Automated Categorization of Wikipedia Pages

Jean Feng JJFENG@STANFORD.EDU
Chuan Yu Foo CYFOO@STANFORD.EDU
Yifan Mai MAIYIFAN@STANFORD.EDU
(Group #31)

1. Introduction

The automatic categorization of documents is often approached from a natural language processing (NLP) perspective in which both linguistic and non-linguistic features specific to each document are used as inputs to a classifier which categorizes each document in isolation (e.g. see [1]). However, this approach neglects the fact that in many instances, these documents are related to each other in some way – for instance through citations, references or hyperlinks – and these relations can provide useful information about the categories of each of the documents. For instance, a set of documents that all reference one another may be more likely to be of the same category than a set of documents that reference disjoint sets of other documents. As such, using *network features* – features that pertain to a document, the documents it links to, and the relations between them – may improve classification performance.

In this paper, we consider using network features for the categorization of Wikipedia articles. Wikipedia is a free online encyclopedia with over 4 million articles (as of November 2012). Articles in Wikipedia can belong to any subset of 800 000 user-created categories, but currently the categorization of articles is done manually on an article-by-article basis by editors who are involved in many other tasks such as producing and proofreading content. This manual categorization of articles is a labour-intensive and error-prone task, and with the number of articles growing at a steady pace of about 1% a month, many articles remain uncategorized, and even more may be incompletely or inappropriately categorized.

2. Prior Work

While there has been previous work on the automatic categorization of Wikipedia articles, most approaches have focused on using natural language features of the articles such as the article text and article title (e.g. [1]). However, in other applications, the use of network features has been shown to significantly improve classification performance. For instance, in [2], logistic regression was used to the classify articles in the Cora and WebKB datasets (which contain scientific publications and their associated citation networks). It was found that compared to using natural language features alone, classification accuracy was significantly improved when simple network features, such as the number of neighbors of each category a document has, were used in addition to natural language features.

Later work using network features for related problems also demonstrated the effectiveness of network features that capture more of the network structure. For instance, in the study of the related problem of link type prediction, [4] used conditional random fields to predict the link type between web pages in a network. It was demonstrated that by adding factors that captured network characteristics – such as factors that suggest that different pages that are linked to by the same page should be linked by similar link types; or factors that suggest that pages that are linked by multiple paths should have similar link types along all such paths – classification performance was better than when only natural language features of the pages were used.

3. Data and Preprocessing

We used data from both the DBPedia database. DBPedia is a semi-automatically generated semantic knowledge base which models Wikipedia pages and the relations between them. In addition to Wikipedia links and categorizations, the database also includes additional semantic relations such as *is-an-actor-in* or *is-married-to* automatically generated from Wikipedia or related datasets.

The data was preprocessed as follows – we first picked two main topics to focus on – movies and companies. For each topic, we extracted all Wikipedia pages on that topic by filtering on DBPedia categories that indicated that the page belonged to the topic. Next, the

DBPedia hierarchy of Wikipedia categories was used to manually select a small set of basic topic categories. For instance, for movies, we selected the basic topic categories Comedy Films, Documentary Films, Historical Films, and so on.

The Wikipedia pages were then labelled with the appropriate basic topic category labels as follows: suppose a page has topic categories $\{w_1, w_2, \ldots, w_n\}$. For each topic category w_i, locate w_i in the hierarchy of categories, and traverse up the tree until either one of the basic topic categories is found or we reach the root. If a basic topic category is found, we add the label for that basic topic category to the page.

4. Graph Model and Evaluation

For each topic, we constructed a separate directed graph as follows: for each Wikipedia page we construct a vertex, and we construct a directed edge from a vertex to another vertex if there is a topic-related relation from the source page to the target page. For instance, for the movies topic, topic-related relations included relational links such as actedIn or directedBy. For convenience, we will henceforth refer to vertices for Wikipedia pages on the topic as *topic nodes* and vertices for other pages as *non-topic nodes*.

The classification task was to predict whether a node belongs to a particular category using the features of that node alone. For all three of the Logistic Regression (LR), Markov Random Field (MRF) and Support Vector Machine (SVM) models we describe in the following sections, the same protocol was used. The protocol is as follows: for each topic T, we extracted a subset of the graph for the topic to yield a graph G_T for the topic. For each category C, we train a separate model M_C to classify whether a node belongs to category C. To train the model, we build train and test graphs as follows. From G_T, we selected a random subset of 10% of the topic nodes. These nodes were removed from the graph to give a training graph. We then computed the features for the topic nodes in the training graph and trained the model on the training graph. To test the model, we then reintroduced the removed nodes to the training graph, and computed the features for these nodes. The trained model was then used to predict whether the node belongs category C, and the accuracy of the model was evaluated against the known labels.

