Task Recommendation on Wikipedia

Eric Huang
Stanford University
ehhuang@stanford.edu

Hyung Jin Kim
Stanford University
evion@stanford.edu

Hajoon Ko
Stanford University
gogo9th@stanford.edu

ABSTRACT

In many open-source projects and user-generated-content websites, one challenge is matching contributors with tasks so that contributors are able to work on things on which they have the most expertise or interests, thus increasing their productivity. In this project, we study one particular domain, Wikipedia, an online collaborative encyclopedia. Our goal is to make finding articles on Wikipedia to work on easier for editors. We created an edit graph of Wikipedia, and formulate the problem as a link prediction problem. Using topological features, we applied and evaluated various machine learning algorithms on the data. We found that decision tree is the best performing algorithm, achieving 74% testing accuracy.

Keywords

Wikipedia, recommender systems, collaborative filtering, network analysis

1. INTRODUCTION

Recently, we have witnessed a shift on the internet from static content distribution to more and more dynamic user-generated contents. Coupled with this is an increase in the formations of online collaborative communities, open-source efforts, and crowdsourcing: Yahoo! Answers, Quora, Github, Amazon Mechanical Turk, and Wikipedia, to name a few. According to social science theory, reducing the cost of contribution has an effect on increasing people's motivation to participate. These online collaborative communities and efforts are made possible with the convenience of the internet. In this paper, we are interested in the domain of Wikipedia, a web-based collaborative encyclopedia. Particularly, we are interested in the question of how to reduce the cost of editors finding finding Wikipedia articles to improve. By reducing the cost of finding articles that align with the editors' interests, we can potentially improve the quality and quantity of the articles on Wikipedia. We aim to build a recommender system for suggesting Wikipedia articles to editors.

1.1 Related Work

Our work is inspired by SuggestBot[1], which is a software system that matches people with work on Wikipedia. It employs three approaches to the problem, using text analysis, collaborative filtering and hyperlink following. Their collaborative filtering algorithms uses the Jaccard coefficient to measure similarity between editors, and recommend by looking at the value of the Jaccard coefficient directly. Our approach employs many more topological features and uses machine learning algorithms.

In [2], the authors evaluated various link predictors on several social networks. In [4], the authors applied a link prediction approach to collaborative filtering. However, they make recommendations on each linkage measure separately, and solely on the value of the linkage measures. In [3], the authors also applied supervised learning methods to solve the link prediction problem, and studied the features. However, they used different datasets which are much smaller graphs in comparison with the Wikipedia network.

2. PROBLEM FORMULATION

We formulate our problem as a link prediction problem. We consider the entire Wikipedia network as a undirected bipartite graph, where editors and articles are nodes in the graph, and an edge between a particular editor-article pair represents that the editor had edited that article at some point in the past. To decide whether a particular article is a good candidate to recommend to some editor, we build a classifier and predict whether an edge is likely to form between that particular editor-article pair in the future. That is, whether the editor will edit that article some time in the future.

More formally, consider a bipartite graph G, with two sets of nodes, N_E and N_A, and a set of edges E, where an edge exists between some $n_{E,i} \in N_E$ and $n_{A,j} \in N_A$ if Editor i has edited Article j at some point in the past. For any given editor, $N_{E,i}$, we want to output a list of $N_{A,j}$'s that $N_{E,i}$ is likely to edit in the future.

3. APPROACH

We apply a supervised learning approach to our link prediction formulation of the problem. In short, we obtain training samples from the graph in the past, we extract topological features from the snapshots of the graph, and evaluate the trained models on the testing samples in the time

period following the training period. The idea is that we believe the topological features from the graph, which represents the interactive relationships between editors and articles, have predicting powers and are informative to whether a link will form between the two nodes in the future.

More formally, we partition our data into two sub-ranges, defined by four points in time, $T_1 < T_2 \leq T_3 < T_4$. The training samples are obtained from the first sub-range, $[T_1, T_2]$, and the testing samples are from the second sub-range $[T_3, T_4]$. The positive samples are the editor-article pairs that did not have an edge between them in T_1, but had an edge by T_2, meaning that the editor edited that particular article during this time frame. The negative samples are those that did not have an edge between the pair in both T_1 and T_2, representing that the editor did not edit that article. We train our models with samples from $[T_1, T_2]$. Then, we make predictions with our trained models on editor-article pairs in T_3. Finally, we evaluate our predictions by examining T_4.

