Utilizing Wikipedia Categories for Document Classification

Timothy Weale
Department of Computer Science and Engineering
weale@cse.ohio-state.edu

Abstract

This paper introduces our technique for integrating Wikipedia as a broad-coverage knowledge base for use in document classification. We outline an algorithm for integrating the Wikipedia categories found from named entities in the articles. We then demonstrate this algorithm on a toy corpus, where we are able to successfully classify our documents.

1 Introduction

Traditionally, document classification has been based on a variation of the "bag of words"(BOW) approach. BOW treats all the words in a document as elements in a classification vector. These words can be a boolean decision (found or not) or weighted based on some measure (usually Term Frequency-Inverse Document Frequency. A new document is judged to be a member of a class based on how similar it is to the training documents.

The underlying assumption to this classification method is that similar documents will exhibit similar words. More advanced clustering methods (SVMs) and applying linguistic information (stopword removal, stemming, pos-tagging) can get better results, but these approaches are all more-or-less limited to the surface forms of the documents.

Document classification is not about the words – classes are organized around common concepts. That related documents will have similar word occurrences is merely a byproduct of the related concepts.

Gabrilovich (GM05) highlights this idea by using document #15264 in the Reuters-21578 corpus, belonging to the category "copper". This is a long article, discussing other mining activities and information about the companies involved in mining. Copper is mentioned only briefly. Thus, the article is typically mis-classified when it considers only surface words. If one were able to capture the common concepts behind the words, they should be able to more successfully classify the article.

A more specific problem is that the BOW approach oftentimes overlooks the information contained in named entities. Named entities carry a lot of informational content about the concepts in a document. Going back to the copper example, if the article mentions *BHP Billiton* and European Minerals (both mining companies) several times, then we should have greater evidence that this is an article about mining, rather than about a particular company due to the overlap in concepts among the entities mentioned.

Taken more generally, an article that discusses *Angelina Jolie* is probably different than one that discusses the *Cleveland Browns* in ways beyond the surface words in the document. However, if we want to make an even finer distinction, say among closely related concepts (whose surface words will overlap), then more information is needed. Ideally, documents focusing on one class of entities {*Harrison Ford*, *Brad Pitt*, *Julia Roberts*} (all actors) should be distinguishable from classes of documents that discuss other, closely related topics {*Steven Spielberg*, *George Lucas*, *Peter Jackson*} (all directors).

Up until recently, there hasn't been an available resource to capture the underlying information conveyed by these named entities. Hand-crafted resources (WordNet, CYC, OpenMind) were expensive to construct and limited in scope. The development of Wikipedia has, however, given us a cheap, broad-scoped resource to capture some of the concepts associated with these entities. In this paper, we propose to use Wikipedia to extract the hidden information in entities for more accurate classification than would be able if we used surface features alone.

The organization of our paper is as follows: In Section 2, we discuss similar attempts to use Wikipedia as a knowledge source. We outline our algorithm in Section 3 before introducing our data set and evaluating our algorithm's performance in Section 4. We conclude in Section 5 and discuss future work in this area.

2 Prior Work

Wikipedia has just begun to be used as a supplemental text resource, similar to WordNet or FrameNet.

Wikipedia has been used to both compete with and also bootstrap WordNet. In their 2006 paper (SP06), Strube and Ponzetto use Wikipedia to extract semantic relatedness between pairs of words. To do this, they utilize Wikipedia categories to

compute path based and text overlap based measures of relatedness between two words. In the end, the resulting relatedness measures are comparable to WordNet. This work has been extended by Gabrilovich and Markovitch (GM07) to process semantic similarity among non-title text, text longer than one word and with a more sophisticated similarity vector. On the other end, Wikipedia has also been used to effectively extract semantic relationships in order to extend WordNet (RCAC05).

Wikipedia has been used to train gazetteers for named entity recognition (TM06). Toral and Munoz utilize Wikipedia to speed up gazetteer creation, which can be a long and tedious process. Additionally, they observe that since their algorithm is language independent, additional languages can be added by simply running their algorithm over the appropriate Wikipedia language set.

Additional Wikipedia research has looked at its internal link structure. In their 2005 paper, Adafre and de Rijke (AdR05), they tackle the problem of missing inter-article hyperlinks. To do this, they propose a two-step ranking mechanism, LTRank, in order to cluster similar pages (which should have similar link structures) and then use these clusters to identify missing links among the documents.

