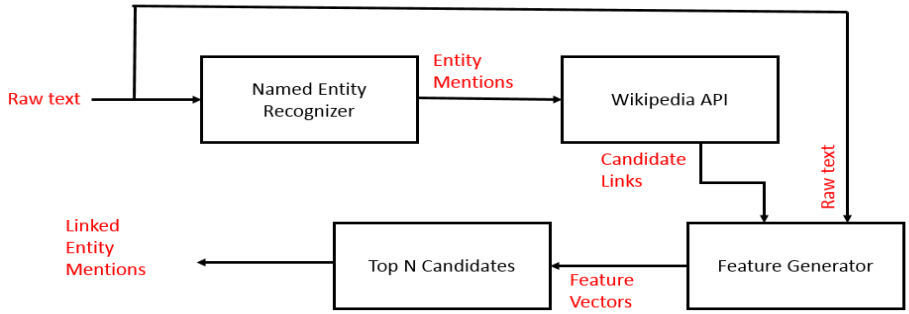

Fig. 1. The overall schematic of the proposed online wikifier.

5 Experiments

In this section, the dataset is first described, followed by the methodology. This is followed by some results and a discussion of those results.

5.1 Dataset

We used the *Wiki* test set first developed by the authors of GLOW [10]. They made the dataset available on the companion project website[3]. The Wiki data came with training data and test data and were extracted from 'difficult' Wikipedia paragraphs (baseline features in GLOW yield at least a 10 percent error rate on these paragraphs). The training data was much larger than the test data. In their work, they first trained a sophisticated system using the training data and then evaluated both a ranker and linker on the test data. Since we're only evaluating unsupervised linking, we used only the test data. Note that the data consists of 40 documents with a total of 839 mentions (extracted by Stanford NER) in those

[3] `http://cogcomp.cs.illinois.edu/data`

documents. Some of these were nulls (that is, had no corresponding Wikipedia page according to the provided ground truth) and for others, the Wikipedia API returned multiple candidates (and which were hence ignored). In total, there were 724 mentions for which the API returned a single link. Some of these links were 'wrong' in that they corresponded to a null in the ground truth or a *different* Wikipedia page than what the API returned. On average, we found about 78 percent of the returned links to be correct. Thus, a naive approach that would randomly sample N links (we do use this as one of our baselines, in fact) would return between 75 and 78 percent on average.

5.2 Methodology

We experimented with each of the features as well as their combinations, using three models: bag of words, bi-grams and tri-grams. For each model, two sets of experiments were conducted. Both experiments use N as the independent variable, and for each document, mark the top N scoring (where the score is the dot product of the feature vectors) candidate links as correct. Accuracy is measured (that is, how many of these chosen links are correct) against N. For example, if all N chosen links are correct, the accuracy is 100 percent. Although N does not technically represent *recall*, it is somewhat analogous, since the higher N is, the more links will show up in a document. Note however that since we have multiple documents, there are two different ways (at least) to calculate accuracy for a given value of N when the algorithm is run on all documents. The first, which we call the *surface forms accuracy*, is to compute the accuracy of disambiguation across the full set of mentions in the entire corpus. There is no averaging across documents. The second is to compute the accuracy for each document individually, and to then average accuracies across documents, with each document getting equal weight (*documents accuracy*). As a simple example of the difference, suppose there are two documents with 20 mentions in the first document and 40 mentions in the second. Suppose, on average, that the accuracy on the first document is 80 percent, and on the second is 75 percent. The documents accuracy would then simply be 77.5 percent. However, the surface forms accuracy would be $(.75 * 40 + .80 * 20)/60 = 76.667$ percent. We show the results for both forms of averaging in the experiments. Recall that one of the features proposed is that of stop words removal. Stopwords were taken from a Web resource[4]. We used two baselines: random selection of N candidates and normalized TF. By supplementing normalized TF with IDF, stop words removal or both, we measure the enhancement in performance for all the models.

5.3 Results and Discussion

The first interesting trend observed in all the figures is that we don't see a standard precision-recall curve where precision, or accuracy, goes down as N increases. What happens instead is that the figures yield a flat curve of about

[4] `http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words`

78 percent for the surface forms accuracy and 77 percent for the documents accuracy. In general, absolute differences aside, the surface forms accuracy and documents accuracy curves were found to mirror each other closely (with only about one percent difference in absolute numbers) so we refer to a single curve in the following discussion (surface forms). The value of N at which accuracy seems to peak is between 15 and 20 for all the figures.

A somewhat surprising aspect is that the TF baseline actually performs worse than the random baseline till this peak value, after which it usually shows slight improvement before leveling off, along with the rest of the curves. Notice also that the combination of features (TF,IDF and stopwords) works best but except on the bigrams model the other features actually show better performance on lower values of N. The best performance for N between 2 and 5 was demonstrated by TF IDF on the trigrams model. Over that range, it briefly outperforms the combined three features of TF, IDF and stopwords.

