Abstract

We explore a novel problem in automatic task design: when a Wikipedia user is viewing an article, which sections would he be most interested in editing? To address this question, we develop a probabilistic model that incorporates both lexical features drawn from a user's revision history as well as simple features of the candidate paragraph. Through experiments on held-out revision histories, we find that the model can accurately predict which paragraph a user is likely to edit. We examine the contribution of different features in the classifier to determine which characteristics of the paragraph are most predictive, and arrive at the somewhat surprising conclusion that most Wikipedia users are copy editors as opposed to content editors.

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 fourfold on average.

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 initial 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.

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. Our main contributions, therefore, 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, attributes pertaining solely to the section itself are the biggest predictors of whether he will edit that section.

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 had 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. For each user, we also downloaded the entire text of his ten most recently edited articles. Each of these articles represented the most recent version of the article available.

From the revision information, we needed to determine which section the user had edited in each of the ten articles. First, we parsed the articles into sections using regular expressions, including a special regex 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-matched section.

Methodology

Multinomial Naive Bayes

The core of our approach is to train 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. For reasons explained below, we chose to only use text that the user explicitly added or deleted. 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 from training only on revisions. This sparseness made smoothing difficult. To remedy this, we chose the vocabulary to be all words present in the most recent ten documents, along with an additional UNKNOWN token for words that did not appear. We then applied add-alpha smoothing.
to the entire vocabulary.

To make a prediction on a test article, 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 the geometric mean of the product of probabilities. We also introduced length normalization to all of our models, taking

\[
\text{argmax}_{c} \frac{1}{n_i} \sum_{j=1}^{n_i} \log p(\text{word } j) + \log \frac{n_i}{T}
\]

The probabilities are calculated using the usual maximum likelihood estimates. In particular, letting \( m \) represent the number of revisions, \( R \) be the words in revision \( i \), and \( V \) be the vocabulary we have

\[
p(\text{word } j) = \frac{\sum_{i=1}^{m} I_{\text{word } j \in R_i} + \alpha}{\sum_{i=1}^{m} |R_i| + \alpha|V|}
\]

To clarify, \( I_{\text{word } j \in R_i} \) takes value 1 if word appears in revision \( i \), and otherwise takes value 0. Our baseline model 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. Therefore, to improve accuracy and correct for length of sections, we introduced a mutual information metric to select features. More formally, the mutual information for a word is given by

\[
MI(\text{word } j) = \sum_{w \in W} \sum_{c \in C} p(w, c) \log \frac{p(w, c)}{p(w)p(c)} = \sum_{w \in W} \sum_{c \in C} p(w|c) \log \frac{p(w|c)}{p(w)}
\]

where \( W = \{\text{present, not present}\} \) and \( C = \{\text{revised, present in a document but not revised}\} \). We chose the top tokens most predictive of whether or not a section would be edited, as determined by the mutual information calculation (we settled on \( n = 1000 \)). Using the selected features, we recalculated the Naive Bayes scores, now only considering the most predictive tokens.

**Context and Document Weighting**

Because revisions are so sparse, we experimented with the inclusion of two other types of text. The revision information we downloaded contained both the lines actually edited and a few lines surrounding the change. We tried including this edit context to increase the amount of training data. We also experimented with the use of text from the entire article when it was available, introducing a discount factor for words that appeared in the article but not in the revised paragraph (after some tuning we set \( \gamma = 0.7 \)). Letting \( D_i \) represent the document associated with revision \( i \), our new maximum likelihood estimates become

\[
p(\text{word } j) = \frac{\sum_{i=1}^{m} I_{\text{word } j \in R_i} + \gamma I_{\text{word } j \in D_i} + \alpha}{\sum_{i=1}^{m} |R_i| + \gamma|D_i| + \alpha|V|}
\]

**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. More specifically, our training set was comprised of a user’s entire revision history (except for those diffs pertaining to the test document), and for certain models also included text from the 9 documents that were not left out. During testing, we ran the model on the left-out article and outputted the predicted ranking of each section in the left-out article. To evaluate our output, we considered 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).

