Predicting Edit Locations on Wikipedia using Revision History

Caitlin Colgrove, Julie Tibshirani, Remington Wong

December 15, 2011

1 Introduction

There has been increasing interest in the machine learning community in automatic task design. In a collaborative problem-solving setting, how can we best break up and assign tasks so as to optimize output? Huang et al., for example, considered the problem of effectively assigning image-labeling tasks to Amazon Mechanical Turkers [1]. In the realm of Wikipedia prediction, Cosley et al. created a successful system for recommending articles a user might be interested in editing [2]. Nicknamed SuggestBot, the system makes use of co-editing patterns and articles the user has previously edited to predict new articles of interest. This system represents a success in automatic task design: deploying SuggestBot increased a user's edit rate an average of fourfold.

We consider a complementary problem in Wikipedia task design: given that a user has chosen an article to view, what sections would he be most interested in editing? Our hypothesis is that, similarly to article recommendations, a user's personal edit history can help determine which sections he will be most drawn to in the future. When viewing an article about the United States, for example, a politically-oriented user might hone in on sections about the government as opposed to cultural or geographic information. Wikipedia edit history has already shown to be useful in other contexts such as predicting the semantic stability of articles [3, 4]. And in the task at hand, it seems to provide a rich source of personalized training data.

2 Contributions

There has been little to no published work on the task of recommending sections in a Wikipedia article. Moreover SuggestBot did not leverage machine learning in making its predictions. Therefore our main contributions lie in formulating the problem, preparing the relevant data, and exploring a plausible machine learning approach based on the user's edit history. Our final contribution is perhaps negative: the hypothesis that a user's edit history will help us recommend sections does not seem to hold. We ultimately find that while revision history may help in predicting what article a user chooses to edit, once a user has settled on an article, the features of the section itself are the biggest predictors of whether he will edit that section.

3 Data

Using Wikipedia's API, we downloaded the entire revision histories of five Wikipedia users: Hammersoft, JerryOrr, SF007, Tangledorange, and The_Egyptian_Liberal. We chose these users because they had substantial edit histories and because they have been editing articles recently. These revision histories contained the diff between the edit and the previous content of the article, as well as information about the text immediately surrounding the revision. We also downloaded the entire text of the 50 most recently edited articles for each user.

From the revision information, we needed to determine the section that the user had edited in each testing article. First, we parsed the articles into sections using the "==" separator. We also had a special regular expression designed to capture the Infobox as a single section. We then tokenized the text of the revision using the Stanford NLP parser [5] and compared it to each section using the Jaccard similarity coefficient to pick the best matching section. Because Wikipedia articles can change a great deal over time, we used only very recently edited articles in an attempt to ensure that there would be a well matching section.

4 Methodology

Multinomial Naive Bayes Our fundamental model was a Multinomial Naive Bayes model over a single user's entire revision history, using individual tokens as features and different sections of a test article as prediction classes. We chose to only use text that the user explicitly added or deleted for reasons explained below. We also needed to make some modifications specific to our data. First, since revisions are typically very short, we could not obtain a reasonable vocabulary only from training on revisions. This sparseness of data made smoothing difficult. To remedy this, we chose the vocabulary to be all words present in the most recent ten documents (excluding the one in the test set), along with an additional UNKNOWN token for words that did not appear. We then applied add-alpha smoothing of 0.1 to the entire vocabulary.

To make a prediction, our initial classifier simply calculated the likelihood of each section given the probabilities of the words it contained. We then ranked sections by likelihood and output the ranks.

Our initial results showed that our classifier overwhelmingly favored short paragraphs. We therefore introduced length normalization to all of our models, taking the geometric mean of the product of probabilities. We also experimented with changing the prior. Instead of assuming a uniform distribution over sections, we tried modeling edits as being uniform over words. The new prior is then given by the number of words in the section divided by the total words in the document. Concretely, let n_i denote the number of words in the ith section and T be the total number of words in the document. Then our prediction for each document is given by

$$c = \arg\max_i \frac{1}{n_i} \sum_{j=1}^{n_i} \log p(\text{word } j) + \log \frac{n_i}{T}$$

These probabilities are calculated using the usual maximum likelihood estimates. Letting m represent the number of revisions, R_i be the words in revision i, and V be the vocabulary,

$$p(\text{word } j) = \frac{\sum_{i=1}^{m} \mathbf{I}_{\text{word } \mathbf{j} \in \mathbf{R_i}} + \alpha}{\sum_{i=1}^{m} |R_i| + \alpha|V|}$$

. Our baseline is based on this new prior, and simply predicts the longest section in the article.

