CS 341 Final Report

Ashwin Apte
Dimuth Kulasinghe
Neeraj Pradhan

1 Introduction

Wikification is the process of recognizing important concepts and phrases in a given text, and linking these concepts and phrases to the Wikipedia pages that explain them. The basis of the process is based on the breadth and depth of knowledge that Wikipedia provides, as well as the effectiveness of linking key phrases in knowledge comprehension, as demonstrated by Wikipedia. The problem of developing an effective Wikifier is rather open ended, but the popular approach involves identifying, in the input document, *anchor texts*, or phrases that were used as links in the actual Wikipedia corpus, and determining which anchor texts to select (Link Prediction), and where the anchor texts should be linking to (Disambiguation). The disambiguation component is especially important; there are several anchor texts that link to multiple pages (entities), in the Wikipedia corpus, often about completely unrelated concepts.

The quality of a given Wikifier is somewhat difficult to evaluate; like ranking systems, the basis for what is considered an important concept is up to interpretation and human intuition. We judge the quality of a Wikifier, then, by taking a manually annotated document, and then seeing how accurately (in terms of both link prediction and disambiguation) the Wikifier matches these results. We used, for our manual annotations, existing Wikipedia documents. Our goal was to develop a Wikifier that utilizes special features present in Wikipedia pages that are not present in standard documents to try and improve upon existing Wikification procedures.

2 Prior Work

James Gardner and Li Xiong of Emory University [4] used a more NLP-oriented approach, learning how authors tended to select links, rather than matching exact n-grams as see in the corpus. Features used in determining links included semantic features of the term in question, features pertaining to the document being processed, such as n previous and subsequent words, and features related to the training corpus, such as tf-idf. Finally, other extraneous features were incorporated such as link probability and relatedness, features we utilized in our classifier. Effectiveness of the Wikifier was evaluated on Wikipedia pages from various categories, such as Biology, Health, People, etc; the system managed a precision rating of 77% across the categories, and a recall of 59%.

The Wikify! wikification system [1] used a keyword extraction system similar to our own, matching exact phrases in the text document to existing anchor texts. Features in link prediction include td-idf measurements, χ^2 independence tests, and link probability, as a ratio of the percentage of times a phrase occurs as a link. Link prediction was performed with 53.37% precision and 55.90% recall. For disambiguation, Wikify! uses both knowledge-based and data-driven methods. Knowledge-based methods include measuring contextual overlap in ambiguous terms; if the term in question is 'bar', the dictionary-use of words surrounding 'bar' are used to determine the sense of 'bar'. Data-driven approaches involve creating feature vectors for all occurrences of an ambiguous term in a document, and running a Naive Bayes classifier to determine usage. Disambiguation was performed with 94.33% precision and 70.51% recall.

David Milne and Ian Witten from the University of Waikato [2] developed a wikifier that was especially effective at disambiguation, primarily using two features coined commonness and relatedness.

Commonness is a general measure giving the general probability that a given anchor text links to a certain page, while relatedness considers, for a given link, how similar the contents of the link are to the generated unambiguous links in the document. Various other features were used as well, such as a confidence measure in the disambiguation, a generality measure, and so on. The wikifier achieved a disambiguation precision of 97% and recall of 96.6%.

3 Data

Our data consists primarily of the most recent Wikipedia dump (4/4/13), stored as an XML file and consisting of around 13 million individual articles and metadata involving each article [5]. We retrieved, for every article, its title, text body, and categories; Wikipedia utilizes a built in category system to help relate articles. We retrieved, on a first pass, the anchor texts in that article from the text body (parsed using regular expressions `[[*]]`), and on a second pass coincidental occurrences of any potential anchor text in the articles, of phrases up to 5 words in length.

We normalized all text blocks in the articles through case-folding and replacing all non alpha-numeric characters with spaces. We also took into account redirecting articles, where the redirects were treated as the articles they were redirecting to. Finally, we also utilized a data dump giving relations amongst Wikipedia categories; this allowed us to create our own categorical tree through which to develop additional features for the wikifier.

4 Solution Methodology

Wikification can be broken down into two component problems:

- Identifying key concepts in a document that need to be linked, i.e predict what to link or link prediction
- Identify the correct sense for each of these identified concepts, i.e. predict where to link to or disambiguation

We discuss the two sub-problems below.