Finally, this work is closest in manner to that of Gabrilovich and Markovitch (GM06). In their 2006 paper, they demonstrate a system for text categorization and use Wikipedia to overcome the bottleneck generated by limitations of the Open Directory Project. Their system uses a multi-resolution feature generator to classify documents using information at the word, sentence, paragraph and document level. Their final classifications are significantly better than previously published results.

3 Feature Construction

Our features are constructed by extracting all named entities from our corpus. These entities are then matched (if possible) to a Wikipedia article, where the category information is used to generate the document features.

3.1 Named Entity Extraction

As we have argued before, one of the major strengths of Wikipedia is that it contains information about a specific entity in the world, not available through other resources. Therefore, the first step in our algorithm is to extract entities from the documents.

To do this, we run an named entity recognizer over each document in the corpus and extract the named entities to be used for article matching. The final output of this step is a list of entities found in each individual document.

Bill Belichick
1952 births, Baltimore Colts coaches, Cleveland Browns coaches, Denver Broncos coaches, Detroit Lions coaches, Living people, New England Patriots coaches, New York Giants coaches, Phillips Academy alumni, Croatian-Americans, The NFL on ABC
AFC North
National Football League, Baltimore Ravens, Cincinnati Bengals, Cleveland Browns, Jacksonville Jaguars, Pittsburgh Steelers, Tennessee Titans, 2002 establishments

Figure 1: Categories for given entities

3.2 Article Matching

We then extract category information from the article about the named entity. This category information forms the basis for our classification vector.

Unlike other categorization resources (such as the Open Directory Project), Wikipedia categories are not rigidly assigned to an overall hierarchy. So, an entity can belong to multiple categories, each found within its own hierarchy and corresponding to a different piece of information about the entity. For example, "Bill Belichick" belongs to categories relating to his high school, career as an NFL coach and birth year. This allows us to capture a wide variety of high-level information about an entity cheaply.

In our implementation, entries that returned zero queries were discarded, as they were usually misnamed entities. If a disambiguation page was reached, the named entity was also discarded. This is a limitation of our current system, and will be discussed in Section 5.

At the end of this step, we have a list of categories present in the document. These will then be turned into our vector for classification.

4 Evaluation

The evaluation was done with the Wikipedia data dump from November 11, 2006[1]. After decompression, the resulting XML file was 6.6GB in size. We used MediaWiki's import routines to populate a MySQL database, which we used to process our named entity queries.

[1] http://download.wikimedia.org/

4.1 Data Set

The evaluation of our method was done on a toy corpus consisting of 40 training documents(30 positive / 10 negative) and 4 testing documents(2 positive / 2 negative) gathered from Google news. We have a very hard corpus – all documents pertain to the 2006 NFL season and therefore share many words and have a high concept overlap. Positive examples are those in which the article mentions a current or former coach of the Cleveland Browns. Negative examples are those that fail to mention a Browns coach.

4.2 Category Discovery

We ran the LingPipe Named Entity Recognizer[2] over our data set in order to get our named entities, an off-the-shelf named entity recognizer. The named entity model was obtained from their MUC-6 training model.

After running our named entity recognizer over the dataset, we noticed that some of the training documents failed to capture the desired entities. We then reconfigured our training examples based on the results of the entity recognition process, resulting in a 28 positive / 12 negative training split.

The resulting named entities were queried against our MySQL database containing the Wikipedia entries. In the case of multiple result queries, only the highest-numbered result query was used (corresponding to the latest version of the topic).

The resulting classification vectors are based on all categories found somewhere in our training set (rather than all possible categories). Any additional categories found in the testing set are discarded. Since the number of categories is enumerable through a search of the Wikipedia dataset, we may wish to expand our vector to include all possible categories in the future.

4.3 Classifiers

In our evaluation, we decided to try two different types of classifiers: Support Vector Machines and Decision Trees.

SVMs are linear classifiers that attempt to split the input space into a hyperplane delineating all positive examples from negative examples. Vectors are considered as a whole. In SVMs, we are looking for hyperplanes with a maximum margin between the positive and negative examples. For our evaluation, we utilize the SVM^{light} classifier[3] and the C4.5 decision tree algorithm[4].