3.1 Algorithms

We will apply various standard classification algorithms for our task, including logistic regression, SVM, decision tree, multilayer perceptron, Naive Bayes, and finally bagging. We will evaluate these algorithms and compare their performances.

3.2 Features

The main component of our task is to come up with a list of features that we believe are informative to feed into the machine learning algorithms. In this project, we chose a set of topological features between an editor-article pair in the graph. Since we have a bipartite graph, we adapted the features that are commonly used in unipartite graphs. We adopt the following notations. For a node x, we define $N(x)$ to be the set of x's neighbors, and $\bar{N}(x) = \cap_{y \in N(x)} N(y)$, or the set of x's neighbors' neighbors.

We consider the following features:

- **Sum of neighbors** For an article, this is the number of editors that edited it. For an editor, this is the number of articles he has edited. This number may be meaningful as the more articles an editor edited before, the more likely he will edit more articles because it suggests that he is more active. If an article is edited by many editors, it may indicate that it is a popular topic; it is controversial; or it concerns a difficult topic.

- **Editing Frequency** For an article, the frequency at which it was edited by people. For an editor, the frequency at which he edits articles. This is similar to the sum of neighbors, but is limited by the number within a certain time frame, thus adding some temporal information. For example, an article with many total edits may only have a few in the past year, suggesting that the content is rather complete and needs no new editing.

- **Shortest Distance** This is the minimum hop count between an editor and an article. We hypothesize that the shorter the distance between the editor and article, the more likely the editor will edit the article. This measure characterizes the "degree of separation" between an editor and a article, or how closely related are the editor's interests and the article's topic.

- **Common Neighbors** In our bipartite graph, for an editor- article pair, (e, a), this is defined to be $|N(e) \cap \bar{N}(a)|$. Since this feature is adapted to be "the intersection of the articles this particular editor edited, and the articles that the editors who edited the article in question edited," it basically captures the notion of "people who edited this article also edited..." This measure implies the commonality between the editor and the article.

- **Jaccard's Coefficient** This is also known as neighborhood overlap. In our graph, this is defined as $\frac{|N(e) \cap \bar{N}(a)|}{|N(e) \cup \bar{N}(a)|}$. This feature is similar to Common Neighbors, but it is normalized by the total number of neighbors, which should make it more informative of the commonality between the editor and article.

- **Adamic/Adar** This measures uses the frequency of common features to compute similarity between two nodes as $\sum_{z:\text{features shared by } x,y} \frac{1}{\log(frequency(z))}$. In our context, the feature is the neighbors, and this measure is defined as $\sum_{z \in N(e) \cap \bar{N}(a)} \frac{1}{\log |N(z)|}$.

- **Preferential Attachment** This is similar to the sum of neighbors measure above and suggests how active the editor is and how popular the article is. It is defined as: $|N(x)| \times |N(y)|$.

4. DATASET

We used the processed metadata for all revisions of all articles on Wikipedia as of 2008-01-03 [5]. This dataset is 17GB compressed, and 2.8TB decompressed.

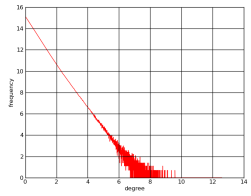
4.1 Data Processing

As mentioned, the size of this dataset is extremely large. However, the revision history also included other information that we do not need for this project, such as external links, images, etc. It also contains all editing records if an editor edits the article multiple times. Because we need a lot of RAM, we used Amazon EC2 in order to deduplicate these records to get the record of the first edit between an editor-article pair. From this extracted data, we further processed it assigning IDs to the editors as they are usernames or IP addresses in the original format. We further reduced the size of the data by removing timestamps, and partitioning the edges into different files as grouped by the year they were created. The resulting processed data is a total of 2.1GB containing all editor-article edges of the Wikipedia network.