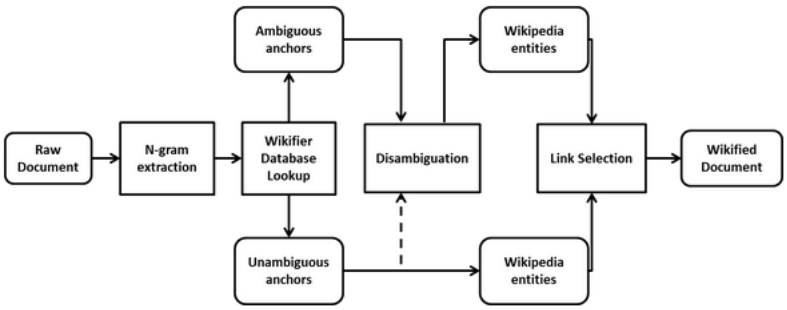


Figure 1: Block Diagram view of the Wikifier

4.1 Link Prediction

This task identifies words or phrases from a given document which need further explanation by a link to the relevant Wikipedia page. In general, there should be enough links on the page so that each key concept is explained, but at the same time too many links make the document unreadable, and readers may lose focus from the main topic. For example, about 6% of words on Wikipedia pages are

hyperlinked to other sources [1]. The approach we take to this problem is that whatever is important enough to be linked has been linked somewhere within the Wikipedia corpus. Though this statement may not be necessarily true, the size and diversity of the WIkipedia corpus allows us to make this assumption with a fairly high degree of certainty that nothing of interest will be missed. We preprocess the entire WIkipedia corpus and extract all anchor texts that appear in the corpus. We treat this as the master list of all possible anchor texts. We also keep count of the number of times each of these n-grams appears in the corpus, as a link or as plain text. From these numbers we extract the Link Probability for each anchor text n-gram.

$$Link\,probability = \frac{No.\,times\,an\,n-gram\,appears\,as\,anchor\,text}{No.\,times\,the\,n-gram\,appears\,in\,the\,corpus}$$

Link probability turns out to be a very informative metric, as it on its own is able to predict the importance of an n-gram. We incorporate some additional signals to further hone link selection.

We calculate a *Relatedness* score for each sense of an anchor for disambiguation. The calculation of this score is described in the Disambiguation section. The *relatedness* score is the extent to which the given sense is related to the anchor, given the context in the article. We use the *relatedness* score of the disambiguated entity for each anchor as another signal to inform how important the anchor is. A high *relatedness* score should mean that the anchor links to a page which is related to the current document, and thus has more value as a link than another anchor with a lower *relatedness* score.

We also use the *disambiguation confidence*, explained in the disambiguation section. This is the output score of the machine learning classifier. This is a measure of how confident we are of the disambiguation. This should weed out anchors for which we are not very confident of the disambiguation, thus reducing the risk of an incorrect link. It turns out that disambiguation confidence does not add to the performance of link selection.

To wikify a document, we create all possible n-grams up to length $n = 5$ from the text. We check these against our master list of anchor text. We retain the n-grams that appear in the master list as possible candidates. We then look up all candidate sense entities for each one of these n-grams. These are the possible entities to which the anchor has been linked to within the wikipedia corpus. We have pre-calculated *link probabilities* for these anchors. We run these anchors through the disambiguation engine to obtain *Relatedness and Disambiguation Confidence* scores. Multiple anchors may disambiguate to the same entity in a document, but we would want to link to each entity only once. We invert this problem by selecting the top-k entities to link to. Each entity is picked only once. If there are multiple possible anchors for an entity, we pick the anchor with the highest link probability.

We use the product of Link Probability, Relatedness and Disambiguation Confidence as a score for link selection. We rank entities by the product score, and select the top-k. The value of k on Wikipedia is in the 5% to 6% range. We leave the parameter k as a tuning parameter to select the density of links that should appear on a page. The user may select k based on application. A popular news item may require fewer links, whereas an abstruse scientific paper may need more links for elucidation.

4.2 Disambiguation

Disambiguation attempts to select the right sense for each link to point to, based on the context present in the document. We use two key scores to disambiguate a given anchor; Commonness and Relatedness. Commonness is pre-calculated for every anchor-text entity combination that appears within the wikipedia corpus. Commonness is the probability that a particular anchor will link to a particular sense entity.

