Using Network Structure to Learn Category Classification in Wikipedia

Caitlin Colgrove, Julia Neidert, Rowan Chakoumakos

December 11, 2011

1 Introduction

Wikipedia, founded in 2001, is a massive, collaboratively-edited online encyclopedia. As of November 2011, the site has more than 20 million articles available in 282 different languages. The site is maintained by a vibrant community of editors, who supervise the editing process to ensure quality and consistency.

The body of a Wikipedia article consists primarily of text, images, and hyperlinks to other articles. In addition to article content, each page is also placed into various relevant categories of articles. The editors of Wikipedia maintain this category hierarchy, manually labeling articles with the most appropriate categories. These categories are often used as the gold standard for semantic NLP problems, such as finding document topics [9]. Categories can also be useful in navigation of Wikipedia, whether simply finding related articles or attempting to find longer paths through the network.

However, as Wikipedia continues to grow, manually labeling categories becomes more and more difficult. With over 80,000 categories, the monumental task requires its own Uncategorized Task Force of Wikipedia editors, whose goal is to categorize uncategorized articles. Even so, many articles remain uncategorized or undercategorized.

Fortunately, many articles in Wikipedia have a wealth of information, from the content of the text to the hyperlink network. These features suggest that an automatic classifier of Wikipedia may be a tractable problem. A classifier with reasonable accuracy could greatly lighten the load of the task force, improve the coverage of articles, and possibly identify and correct miscategorized articles.

2 Prior Work

Previous work in categorizing Wikipedia articles has focused on NLP to automatically classify categories [4]. In the paper Automatic content-based categorization of Wikipedia articles, Zeno Gatner and Lars Schmidt-Thieme were able to achieve a F-measure on predicting categories between 0.55 and 0.8 using NLP features. Assigning categories, however, does not have to be only based on NLP, but instead can incorporate features derived from the structure and properties of the underlying network. Qing Lu and Lise Getoor, in their paper Link-based Classification, saw a statistically significant increase in performance on Cora, CiteSeer, and WebKB datasets when network features such as the degree distribution were used rather than just content features [7].

Network based features have also been successfully applied to the Wikipedia network [3, 8]. In Predicting Crosslingual Link Prediction, Phillip Sorg and Phillipp Cimian, were able to predict crosslingual links by using mainly network derived features (links that connect concepts across different language versions of Wikipedia) with recall of around 70% and precision of around 94% [8]. Therefore, we believe that predicting category links based on only network derived features may result in an increase performance over previous NLP attempts, or at the very least increase in performance when used in combination with NLP.

Other work has been done with the Wikipedia category graph. In Analyzing and Visualizing the Semantic Coverage of Wikipedia and Its Authors, Holloway et al. analyzed the category graph and noted that it contains cycles, is disconnected, and based on the edit history is kept up to date [5]. Holloway et al. created super categories to deal with the unwieldy nature of the Wikipedia category. Given that many categories may be small, using supercategories for prediction may be more accurate.

3 Data

We are using data from the November 7th, 2011 dump of Wikipedia. Our data includes the mapping from pages to the links on the page (this includes both article links and other types of links like categories) and a mapping from category name to category id. After downloading the initial dumps, we converted the SQL of the dumps into a simple CSV format and imported this into a MongoDB database hosted on Amazon EC2. In order to achieve reasonable database performance to run experiments, we had to use eight EBS drives in RAID 0 formatted with the XFS file system. MongoDB allows us to store an arbitrary number of calculated features for each Wikipedia page, which enabled us to easily build the network.

4 Model

We choose to treat Wikipedia as a network, with pages as nodes and hyperlinks as edges. Other research [15] has found that the clustering coefficients of the Wikipedia networks are considerably higher than the expected clustering coefficient of a random graph of the same degree distribution. This suggests that the Wikipedia network exhibits community structure. We exploit this property by hypothesising that these communities correspond to categories. We thus use features dependent on the pages close by in the graph, presumably in the same category community, to classify Wikipedia pages.

4.1 In-links and Out-links

4.1.1 Basic Algorithm

In order to directly take the community structure of the Wikipedia network into account, we consider the count and portion of neighboring pages in a given category. We consider both the number of pages that the page in question contains hyperlinks to (out-links) and the number of pages with hyperlinks to the page in question (in-links). When classifying a page as a member of a particular category, then, we have four features:

- the count of out-links from the page to pages of the category
- the portion of such pages out of all out-links
- the count of in-links to the page from pages of the category
- the portion of such pages out of all in-links

We further look at these attributes for different levels out from the page in question. A page is in level n of a page if it can be reached by following exactly n links from that page. For a particular page and level n we have four additional features:

- the count of the paths of length n from the page that lead to pages in the category
- the portion of such paths out of all possible paths of length n coming from the page
- the count of the paths of length n to the page that come from pages in the category
- the portion of such paths out of all possible paths of length n leading to the page

This allows us to consider pages further from the page in question that may also be in the category community. In addition, pages that are more connected to the page are given more weight. These features are extracted simply by following out-links and in-links of the graph and maintaining counts of all pages reached and counts of pages reached in a particular category.