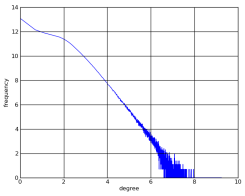
4.2 Network Characteristic

Before we proceeded on the link prediction task, we wanted to have an understanding of the general characteristics of the graph. In 2006, our graph consists of 2,928,834 nodes and 12,199,098 edges since the inception of Wikipedia in 2001. By 2007. it grew to 7,668,863 nodes and 32,864,902 edges. By 2008, it had 13,930,517 nodes and 58,638,292 edges, with near 10 million editor nodes. It shows that the growth of Wikipedia really took off since 2006 and had been accelerating.

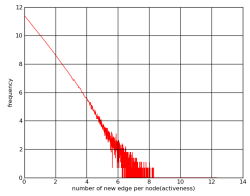
Figure 1(a) and 1(b) show the degree distribution of articles and editors respectively. They follow a power-law distribution, which is typically expected in a real network. Since we are performing link prediction, we are interested in what



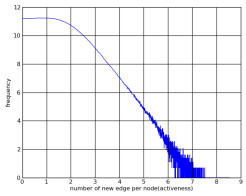
(a) Log-log Plot of Degree Distribution (articles).



(b) Log-log Plot of Degree Distribution (editors).



(c) Log-log Plot of Newly added edges to an article node distribution.



(d) Log-log Plot of Newly added edges to an editor node distribution.

Figure 1: Node degree distribution and newly added edges distribution in 2007.

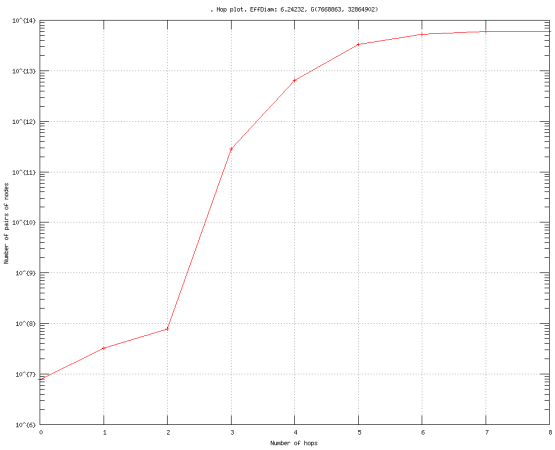


Figure 2: Cumulative Shortest Path Distribution

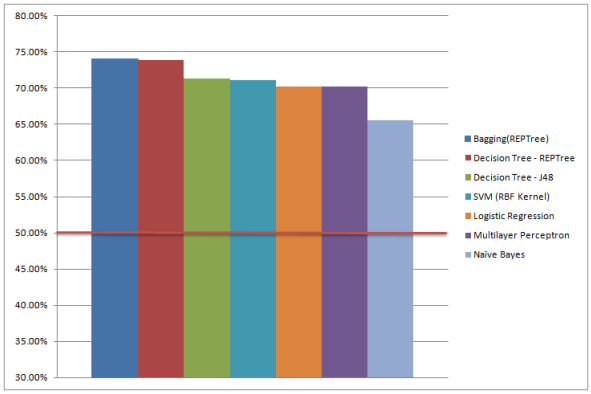


Figure 3: Testing accuracies compared to baseline.

the distribution of the number of newly added edges to the graph. Figure 1(c) and 1(d) show the distribution of newly added edges between the year of 2006 and 2007. They also follow a power-law distribution, as we would expect only a few number of articles would have many edits and a few number of editors would be the power-users. We also looked at the distribution of shortest paths in the graph, as shown by Figure 3. It shows that most of the pairs have a distance of three to four, and no longer than eight. This shows that the graph exhibits the small-world phenomenon, which is also typically expected in a real social network.

4.3 Obtaining Samples and Features

To obtain the samples for training and testing, we chose the four times that define the training and testing periods to be $[T_1, T_2] = [2006, 2007]$ and $[T_3, T_4] = [2007, 2008]$. We chose those times because the number of edge additions in both periods are comparable, and we chose the range of one year as it reduces possible variance if the range is too short.