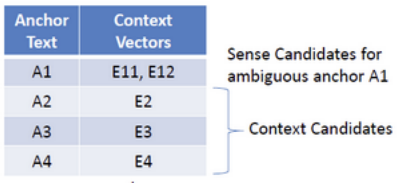
$$Commonness = \frac{No.\,of\,times\,an\,anchor\,links\,to\,an\,entity}{No.\,of\,times\,the\,anchor\,appears\,as\,a\,link\,in\,the\,corpus}$$

Commonness does not use any context information. This is just the prior probability of an anchor linking to a particular entity, before taking into account context information.

Relatedness is calculated from context information. On every page, there are some anchor candidates that always disambiguate to a fixed entity (i.e. commonness score of 1). Empirically, we find that around 25% to 30% of the possible anchor candidates in a document tend to be unambiguous. We use these as context markers. We calculate the similarity between a candidate sense for an ambiguous anchor with each of the entities linked by the unambiguous anchors. For the similarity measure, we use Jaccard similarity. Similarity between two entity pages is given by the Jaccard similarity between the set of pages to which each entity page links to.

We weight the relatedness measure to make sure the importance of context terms is captured. Some context markers may be more valuable than others. We use the link probability of each context marker as a weight. A high link probability indicates that the context marker is an important term. and thus more likely to provide more reliable context information. The other weight we use is the average similarity between a particular context marker and all other context markers. This weight is designed to weed out unambiguous anchors that may be unrelated to the main topic in the document. In taking average similarity, the assumption is that a large proportion of the context markers are related to the documents topic. The weight we use for each context marker is an average of the above two weights.

Relatedness thus is a sum of similarities between a candidate sense and each one of the context markers, weighted by the weight for each context marker discussed above. An example of the Relatedness calculation below:



$$\text{Rel(E11)} = \sum_{i=2}^{4} L_i \cdot W_i \cdot Sim(\text{E11}, E(i))$$

$$W_i = Avg_{i \neq j}(Sim(E(i), E(j)))$$

Figure 2: Example Relatedness Calculation

We also derive a Context Quality metric as the sum of weights for all the context markers. This metric indicates the quality and quantity of context that exists in a document. A high score indicates that there are a relatively large number of high quality context markers. This context quality score is useful for machine learning approaches to disambiguation discussed below.

The disambiguation engine takes in a set of anchor candidates. It identifies the unambiguous anchors as context markers, and calculates average similarities for each of these context markers with the remaining context markers. It then calculates the sum of similarities between each candidate sense for each candidate anchor with each of the context markers, weighted by the average similarity and link probability for the context marker. This sum is the relatedness measure. The naive disambiguation

method takes the product of commonness and relatedness as the disambiguation score. The entity with the highest disambiguation score for a given anchor is selected as the disambiguated entity. The disambiguation engine thus returns a disambiguated entity for each ambiguous anchor. Unambiguous anchors by definition do not need disambiguation. We calculate relatedness scores for entities linking to unambiguous anchors as well since these are needed as part of the link selection process described in the previous section.

4.3 Machine Learning Approach to Disambiguation

Wikipedia pages can be treated as a source of labelled data for a machine learning classifier for disambiguation. Each link on Wikipedia has been placed manually by a contributor. This may introduce an element of inaccuracy in the corpus, since a contributor may select a link at their discretion. But taken as a whole, it is a large corpus of manually annotated links that gives us millions of examples for training a classifier.

We extracted links from a set of 500 randomly selected wikipedia articles. We chose articles from articles that have appeared as Featured Articles on the wikipedia home page. This ensures that the article has a substantial amount of text and links. Also, these articles are likelier to have more reliable links to learn from, since they have been curated to appear as featured articles. The 500 pages gave us a total of 72,404 links. The actual entity that each of these links points to gives us a positive example, and all other possible senses of the anchor give us negative articles. Our data set consisted of 72,404 positive examples, and 2,775,654 negative examples.

We used the following features for the machine learning approach:

- Commonness
- Relatedness
- Context quality
- Minimum depth in category tree: This feature attempts to capture the generality of the article. Wikipedia has a category hierarchy that maps each article to a number of related categories. We pre-calculated the minimum depth of each entity in Wikipedia. A given entity may be tagged with multiple categories. We again pick the minimum depth value from this set. This ensures that we have the minimum category hierarchy depth over all categories that the entity may be part of. We calculate this feature value for both the page on which the entity appears, as well as for each entity.
- Textual Similarity: This is word level Jaccard similarity between the anchor and the entity. Very often, the anchor text and the linked entity are very similar. This feature tries to capture the similarity and relate it to disambiguation.
- Word count: Word count for each article.
- Number of occurrences of the n-gram as an anchor in the wikipedia corpus
- Number of occurrences of the n-gram in the entire wikipedia corpus

For the classification task, we experimented with Logistic Regression, Naive Bayes and Random Forest Decision Trees. We found that the Random Forest classifier performs best in our implementation. Our hypothesis is that this is due to the non-linearity in the objective function itself, which a Random Forest classifier is able ascertain best. For instance, if the relatedness score is insufficient, the classifier needs to make a decision based on the commonness scores and other features. On the

other hand, a high relatedness score is indicative of high relevance of the anchor-entity pair which should be favoured even if the commonness score is low for the pair.