[2]http://www.alias-i.com/lingpipe/

[3]http://svmlight.joachims.org/

[4]http://www.rulequest.com/Personal/

Decision trees are models that make incremental checks on vector elements to decide the class of an object. The algorithm finds the most distinguishing element from the vector and splits based on that element. Then, it recursively finds the most distinguishing element of its subsets, until a sufficient level of generalization is found. Decision trees have many benefits – the results are easy to interpret, easily human-reproducible and can provide generalizations with little hard data. However, they can be prone to over-fitting. We use the C4.5 decision tree algorithm, which is based on the ID3 decision tree generator.

4.4 Results

In our experiment, we compared the output of three things: an un-optimized bag-of-words model, a bag-of-named-entities model and bag-of-categories. All input vectors consisted of boolean values – either the item was found in the document, or it wasn't.

	Word	Entities	Concepts
SVM	50%	50%	75%
C4.5	–	50%	100%

Table 1: Classification Accuracy

As expected, the BOW classification and the named entity classification struggle with the concept overlap in the data set. In all of these cases, the classifiers tried to include all the testing examples.

With the category-based SVM vectors, the classifier was able to disregard one of the testing examples. Our negative example set was simply too small for it to pick out the appropriate features for classification. With more negative training examples, we should see improved performance from the SVM.

C4.5 was able to distinguish the appropriate category with ease and generated a one-level decision tree. If "Cleveland Browns coaches" category was found, then the document was classified as a positive example. This is expected, given the corpus involved.

5 Conclusions and Future Work

In this paper, we have introduced a method for document categorization using Wikipedia as a large-scale knowledge base for information about named entities. We have also demonstrated this algorithm on a toy dataset, where it successfully performs the expected categorization.

There is a lot of work to be done in this domain. Our first step is to rebuild the Wikipedia database. While MediaWiki was able to successfully populate our MySQL database, the provided database

New England	Browns	Green Bay Packers	Packers	Bengals
Patriots	*Romeo Crennel*	Green Bay	Green Bay	Marvin Lewis
Bill Belichick	Charlie Frye	Matt Hasselbeck	*Forrest Gregg*	Cleveland
Tom Brady	Terry Pluto	Brett Favre	Milwaukee	Chris Perry
Chicago	Georgia Dome	Seattle	William Perry	Browns
Train 2 (+)	**Train 20** (+)	**Train 36** (-)	**Test 1** (+)	**Test 3** (-)

Figure 2: Example Named Entities from Corpus

structure is not optimal for our needs. Article links, Wikipedia categories and other pertinent information should be pre-computed and available within our SQL queries.

Disambiguation needs to be handled. In our current system, we discard entities that need disambiguation. Obviously, this needs to be handled. Perhaps a two-pass method that gathers all non-ambiguous entities first, and then does disambiguation based on the article text and the disambiguated article pages.

Additionally, we wish to run our algorithm on additional data sources. Our toy corpus was used to demonstrate the initial concept, but we need to see how this might work in a more general classification context. To this end, we have obtained the RCV1 corpus, and will be running experiments to see how our algorithm works on this corpus.

Finally, we should investigate additional weighting algorithms. The boolean classifier was sufficient for the toy corpus, but established methods such as TF/IDF may show some improvement, especially on a corpus such as RCV1.

Further out, there is a lot of potential work. Article consistency needs to be checked to ensure that appropriate inter-article links are maintained. For example, the article on "Bill Belichick" lists him as a living person. Yet, the article for the current Browns coach (as of this writing) "Romeo Crennel" does not. Such consistency needs to be maintained. Then, we should be able to use the link structure to find a the relatedness of different articles, based on what we have seen before. Also, we would like to implement the Gabrilovich and Markovitch (2006) algorithm, which would provide us with the closest comparison to state-of-the-art in this field.

References

Sisay Fissaha Adafre and Maarten de Rijke. Discovering missing links in wikipedia. In *Link-KDD*, Chicago, Illinois (USA), August 2005.

Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *The 19^{th} International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1048–1053, Edinburgh, Scotland, UK, August 2005.

Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of The 21st National Conference on Artificial Intelligence (AAAI)*, pages 1301–1306, Boston, July 2006.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of The 20th International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, January 2007.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In *NLDB*, number 3513 in Lecture Notes in Computer Science, pages 67–79, 2005.

Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *AAAI*, 2006.

Antonio Toral and Rafael Munoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *Workshop on New Text, 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento (Italy), April 2006.