Web-KB: Learning classifier to predict the type of webpage from the text

Ruochi Li, Dingdong Yao, Yulin Zhang
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
(rli14|dyao3|yzhan114)@ncsu.edu

1 Introduction

After reading the article Learning to Extract Symbolic Knowledge from the World Wide Web (1) and obtaining algorithms from our course and online resources, our team is curious that among three of our candidate text classification algorithms - Naïve Bayes, Maximum Entropy, and PCA+KNN (Principal Component Analysis + K-Nearest Neighbors) - which one yields the highest performance on the Web-KB data set.

1.1 About Web-KB

The World Wide Web is an essential resource that can be accessed by computers. However, it can only be understood by humans. To automatically generate a computer-understandable knowledge base whose content reflects the World Wide Web, a trainable information extraction system needs to be developed.

The team that initiated the Web-KB project aims to "develop a probabilistic, symbolic knowledge base that mirrors the content of the world wide web" (1). They first trained the system to extract information of the desired types and then allowed the system to browse new web sites in order to automatically populate a knowledge base with new assertions. Two inputs are needed to train this system. One is a specification of the class instances and relations of interest, which is the ontology that defines the vocabulary for the target knowledge base. Another input is the training examples that describe instances of the ontology classes and relations. By building upon these two inputs, the Web-KB system was able to achieve a classification accuracy of better than 70 percent at coverage levels of approximately 30 percent.

The Web-KB system does more than statistical text classification on every single page. To learn from the web, it relies more on relationships among pages, building a symbolic network graph that allows information extraction in clusters. In this project, we are not going to explore the relational aspect of their work, as building such an infrastructure is beyond our goal of building a simple learning classifier.

2 Inspirations

Before jumping into the data preprocessing step and following a classic text cleaning pattern, we went ahead investigating the data we had in hand. Specifically, we wanted to inspire ourselves with ideas that could improve the success rate of our classifier. Here are some inspirations on how we plan to modify the classic text cleaning pattern to make it more suitable for our task.

Preprint. Under review.

> **Inspiration 1**: The more frequent a word appears in a document, the more important this word is to that document.
>
> **Inspiration 2**: The more frequent a word appears in a type of documents, the more important this word is to this type of documents.

Both of these ideas are borrowed from TF-IDF (term frequency-inverse document frequency). According to a Data Mining (2) textbook from Stanford, TF-IDF measures how concentrated into a few documents are the occurrences of a given word. The measurement, which is often used as a weighting factor, indicates the relative importance of the given word to a given document or a given type of document. If a word is highly concentrated into a particular type of documents, then by identifying which documents have the word showed up often, the algorithm could mark them into one type. With these inspirations presume Naïve Bayes to be most suitable for this task.

> **Inspiration 3**: The word that is more eye-catching gets more attention from the reader, and is, therefore, more important.

Here, we define eye-catchiness as any effect putting onto the words, making them different in appearance. Any words surrounding by styling tags (e.g. `<h1>-<h6>`, `<b>`) can more easily be seen than words crowding in blocks of texts. We, humans, locate these emphasized words to determine the type of document we read. For example, when we see "Assignments" in the subsection header, we may confidently assume this page to be a course page. To reflect the importance of stylized words, we multiply some predefined weighting factors to words of different styles. However, the problem with this approach is that some words are less relevant to the document type, and weighting them adds up to the confusion of the algorithm. Such words are human names and professional terms, which when looking at individually, reveals no information about the type of document they belong to.

> **Inspiration 4**: The number and type of personal pronouns may differ among different document types.

After discussion, we arrived at a consensus that personal pronouns could be instrumental in identifying each document's type, at least in differentiating human and non-human pages. Student's pages contain abound first-person pronouns, whereas faculty's pages could have both first-person and third-person pronouns. In reverse, one seldom spots personal pronouns in department and course pages. Factoring personal pronouns into our input data may, therefore, lead to a boost in the accuracy of our classifier.

3 Method

3.1 Text Preprocessing

We are using the dataset from CMU World Wide Knowledge Base project (3), which can be found at

`http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz`

Since HTML files are raw data, the dataset needs to be transferred from human language to a single canonical form in order to be recognized by our algorithms. This process, known as text normalization, includes converting all letters to lower case, converting numbers to words or removing them, removing punctuation, accent marks and other diacritics, removing extra whitespaces, expanding abbreviations, removing stopwords, and stemming. All data files become vectors of tokens after normalization and are stored in a CSV file for future usage.

3.2 Naïve Bayes

Naïve Bayes is a probabilistic machine learning model rooted in the Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

As the number of attributes grows, solving Bayes theorem in its exact form becomes infeasible by reason of the daunting number of parameters to calculate before sending them in use. Naïve Bayes

solves this problem by incorporating a "naive" assumption that there is a strong independence among features, thus simplifies the Bayes' theorem into a less resource-consuming calculation. Although this assumption is not necessarily true at all times, the model has proved to be a powerful and effective approach to many classification problems. In the context of text classification, it assumes the occurrence of each word is unrelated to the occurrence of other words. And the bag of words model further simplifies the theorem by counting in an additional assumption that position in the document does not matter. The resulting revised Bayes theorem looks like the following:

$$P(y|w) = P(y) \prod_{w=i}^{W} P(w|y)^{count_w}$$

where w is the word and y is the class label. Our task here is to apply this formula to every word in every type of document, and given a map of word to word count, calculate the posterior probabilities for each document type. The one that has the highest posterior probability is going to be the prediction outcome. We hand-coded this algorithm, and used MultinomialNB from sklearn for comparison.