Again, since the dataset is so large, instead of using all the edges, we only obtained a subset of the edges to be samples. We decided to select samples as follows. For positive training samples, we randomly select an edge in the 2007 snapshot of the graph. We check that both nodes exist in the 2006 snapshot of the graph. If they do, we add this pair to the set of positive training samples. We then fix the editor node, and randomly select an article in the 2006 graph. It there is no edge between that pair in 2007, we add that pair to the set of negative samples. We decided to fix the editor node for choosing the negative samples for two reasons. First, since there are so many pairs of nodes in the graph that are not linked, it is possible that we just select pairs that have little in common. This makes the problem possibly less interesting and easier. The second reason is that since our goal is to ultimately recommend articles, we would like to know whether our approach can actually distinguish good candidate articles from worse candidate articles for the same editor. Thus, for each chosen editor, we have a positive sample and a negative sample. The testing set was also obtained by the same method.

Using this method for choosing editor-article pairs, we computed the aforementioned features on all the pairs for the training and testing sets. We obtained four thousand samples in training set, and the same number in testing set. Both sets are balanced with 50% positive and 50% negative samples.

5. RESULTS

We applied various machine learning algorithms. We trained the models with the obtained training set and evaluated them on the testing data. For multilayer perceptron and SVM, we performed a cross-validation to select the optimal parameters before testing. Table 4.3 summarizes the results.

The result shows that the topological features are indeed informative for predicting links in the Wikipedia edit graph. Looking at the testing accuracy, we see that most algorithms are in the 70% range, except for Naive Bayes with 65.50%. This suggests that the Naive Bayes model probably is not

	Testing Accuracy	Class	TP Rate	FP Rate	Precision	Recall	F-value
Bagging(REPTree)	74.08%	0	0.639	0.157	0.803	0.639	0.711
		1	0.843	0.362	0.7	0.843	0.765
Decision Tree - REPTree	73.88%	0	0.66	0.182	0.784	0.66	0.716
		1	0.818	0.341	0.706	0.818	0.758
Decision Tree - J48	71.35%	0	0.596	0.169	0.779	0.596	0.675
		1	0.831	0.404	0.673	0.831	0.744
SVM (RBF Kernel)	71.08%	0	0.76	0.338	0.692	0.76	0.724
		1	0.662	0.241	0.734	0.662	0.696
Logistic Regression	70.25%	0	0.749	0.344	0.685	0.749	0.716
		1	0.656	0.251	0.723	0.656	0.688
Multilayer Perceptron	70.18%	0	0.693	0.289	0.706	0.693	0.699
		1	0.711	0.308	0.698	0.711	0.704
Naive Bayes	65.50%	0	0.912	0.602	0.602	0.912	0.725
		1	0.399	0.089	0.818	0.399	0.536

Table 1: Testing results.

as powerful as the other algorithms for capturing the relationships between the nodes as suggested by the features. This makes sense since the Naive Bayes model assumes all features to be independent, which is often a wrong assumption because many of the features are somewhat correlated as they all concern the topology around the two nodes. We also see that decision trees have the highest testing accuracy. Since decision trees can represent non- linear decision boundaries, it may be able to train more suitable models for this domain. With bagging, we were able to boost the decision tree's performance by a little bit to 74.08%, which is 48% better than the baseline predictor which would have a 50% accuracy.

The testing accuracy gives us a sense of how well the models can predict whether links will form in the Wikipedia edit graph. However, since we are interested in recommending the articles to editors, we are not necessarily as interested in the prediction of the negative samples because we will only be recommending the links that the algorithms predict to exist. Thus, the more suitable measures to compare are precision and recall for the positive classes. We see that with bagging, decision trees are able to achieve a precision of 0.70 and recall of 0.84. This means that if we recommend articles that the algorithm predicts will form to the editor, we would be right 70% on average. And of the articles that the editor will edit, our recommendations would cover 84% of them on average. Note that the Naive Bayes model has the highest precision of 0.82. However, its recall is also really low at around 0.40, meaning that it only covers a small subset of the articles that the editor will edit, making it a not so desirable model overall. The F-value is a harmonic mean of precision and recall, so it takes both values into account. We see that decision trees have the highest F-value, and can be considered the best model in this domain.