We trained the Random Forest classifier on 2 million examples, and tested on 848,058 examples. Each example corresponds to an anchor-entity combination. Positive labels indicate the correct disambiguation. This is a highly unbalanced dataset with 25,297 positive examples in the test dataset (2.98 per cent of the test examples).

5 Results

5.1 Disambiguation Engine

For testing the disambiguation engine, we take the number of positive examples in the test set as the evaluation criterion, and report precision and recall against this.

The feature importances derived from the Random Forest classifier are given in Table 1. As can be seen, commonness and relatedness account for a major chunk of the importance contribution. It is worth noting that while features like textual similarity are deemed important based on this calculation, many of these features may be highly correlated due to which their contribution towards the final disambiguation accuracy is highly skewed. This is discussed further.

Feature	RF Feature Importance
Commonness	0.5213
Relatedness	0.2093
Textual Similarity	0.1327
Number of n-gram occurrences as anchor	0.0619
Number of n-gram occurrences as Wikipedia text	0.0412
Context Quality	0.0142
Page Word Count	0.0094
Entity Hierarchy Depth	0.0064
Page Hierarchy Depth	0.0035

Table 1: Feature Importance for Random Forest Classifier

The disambiguation engine was tested using different benchmarks as shown in Table 2. As seen in the table, commonness alone gives a very large contribution predicting disambiguation. This is true by construction of the feature itself, as this is indicative of the prior probability of linking to the most common entity.

Relatedness alone is not expected to predict disambiguation, but is nevertheless an important feature that can predict the right entity 68% of the times. Another benchmark is a simple product of commonness and relatedness used by Mihalcea et al [1], which gives us a disambiguation accuracy of 90%. This is a 3% improvement over commonness alone. The Random Forest classifier also gives very similar results.

The results seem to indicate that the signal from the other features that we use are already extracted through the commonness and relatedness scores. Further improvement in these scores will require either a different feature set, or a more robust notion of relatedness. For each of these approaches, we are predicting a disambiguated entity for every anchor. So the precision and recall scores are identical. Precision and recall are relevant only in the last case above, where we may predict more than one entity for an anchor, or not predict an entity at all for some anchor.

Results of the five approaches are as below:

Classification Metric	Accuracy
Commonness	0.8733
Relatedness	0.6757
Commonness x Relatedness	0.9042
Random Forest Classifier with 12 estimators – most valid sense	0.9093
Random Forest Classifier with 12 estimators – all valid senses	Precision – 0.93 Recall – 0.88

Table 2: Comparative Results for Disambiguation Engine

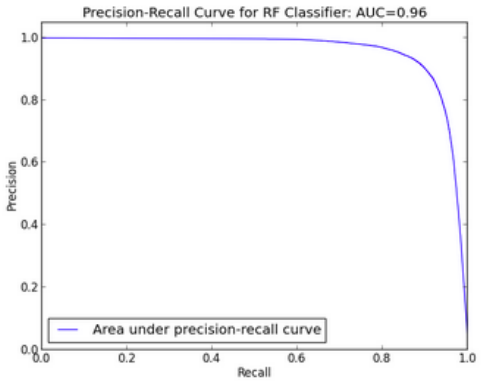


Figure 3: Precision-Recall curve for Random Forest Classifier

Comparing these results against a benchmark paper [2], we see very similar patterns. Commonness alone accounted for 90% of prediction accuracy in this study, and their best approach gave a 7% improvement over this. The difference in the commonness benchmark is possibly due to the evolution of the wikipedia corpus. The paper used a data set from 2007. As Wikipedia has evolved over time, the reversion to the most common sense has reduced over time, as more and more senses have been added for each anchor.