5. Models

5.1. Baseline: Logistic Regression on Text

To demonstrate the usefulness of network features and to provide a baseline for comparison, we trained a model which used only the text of the pages for classification. For this model, we extracted a vocabulary of around 5000 of the most frequently encountered words from the text of the Wikipedia pages on the topic. Following this, for each page we created a text feature vector with a feature for each word which was equal to 1 if the word was present in the page and 0 otherwise. These features were then used to train a logistic regression model on the training data. This model was then used to predict the categories of the test pages. Some representative results are shown in table 1 below.

Topic	Category	Prec.	Recall	Acc.	F1
Movies	Drama	0.355	0.417	0.594	0.384
Comp.	Enter.	0.460	0.537	0.694	0.496

Table 1. Logistic regression using text features classification results for two representative subcategories – Drama Movies and Entertainment Companies

5.2. Markov Random Field Model

Inspired by the work in [4], we tried using a simple Markov Random Field (MRF) for the classification task. For each category, we built an MRF as follows – for each topic node in the graph, create a random variable C_i which indicates whether the topic node belongs to the category.

To capture the prior probability that a topic nodes belongs to the category, we define a template singleton factor $\phi(A)$ over a single topic node. To capture the notion that nodes that are related should have similar categories, we define a template pairwise factor $\psi(A, B)$ over pairs of topic nodes. Since similarity should be symmetric, we restricted the values of ψ such that $\psi(a, b) = \psi(b, a)$.

Using these templates, we constructed an MRF as follows: for each topic node C_i, create a factor $\phi_i(C_i)$. For each pair of topic nodes C_i and C_j, if they share any neighbours, create a factor $\psi_{i,j}(C_i, C_j)$. Note that since this is a template model, all the generated ϕ factors and all the generated ψ factors use the same parameters in their conditional probability table. By taking the factor product of all the factors and normalizing, we obtain the probability distribution over assignments to all the C_i in the graph. This can then be used to find the most likely categorization for the topic nodes.

The model was parametrized using four parameters: $\phi(0), \phi(1)$ and $\psi(0,0) = \psi(1,1)$ and $\psi(0,1) = \psi(1,0)$. Since the structure of the MRF is known and the data is complete, by imposing an additional regularizing constraint that both ϕ and ψ define a probability distribution, we can learn the parameters efficiently by doing maximum-likelihood estimation using the training data.

Once the parameters have been learned, a similar MRF can be created for the test data. As per the protocol, we revealed 90% of the labels in the test data, and attempted to predict the labels for the remaining 10%. Due to the large size of the dataset, exact inference was not possible. As such, we used Gibbs sampling to sample a large number of samples from the distribution, and to estimate the most likely assignment by taking the most frequently sampled assignment. Some representative results for are shown in table 2 below.

Topic	Category	Prec.	Recall	Acc.	F1
Movies	Drama	0.526	0.465	0.695	0.494
Comp.	Enter.	0.430	0.406	0.678	0.417

Table 2. Markov Random Field (MRF) classification results for two representative subcategories – Drama Movies and Entertainment Movies

Analysis of the results suggested that, as expected, the MRF tended to classify groups of nearby nodes as belonging to the same category. This resulted in an improvement of the text features as the graphs did indeed have the property that nearby nodes were more likely to belong to the same category, and the MRF was able to take advantage of this property. This is consistent with previous findings that using even simple network features can result in significant improvements in categorizing nodes.

We attempted to further improve the model by expanding the pairwise factor to depend on the number of neighbours shared by the two nodes as well, that is, by replacing the factor $\psi(A, B)$ by a factor $\psi(A, B, X)$, where A, B are topic nodes and X is the number of neighbours shared. However, this did not improve the classification results significantly, probably due to the fact that there were few topic nodes that shared more than 1 neighbour.

We briefly considered further improvements to the model such as adding another pairwise factor to capture longer range dependencies (for instance, if nodes A and B are connected and nodes B and C are connected, then we could introduce a pairwise factor between nodes A and C). However, as it was difficult to make predictions with the MRF, and as the Support Vector Machine (SVM) model (described in the next section) was producing more promising results, we decided to focus on that instead.

5.3. Support Vector Machine Model

In contrast to the MRF model described in the previous section, we also tried a simpler Support Vector Machine (SVM) model, partly inspired by the work in [2] and [5]. For this model, we compute a number of network features (described in more detail below) for each node, and concatenate these features to form a single feature vector. We then train an SVM on the training graph, and use this trained model to predict the categories of nodes in the test graph.

Our hypothesis is that nodes of some category C will tend to have many short paths to topic nodes of the same category C. For example, in our movie data, nodes in the graph include movies and people, such as directors and actors. Since one would expect that a director of a horror film is likely to direct another horror film, one would expect to have many 2-edge paths starting at a horror movie node h to the node representing the movies director and ending at other horror movie nodes. To test this hypothesis, we tried two different types of features – k-hop genre score and random walk features, of which random walk features performed the best. These features are described in more detail below.