**Results**

While we ran many experiments on our data, we can represent our main findings through the results of five different classifiers: (1) our “longest section” baseline, (2) pure Naïve Bayes, (3) Naïve Bayes with feature selection, (4) Naïve Bayes using context and downweighted document text, and finally (5) the best out of the previous three with our new prior based on section length.

Considering a “success” to be ranking the correct section in the top three, our algorithms performed very well (see Table 1). However, the new prior was the largest contributor to this success. In most cases, our baseline – which makes predictions solely based on section length – 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 (Figure 1). Naïve Bayes did improve somewhat on this baseline for two of the other three users (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

<table>
<thead>
<tr>
<th>User</th>
<th>Baseline</th>
<th>NB</th>
<th>NB with MI</th>
<th>NB with full context</th>
<th>Best with prior</th>
<th>Max</th>
<th>Average # of sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>JerryOrr</td>
<td>0.8</td>
<td>0.4</td>
<td>0.3</td>
<td>0.6</td>
<td>0.5</td>
<td>0.8</td>
<td>15.8</td>
</tr>
<tr>
<td>Hammersoft</td>
<td>0.7</td>
<td>1</td>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9.6</td>
</tr>
<tr>
<td>SF007</td>
<td>0.8</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.7</td>
<td>0.8</td>
<td>12.8</td>
</tr>
<tr>
<td>The_Egyptian Liberal</td>
<td>0.6</td>
<td>0.8</td>
<td>0.7</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>20.5</td>
</tr>
<tr>
<td>TangledOrange</td>
<td>0.8</td>
<td>0.8</td>
<td>0.9</td>
<td>0.8</td>
<td>0.8</td>
<td>0.9</td>
<td>11.7</td>
</tr>
</tbody>
</table>

Table 1. Fraction of test documents in which the edited section was ranked in the top three.
documents meant using much older revisions, the accuracy of our matching was worse, and thus likely negatively impacted the overall prediction.

**Error Analysis and Discussion**

Our models overall 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 mutual information for Hammersoft included the Infobox markup tokens "{", "}", as well as ".png").

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

In many cases, then, it seems that attributes pertaining only to the section itself have the most predictive power. Indeed, we achieved a startlingly good performance with our baseline using a single, simple attribute 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 immediately be ranked very low even if it had a significant number of moderately informative words.

**Future Work**

Our error analysis strongly suggests that many users do not edit for content, so 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 based on 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” could greatly improve accuracy on our copy editors.

A more sophisticated model would posit a latent binary variable representing 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 in comparison to copy editors, for whom attributes of the section itself would become more important. Elaborating on this idea of user profiling, another idea is to cluster users into groups based on editing behavior. Then, when making predictions for a particular user, we could also take into account the revision histories of other users in the same cluster. Such clustering could also 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, this paper 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. In 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.
References


Caitlin Colgrove graduated class of 2012 with a BS and MS in Computer Science from Stanford University, with concentrations in Theory and Artificial Intelligence. Outside of academics, Caitlin enjoys playing clarinet and spent four years with the Stanford Wind Ensemble. She is currently working as a software engineer at Palantir Technologies in Palo Alto, CA.

Julie Tibshirani is a master’s student in computer science at Stanford University, and this paper was written while she was a senior studying math (also at Stanford). Her academic interests lie mainly in machine learning and databases, and she is currently a member of Stanford’s Natural Language Processing group. She loves to teach and has worked both as a tutor and teaching assistant. In addition to computer science, Julie enjoys dancing, hiking, and reading philosophy.

Remington Wong is a second-year coterminal student in computer science. His main academic interests are systems, artificial intelligence, and theory. He enjoys coding and teaching, and has written several Web applications that help CS 103 students learn about finite automata and regular expressions.