Mutual Information Related to our preference for short sections, we hypothesized that the likelihoods for long sections were being swamped by uninformative words. So to improve accuracy and correct for length of sections, we introduced a mutual information metric to select features. More formally, the mutual information for word j is given by

$$\begin{aligned}\text{MI}(\text{word } j) &= \sum_{w \in W} \sum_{c \in C} p(w,c) \log \frac{p(w,c)}{p(w)p(c)} \\ &= \sum_{w \in W} \sum_{c \in C} p(w|c)p(c) \log \frac{p(w|c)}{p(w)}\end{aligned}$$

where $W = \{\text{present}, \text{not present}\}$ and $C = \{\text{revised}, \text{present in a document but not revised}\}$. We chose the top n tokens most predictive of whether or not a section would be edited, as determined by the mutual information calculation (we found $n = 1000$ to be reasonable). Using this feature selection, we recalculated the Naive Bayes scores, but this time only considering the most predictive tokens.

Context and Document Weighting Because revisions are so sparse, we experimented with including two other types of text. The revision information we downloaded contained both the lines actually edited and a few lines surrounding the change. We experimented with including this edit context to increase the amount of training data. We also experimented with using the entire article to calculate probabilities, introducing a discount factor γ for words that appear in the article but not in the revised paragraph (after some experimentation we set $\gamma = 0.7$). Letting D_i represent the document associated with revision i, our new maximum likelihood estimates become

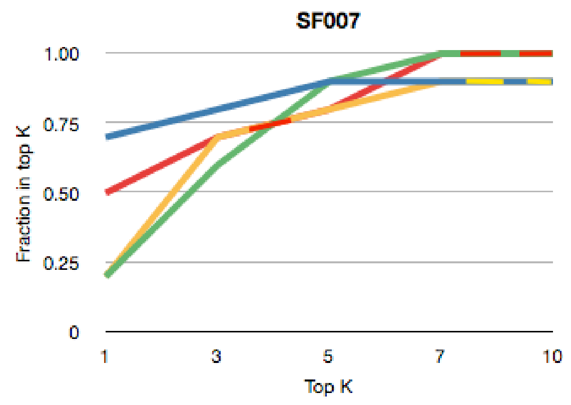
$$p(\text{word } j) = \frac{\sum_{i=1}^{m} \mathbf{I}_{\text{word } \mathbf{j} \in \mathbf{R_i}} + \gamma \mathbf{I}_{\text{word } \mathbf{j} \in \mathbf{D_i}} + \alpha}{\sum_{i=1}^{m} |R_i| + \gamma |D_i| + \alpha |V|}$$

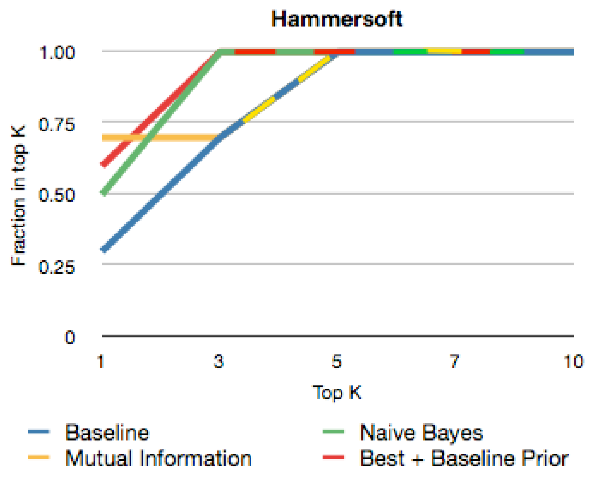
Testing We trained separate models for each of our five users. To test our results on a particular user, we performed leave-one-out cross validation on the user's ten most recently edited documents. We then outputted the predicted ranking of the section of the left-out article. To evaluate our ranking, we looked at the rank of the highest ranked section that was actually edited by the user (the great majority of the time there was only one, but occasionally there were many edited sections).

5 Results

While we ran many experiments on our data, we can represent the important information through the results of five different classifiers: our longest section baseline, pure Naive Bayes, Naive Bayes feature selection, Naive Bayes using context and downweighted document text, and finally the best out of the previous three with our new prior.

Considering a "success" to be ranking the correct section in the top three, our algorithms performed very well (see Table). However, the informed baseline – just predicting sections based on their length – was the largest contributor to this success. In most cases, just this baseline was able to successfully predict the section edited in eight out of the ten test documents. In two out of our five users, this baseline gives arguably the best results of any method we tried (see Figure 1). Naive Bayes did improve somewhat on this baseline for two of the other three users (see Figure 2) and matched it on the third.