The first type of feature we tried is a direct mathematical translation of our hypothesis. The feature, which we termed k-Hop Genre Score, is defined as follows: given a node n and category C, let $P_k(n)$ be the set of k-hop paths starting from v ending at a topic node and let $P_k(n, C)$ be the paths in $P_k(n)$ that end at a topic node belonging to category C. The k-Hop Genre Score of a node n for the category C is $\frac{|P_k(n,C)|}{|P_k(n)|}$.

We therefore expected that the k-Hop Genre Score for a node n to be high for the categories the node belonged to and low for the categories that were irrelevant. Using dynamic programming, one can compute the k-hop genre scores efficiently. Computing the scores for all the nodes in the graph takes $O((|E| + |V|)k)$ time. For these results, we concatenated the random walk feature vectors for $k = 1, 2, \ldots, 5$. The results are as follows:

As visible from the table, the k-hop genre score does not perform very well, with accuracy and F-measure score similar to that of the MRF model. One problem with this feature is that nodes are weighted equally, regardless of their distance from the source node. This is contrary to the intuition that nodes further away

Topic	Category	Prec.	Recall	Acc.	F1
Movie	Drama	0.344	0.742	0.593	0.470
Comp.	Enter.	0.412	0.822	0.833	0.549

Table 3. SVM classification results for k-hop Genre Score features for two representative subcategories – Drama Movies and Entertainment Companies

should have less "influence" that nodes that are closer by.

To overcome this problem, we designed a second set of features based on the random walk algorithm. A k-hop random walk starts from a node n and travels through k edges. At each node, the algorithm chooses an edge connected to the current node uniformly at random, and follows it to the next node. We define the random walk score of node n for a category C to be the probability a k-hop random walk ends in at a node of that category given that it ends in a topic node.

Similar to that in k-hop genre score, we expect the random walk score of a node to be high for the categories the node actually belongs to and low for the categories the node does not belong to. In addition, random walk features can also be computed efficiently using dynamic programming with a runtime of $O((|E|+|V|)k)$. Results for this feature are discussed in the section below.

6. Discussion

Our best classification results for the graphs on company and movie DBpedia data were from the SVM trained on the concatenated random walk feature vectors for paths of length $k = 1, 2, \ldots, 5$. The final performance values are given in Table 4.

Topic	Category	Prec.	Recall	Acc.	F1
Movie	Drama	0.481	0.552	0.746	0.513
Comp.	Enter.	0.830	0.857	0.961	0.843

Table 4. SVM classification results for random walk features for two representative subcategories – Drama Movies and Entertainment Companies

These results are significantly better than those from MRF and the SVM trained on k-hop genre score feature vectors. For comparison, representative results for all four models are shown in table 5.

One reason why the random walk scores probably did well was that the random walk scores of the correct categories of a node were usually significantly higher than the random walk scores for the incorrect categories.

To illustrate this, we plotted the random walk scores for all the nodes belonging to a certain category. In Figure 2, we show the distributions of the random walk scores for nodes belonging to the category Entertainment in the company dataset. The sorted random walk scores for entertainment from highest to lowest is shown in blue and the sorted random walk scores for the other categories are shown in red. In this graph, the blue line is above the red lines, which suggests that the random walk scores for the correct category, Entertainment, are higher than the random walk scores for the incorrect categories. Many of the other categories in the company and movie dataset also show a similar trend. These results confirm our hypothesis that nodes near a given node n appear in higher concentration, which our algorithm measures via the random walk score.

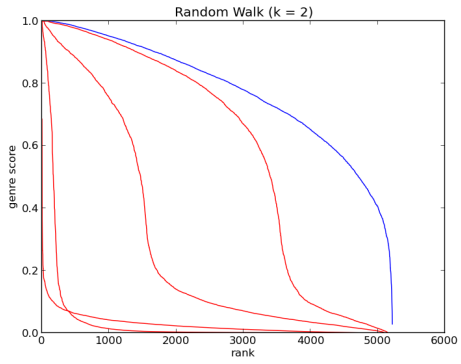

Figure 1. Lines in the plot are the distributions of random walk scores for nodes belonging to the Entertainment category in the Company data for $k = 2$. The difference in the distribution suggest that the random walk score is useful in distinguishing the category a node belongs to.

To better understand the random walk features, we also tested how performance varied with the value k. From figure 2, it can be seen that $k = 2$ gives the best performance results, with $k = 4$ doing next best. The fact that the even k values give better results than all of the odd k values actually illustrate an interesting property of the particular datasets, if not DB-

Model	Prec.	Recall	Acc.	F1
Text Baseline	0.460	0.537	0.694	0.496
MRF	0.430	0.406	0.678	0.417
SVM: Genre score	0.412	0.822	0.833	0.549
SVM: Random walk	0.830	0.857	0.961	0.843

Table 5. Results for all four models on Entertainment Companies

Pedia data in general. The performances peaking at even values of k most likely results from the fact that topic nodes tend to be separated by an even number of edges. Because topic nodes carry category information, they also give the most relevant information for calculating features.