4.1.2 Heuristic

Considering levels further and further away from a page becomes more and more computationally expensive. In fact, the increase is exponential, and with on average hundreds of links from each page, this is not a trivial exercise. Thus, in order to compute features for far out levels, we use a heuristic that gives a more equal weighting to all pages of a particular level, reducing the preference for more connected pages. To do this, we initially find and store all pages of distance 1 away from a page in a particular category (pages with a hyperlink to a page in the category or linked to from a page in the category). We then use this set of pages to find the counts of linked pages from one level away. That is, we estimate the number of paths to pages in the category at level n by getting the count of all pages linked to by pages in level $n-2$ that are in the set of pages of distance 1 from the category. This does not count duplicate paths from pages in level $n-2$ to pages in level n, and also does not disambiguate between in-links and out-links at the last link, but nevertheless provides a useful heuristic for find features involving a wider span of pages.

4.2 HITS

4.2.1 Basic Algorithm

The HITS algorithm was originally developed to model the fact that in a search for information on the Internet, there are two distinct types of valuable pages: authorities, which contain the desired information, and hubs, which are lists of authorities. The quality of hubs and authorities are defined in a mutually recursive manner: a good hub points to a lot of good authorities and a good authority is pointed to by a lot of hubs. In fact, the notion of authorities lines up quite well with what we are trying to detect: good examples of pages in a particular category. Thus, the feature we select from this algorithm is the authority score for a node.

The simplest version of HITS initializes a hub score $h_i = 1$ and an authority score $a_i = 1$ for each node i. Additionally, the algorithm computes the adjacency matrix M, where $M_{ij} = 1$ if there is a directed link from node i to node j. The algorithm proceeds as follows:

Repeat until convergence:

1. Update $h = Ma$
2. Update $a = M^T h$
3. Normalize to $\sum_i a_i = 1$, $\sum_i h_i = 1$

HITS is a good model for Wikipedia because of frequent hub pages (for example, any List of... article). Additionally, HITS captures weighting of edges in a way that simple counting does not. For example, a page for a movie might be a good hub for the Actors category, so a link from this movie to a page has a lot of weight, even though the movie is not even in the category being examined.

4.2.2 Modification

As presented above, HITS has a few drawbacks. The principal problem is that if the initial graph is chosen to be all of Wikipedia, the algorithm will output the hubs and authorities for all of Wikipedia, not for a topic specific subset. Additionally, because Markov models converge to a single fixed point regardless of their origin, there is no way to initialize the matrix M to avoid this. However, in his discussion of HITS for web-search, Kleinberg proposes a modification to produce a topic-sensitive subgraph [16]. First, he finds the set of documents that match a query. Presumably, the vast majority of good authorities are located within this set. Consequently, it is highly likely that any good hub points to something in this set. Thus, to obtain a topic-specific subgraph, he takes the graph formed by the queried documents and all the documents that have a direct link to or from this set.

Analogously, we create a topic-sensitive subgraph of Wikipedia. To do this, we take all the members of a category (minus the members we will be testing on) and include every page that links to or from one of these pages. This increases the likelihood that pages important to that topic will have many links within the graph, while pages peripheral to the topic will not.

This suggests a naive approach to calculating the HITS scores for a node.

Repeat for all nodes i:

1. Add i and all of its links to the current graph
2. Calculate the HITS scores for the graph
3. Report h_i, a_i
4. Remove node i

However, this produces a second, purely a technical, problem. The subgraph of actors plus their direct connections is around 500,000 nodes, taking about 15 minutes per HITS computation. This makes it infeasible to compute these features on more than a few nodes. With datasets up to 1600 entries in size, we could not afford to run such an expensive computation for each node.

To avoid this, we make the following assumption: if HITS has iterated to convergence on a graph, the addition of a single node will not significantly change the already computed HITS scores. Thus, to approximate the HITS authority scores, we calculate HITS once and cache the hub scores. We then calculate an approximate authority score for each node by taking a simple sum over the hub scores of the nodes that link to it.

4.3 Classification

We ran several different learning algorithms on our feature sets: Naive Bayes, Logistic Regression, and SVMs. We also ran parameter selection algorithms to optimize performance of each learning algorithm, as well as feature selection algorithms in order to find the most significant features.

5 Results

6 Discussion

7 Future Work

8 References