We also carried out ablative analysis in order to gain insight into which features are more informative than others. Using the decision tree, we report the testing accuracy of the model excluding one feature at a time. Table 2 summarizes that result. We see that when the feature"new links to article added in the past year" is removed, the testing accuracy decreases the most, suggesting that this is more informative than others. This implies that the number of editors that edited an article in the past year is a significant indicator of whether another editor will edit that article.

Excluded Feature	**Testing Accuracy**
Neighbors of editor	72.3
Neighbors of article	73.68
Shortest Distance	73.68
Common Neighbors	73.65
Jaccard's Coeff	72.05
Adamic/Adar	74.38
Preferential attachment	74.35
New links to editor	73.45
New links to article	**67.4**

Table 2: Ablative Analysis Result.

6. DISCUSSION

It is important to note that due to time constraints, we evaluated these algorithms on historical data and based on the articles that the editors actually edited. One might ask: what is the use of a recommendation system that recommends what the editors would edit regardless? One answer is that by recommending and presenting these articles early, we are potentially reducing the cost of contribution of the editors. Even though they would eventually edit the articles anyways, it may cost them time and effort to search or discover those articles by themselves. By reducing this cost, the editors may have more time to work on more articles. We have then effectively increased the participation rate by reducing cost of contribution, potentially leading to increases in quality and quantity of articles on Wikipedia.

Also, note that the testing accuracies reported above are again evaluated on the actual edits made by the editors, which does not completely represent the effectiveness and accuracy of the recommendations. This is so because the instances counted as wrong classifications in the evaluation might not be "wrong" had we actually deployed the system. The recommendation we presented might lead the editor to edit articles he would not have otherwise. Thus, the system might have a higher accuracy than the reported testing accuracy numbers.

6.1 Future Directions

Many future improvements could be made upon this project:

- **Additional Features**.

  In addition to topological features, we may consider page-to-page link relationship, text similarity between articles, categories of articles, and single article characteristics, such as length of article, topic, etc.

- **Vary Length of Learning Period**.

  In our experiments, the samples were obtained with the time frame being one year in length. It may be interesting to explore how accuracy might change if we vary the length of the learning period. Perhaps links formed in shorter intervals are more predictable, or maybe by lengthening the period, we allow more time for the recommendations to take effect as people have more time to react to the recommendations.

- **Actual System Deployment**.

  As previously mentioned, our system is not actually deployed, so we do not observe the actual effects the recommendations have on the editors. This limits the evaluation of our method's effectiveness.

7. CONCLUSION

In this project, we studied the problem of recommending articles on Wikipedia for editors to work on. We formulated the problem as a supervised learning link prediction problem on a bipartite edit graph of Wikipedia. We processed and extracted relevant data from the extremely large Wikipedia revision history dump. We studied the network characteristics of the Wikipedia edit graph. Finally, we applied machine learning classification algorithms on the data and achieved 74% accuracy with decision trees. Using only topological features, we obtained promising results and opened up many directions for future work.

8. REFERENCES

[1] Cosley, Dan and Frankowski, Dan and Terveen, Loren and Riedl, John. SuggestBot: using intelligent task routing to help people find work in wikipedia. Proceedings of the 12th international conference on Intelligent user interfaces (IUI '07), Honolulu, Hawaii, USA

[2] Liben-Nowell, David and Kleinberg, Jon. The link prediction problem for social networks. Proceedings of the twelfth international conference on Information and knowledge management, CIKM '03.

[3] Mohammad Al Hasan and Vineet Chaoji and Saeed Salem and Mohammed Zaki. Link prediction using supervised learning. In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security.

[4] Zan Huang and Xin Li and Hsinchun Chen. Link prediction approach to collaborative filtering. In Proceedings of the Joint Conference on Digital Libraries (JCDL05). ACM

[5] The Wikipedia dataset was created from a publicly available database snapshot by Gueorgi Kossinets (Cornell University) supported by NSF grant BCS-0537606.