After noticing that some documents were quite short, we ran several trials in which we chose the ten most recent documents with ten or more sections, and we saw that the results were significantly worse than the results on the long documents in our original training set. However, because finding these ten longer documents meant

User	Informed Baseline	NB	NB with MI	NB with full context	Best with prior	Max	Average # of sections
JerryOrr	0.8	0.4	0.3	0.6	0.5	0.8	15.8
Hammersoft	0.7	1.0	0.8	1.0	1.0	1.0	9.6
SF007	0.8	0.6	0.6	0.6	0.7	0.8	12.8
The_Egyptian_Liberal	0.6	0.8	0.7	0.8	0.8	0.8	20.5
Tangledorange	0.8	0.8	0.9	0.8	0.8	0.9	11.7

Table 1: Fraction of test documents in which the edited section was ranked in the top three.

using much older revisions, the accuracy of our matching was worse, and thus likely negatively impacted the overall prediction.

6 Error Analysis and Discussion

Our classifier performed well on all users, but different versions performed better for different users. For example, Naive Bayes worked quite well on Hammersoft because his edits followed a distinct trend: he often focused on correcting factual information in Infoboxes and taking down images that did not comply with Wikipedia standards. This meant that tokens common in Infoboxes and image links were highly predictive (indeed, the top features selected by MI for Hammersoft included the Infobox markup tokens "{", "}", plus ".png").

On the other hand, while SF007 frequently edited articles about software, he did not display a preference as to which sections he edited. For users like these, it seems as though modeling the overlap in content between a users previous edits and the test section is not the best approach. While using this overlap to predict what articles a user will edit may be effective, the sections within an article are already semantically similar. It is very difficult to capture content in such a fine-grained way so as to make accurate distinctions between sections. Moreover, our experience with the data suggests that most users might not even edit based on their interest in the content of the particular section.

Rather it seems that features of the section itself have the most predictive power. Indeed, we achieved a startlingly good performance with our baseline using a single, simple feature of each section. These observations also help explain why adding the context around an edit and using the full text of the revised document did not significantly improve our results.

While most attempts to deal with length problems had some beneficial effect, using mutual information did not seem to produce the same improvements. Using only a few features seemed like a natural way to make sure that all the words being counted were of high quality. However, this approach suffered from the fact that our training data was already sparse, and so discarding terms meant losing some of the little information we had. As a result of this sparsity, if a paragraph had none of the words we determined to be important, it would be automatically ranked very low even if it had a significant number of moderately informative words.

7 Future Work

Our error analysis strongly suggests that many users do not edit for content, and accounting for this difference in editing patterns could significantly improve accuracy. One plausible interpretation is that some users are "copy editors" while others are "content editors". It seemed that most of our users, especially SF007, were copy editors and simply focused on paragraphs that were poorly written or not well cited, while a few users like Hammersoft had a distinct preference for editing tabular and visual content. Adding features of the section itself such as the previous number of additions, deletions, and revision reverts; the number of distinct users that have edited the section; and perhaps some sort of “controversy rating would greatly improve accuracy on our copy editors. (Indeed, the logistic regression classifier we built in the milestone used only the revision histories of particular articles, and demonstrated that these features can be highly predictive of the attention a piece of text will receive).

A more sophisticated model would posit a latent binary variable that represents the user's editing style. Based on the value of this variable, certain features would be weighted differently: for content editors, the user's revision history would play a larger role as opposed to copy editors, for whom features of the section itself would become more important.

Elaborating on this idea of user profiling, we could first use clustering to group users by similar editing behaviors, and use the revision histories of similar users to help predict other users in the cluster. Such clustering would also help solve the problem of sparse revision text.

Finally, we spent a good deal of time on data collection and there are a number of logistical issues that could be resolved. In particular, in our project we matched each revision to a corresponding section in the latest version of the article, which forced us to consider only recent revisions in the testing set. For future work, we would expand our testing set by downloading the actual version of the article that the user saw when making the particular edit.

8 References

[1] E. Huang, H. Zhang, D.C. Parkes, K.Z. Gajos, and Y. Chen. Toward Automatic Task Design: A Progress Report. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, New York, NY, USA, 2010.

[2] D. Cosley, D. Frankowski, L. Terveen, and J. Ridel. SuggestBot: Using Intelligent Task Routing to Help People Find Work in Wikipedia. In *Proceedings of the 12th International Conference in Intelligent User Interfaces*, Honolulu, HI, USA, 2007.

[3] C. Thomas and A.P. Sheth. Semantic Convergence of Wikipedia Articles. In *IEEE/WIC/ACM International Conference on Web Intelligence*, Fremont, CA, USA, 2007.

[4] E. Yamangil and R. Nelken. Mining Wikipedia Revision Histories for Improving Sentence Compression. In *Proceedings of the 46th Annual Meeting of the Association for Computation Linguistics on Human Language Technologies: Short Papers*, Stroudsburg, PA, USA, 2008.

[5] D. Klein and C. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 2003.

[6] M. Pennacchiotti and A. Popescu. Democrats, Republicans, and Starbucks Afficionados: User Classification in Twitter. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2011.