For a more concrete example, consider the movie graph. The nodes one edge away are mostly people like actors and directors while there are many fewer links to other movies. Therefore the features from $k = 1$ will not have as much as $k = 2$ features, which mostly contain features generated from nodes that were reached via a movie-person-movie path. Similarly, $k = 4$ features are mostly based on movie nodes reached via a movie-person-movie-person path. However, this property might be particular to the company and movie graphs. For categories with nodes that tend to link to one another directly, instead of sharing nodes, it is likely that even k values will not have significantly better performances than the odd k values.

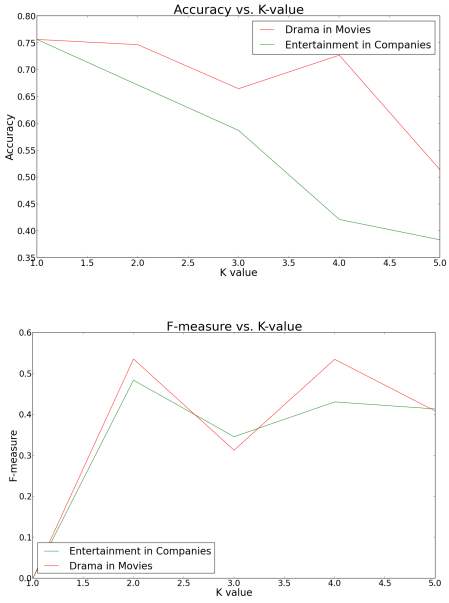

Figure 2. The accuracy (top) an F-measure (bottom) of the algorithm for random walk features are plotted with respect to different values of k. The even k values show the best performance.

The performance of the algorithm in classifying nodes under different categories varied depending on the network properties for nodes that belonged to that category. For the categories we tested, there was a positive correlation between the accuracy of the method with the average degree of the nodes in the graph, as shown in Figure 3. The figure shows that distribution companies had the best result of 96% and manufacturing companies had the worst result with 65%. At the same time, distribution company nodes had an average degree of 74 while manufacturing company nodes had an average degree of 15. This positive correlation between degree and accuracy is expected, as the random walk feature for a given node is calculated based on its neighboring nodes. If there are few nodes surrounding that node, then the probabilities of ending random walks in each category are heavily skewed.

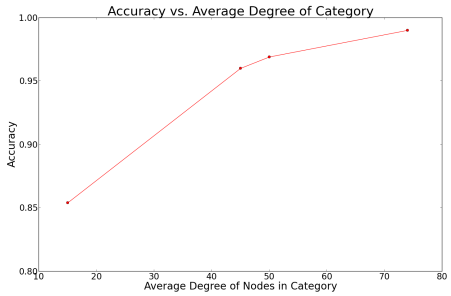

Figure 3. Categorization accuracy with respect to the average degree of the nodes in the category.

Finally, the difference in the overall performance of the algorithm over the movie and company graphs is also worth investigating, since ideally, we would want this algorithm to generalize over a wide range of subcategories. The large difference most likely stems from the quality of the data. Companies in general tend to have more information, and thereby more links, as compared to movies, which could be due to many reasons. Companies have a vested interested in maintaining their Wikipedia site, making sure to reference the correct people and affiliate companies. In addition, many companies produce a large number of products and because of their more long-lasting nature, can also be associated with more people. These products are then linked to other companies that might produce similar or complementary products. Therefore, we expect our model to perform better on categories with entities that are annotated more thoroughly.

7. Conclusion

In this paper, we presented a number of different methods for the automated categorization of Wikipedia articles using network features. The results show that applying a support vector machine model over random walk features gives the best performance for the articles from the categories Movies and Companies.

References

[1] Gantner, Z. & Schmidt-Thieme, L. (2009). Automatic content-based categorization of Wikipedia articles. *Proceedings of the 2009 Workshop on the Peoples Web Meets NLP*, pp. 32 - 37.

[2] Lu, Q. & Getoor, L. (2003). Link-based classification. *Proceedings of the Twentieth International Conference on Machine Learning.*

[3] Clauset, A., Moore, C. & Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks *Nature.*

[4] Taskar, B., Wong, M., Abbeel, P. & Koller, D. (2003). Link prediction in relational data *Advances in Neural Information Processing Systems* 17.

[5] Neville, J. & Jensen, D. (2000). Iterative classification in relational data *Proceedings of the AAAI 2000 Workshop Learning Statistical Models*, pp. 42 - 49.