5.2 Wikifier

We tested the Wikifier as a complete unit, including the link detection and disambiguation part on a set of 50 randomly chosen articles from wikipedia featured articles. As mentioned in the link prediction section, we leave the parameter k for link density on the page for the user to select as per the application. For testing purposes, we chose the same number of links from each page as appeared on the original wikipedia article. Thus, we predict exactly the same number of links for each page as the original contributor decided was appropriate. In this context, the precision and recall are both identical.

For our set of 50 random articles, we get a precision and recall of **64.50%**. The precision and recalled can be tuned using the parameter k, i.e. the number of links to be included on a document. The precision-recall curve for the wikifier using k as the tuning parameter is as below.

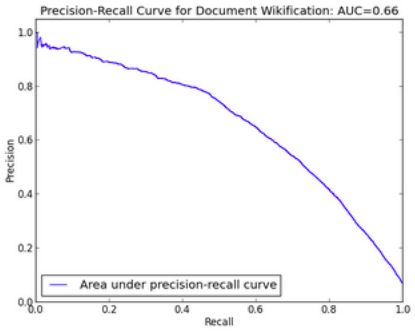


Figure 4: Precision-Recall curve for Wikifier

5.3 Wikifying Documents

The objective of our project is to wikify documents such as news articles, blogs etc. We have used Wikipedia articles to test our wikifier since that is the only available source of ground truth. The wikifier works reasonably well on other documents as well. We ran the wikifier on a set of news articles, and manually checked the resulting links. The wikifier seems to detect the important concepts and link them to the correct sense of the term on wikipedia. We do not present results of this testing here as there is no ground truth to evaluate against.

5.4 Evaluation of the Wikifier

Evaluation of the Wikifier against Wikipedia as the ground truth is fraught with dangers of trying to achieve a benchmark that in itself may be flawed. Links on Wikipedia are created manually, and are thus prone to the biases and intuition of the creator. Though in aggregate the corpus may be valuable, taken at an individual link level there is very little guarantee of the quality of links. Thus, the evaluation against Wikipedia should be treated with circumspection.

A major factor that we saw throughout the evaluation was that wherever our Wikifier does not predict the same entity as linked by the Wikipedia contributor, in most cases the entity we predict can also be plausibly selected as a link. It is a matter of personal preference to select one link over another. The table below shows examples where the predicted entity may appropriately replace the actually linked entity. Arguably, in some cases our prediction seems to be more appropriate than the actual entity. It is difficult to estimate this phenomenon in quantitative terms because again the closeness of the predicted and actual entities is a subjective evaluation devoid of ground truth.

Anchor	Predicted Entity	Actual Entity
british rule	british raj	british empire
catalan	catalan language	catalan people
arabic alphabet	arabic alphabet	arabic script
e-infrastructure	cyber-infrastructure	internet
motion sensor	motion detector	motion detection

Table 3: Example of incorrect disambiguation

A plausible evaluation method may be to wikify a set of documents and base the evaluation on a crowd-sourcing platform such as Mechanical Turk. The results of the wikifier may be compared against the results of the crowd-sourced wikification for the same set of documents. Again, the human element cannot be overcome in this approach either.

6 Conclusion

We see that the disambiguation engine gives an accuracy of 90%. We use a Jaccard similarity based measure for relatedness, and use unambiguous links as proxy context markers for the document. Improvement in this performance will probably require a more robust measure for relatedness. The overall accuracy for the WIkifier is in the neighborhood of 65% when evaluated against Wikipedia. Some applications may require recall to be high. An example is wikifying a news dataset to identify what entities are being referred to. In other applications, precision might be important, for example if are to suggest top-5 links to a blogger to annotate their blog. The tuning parameter k allows the user to vary precision and recall as required by their application.

References

[1] Mihalcea, R., & Csomai, A. (2007). Wikify! Linking Documents to Encyclopedic Knowledge.

[2] Milne, D., & Witten, I. H. (2008). Learning to link with wikipedia. In Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08).

[3] Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. 2009. Mining meaning from Wikipedia. Int. J. Hum.-Comput. Stud. 67, 9 (September 2009)

[4] Gardner, J. (2009). Automatic Link Detection : A Sequence Labeling Approach, 03.

[5] http://en.wikipedia.org/wiki/Wikipedia:Database_download

[6] http://en.wikipedia.org/wiki/Wikipedia:Statistics

[7] http://googleresearch.blogspot.in/2013/03/learning-from-big-data-40-million.html

[8] http://www.iesl.cs.umass.edu/data/wiki-links