It is interesting to compare the unsupervised performance with the supervised performance using a much more sophisticated method and features than what we've used (in the original GLOW paper) [10]. Using a trained linker, the authors were able to achieve about a 92 percent accuracy . However, as mentioned before, the training data was much larger than the test data and the ranker accuracy was assumed to be perfect. Thus, the only task addressed was determining the 'nulls' in the ground truth. On the other hand, we had to rely on the Wikipedia API for our 'ranker candidate'. Some of these were wrong. In this sense, the unsupervised linker problem is harder since we have to determine both nulls (mentions that don't correspond to a Wikipedia page at all in the gold standard) as well as mentions for which wrong links were returned by the API. Although the 10 percent difference is still considerable, we believe it can be narrowed by even better features. An encouraging sign is that by using a tri-gram model with some intuitive enhancements we were experimentally able to push the accuracy to over 83 percent for a small practical range of N.

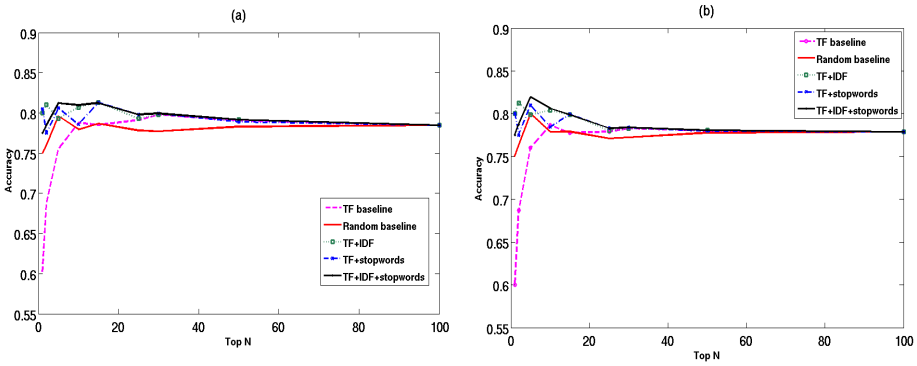

Fig. 2. Results of the procedure when no n-gram features are employed. (a) represents surface forms accuracy and (b) shows documents accuracy

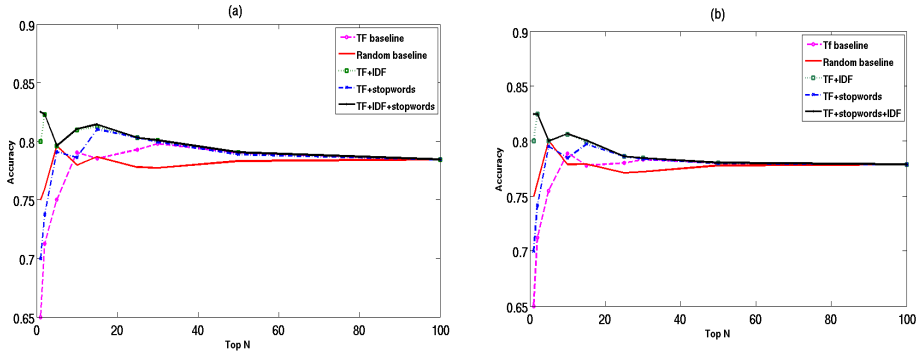

Fig. 3. Results of the procedure when TF includes bi-gram features. (a) represents surface forms accuracy and (b) shows documents accuracy

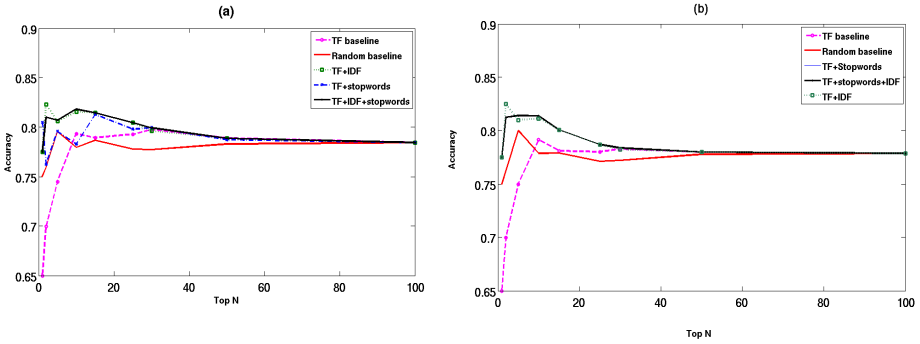

Fig. 4. Results of the procedure when TF includes tri-gram features. (a) represents surface forms accuracy and (b) shows documents accuracy

6 Conclusion

In this paper, we attempted to present a practical solution to unsupervised entity linking using the Wikipedia API. Using bi-gram and tri-gram models together with enhancements to basic features like TF, we were able to achieve up to a 5 percent performance boost on baseline features and narrow the gap between supervised and unsupervised linker accuracy to less than 10 percent. Although more performance boosts are needed, fuelled perhaps by better features, we believe that fast run-time and judicious use of the Wikipedia API makes

the method a feasible starting point. We hope that future work will addressing narrowing this performance gap.

References

1. R. C. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *EACL*, volume 6, pages 9–16, 2006.
2. S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 7, pages 708–716, 2007.
3. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.
4. E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.
5. X. Han and J. Zhao. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 215–224. ACM, 2009.
6. S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM, 2009.
7. R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM, 2007.
8. D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.
9. D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.
10. L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.
11. M. Stevenson and Y. Wilks. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics*, pages 249–265, 2003.