3.3 Maximum Entropy

According to Using Maximum Entropy for Text Classification (4) written by Kamal Nigam and his fellow co-writers, maximum entropy is a probability distribution estimation technique widely used for a variety of natural language tasks. Particularly, it is used to estimate the conditional distribution of the class label given the document, which is formulated as:

$$P(c|d) = \frac{1}{Z(d)} exp(\sum_i \lambda_i f_i(d, c))$$

where each $f_i(d, c)$ is a feature, λ_i is a parameter to be estimated and $Z(d)$ is the normalization factor. The underlying principle of this approach is that when nothing is learned, the distribution should be as uniform as possible, or in other words, having the maximum entropy. Learning training data puts constraints on the distribution, alters the probability distribution, and makes it less uniform. At the end of the learning cycle, we would get many models that fit our training data, and one should in favor of the one with maximized entropy since it most resembles the original state when no constraints are imposed.

This report discontinues from discussing the theoretical side of the Maximum Entropy approach in depth. With regard to implementation, we used the existing maximum entropy model from the NLTK library (5).

3.4 Principal Component Analysis + K-Nearest Neighbors

K-Nearest Neighbors a type of supervised machine learning algorithm we learned from the class. It is lazy and non-probabilistic compared to the two described above, for that no parameters are needed to be prepared before prediction. It delays model training until a test example has been inputted in. The basic idea is to find the k most similar data points, and by taking the predominance of class labels among these data points, one can predict the class of the test example. The similarity can be interpreted as the distance between one data point to the other. With this being said, the measure of similarity is inversely proportional to distance. For this project, we are using Euclidean distance to sort out which points are adjacent to the test example.

$$d(p, q) = d(q, p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Principal Component Analysis (PCA) is a dimensionality reduction technique used to retain meaningful properties of the original data. It involves computing the principal components and applying them to carry out a change of basis on the data. We reserved 500 principal components (the elbow) and averaged classes of 50 nearest neighbors.

4 Experiment Setup

4.1 Data

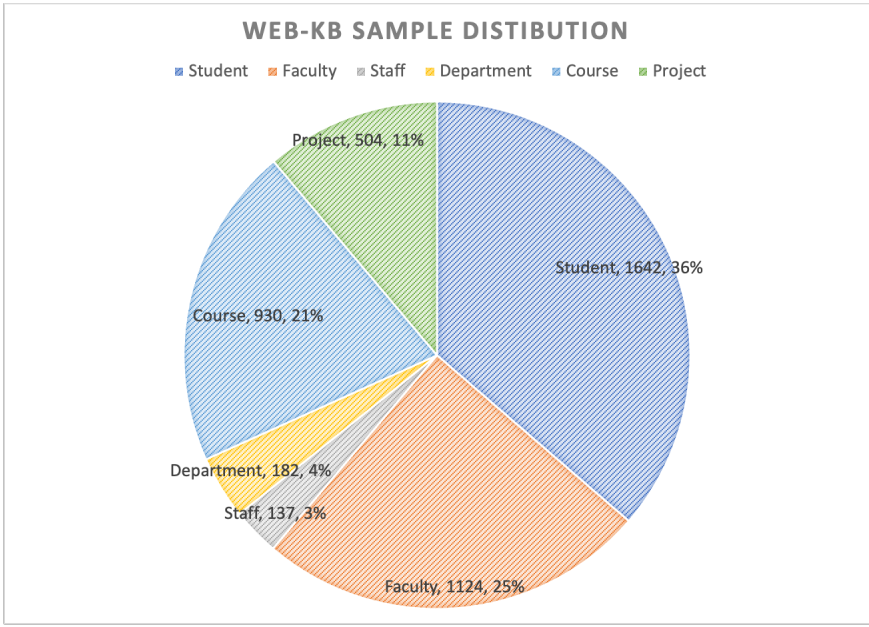

Figure 1. Web-KB Sample Distribution

The 4,519 pages were manually classified into the following categories: student (1642), faculty (1124), staff (137), department (182), course (930), and project (504). We removed from the dataset 2 files that could not be read by any encoding supported in Python. Therefore, the total number of data files is 4,517. We then used Python to convert them into usable data.

The result row had 5 reserved attributes and token occurrences:

d_type | **d_school** | **#p1** | **#p2** | **#p3** | word1 | word2 | ... |

where # stands for "the number of", and p stands for "pronoun".

The first column (d_type) is used to train the model. Type refers to the type of each document.

The second column (d_school) is used to split data into training and testing sets. The rule of split is described in the latter cross-validation section. To put it simply, for each university, we congregated pages from that university to be the testing set and the others to be the training set.

4.2 Function Headers

Each algorithm came with a pair of `fit()` and `predict()` functions.

`fit(X_train, Y_train)`
Input:
`X_train` - 2D array with each row representing a data object and each column representing a feature
`Y_train` - 1D array of labels
Output: None

`predict(X_test)`
Input:
`Y_train` - 2D array with each row representing a data object and each column representing a feature
Output: 1D array of predicted labels

In addition, extracting input arrays from the CSV file was handled in app.py. Within app.py, for example, one can call `NaiveBayes.fit(X_train, Y_train)` to train the NB model.

4.3 Model Evaluation

4.3.1 Cross Validation

In k-fold cross-validation, the original sample was randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample was retained as the validation data for testing the model, and the remaining $k-1$ subsamples were used to train the model. The cross-validation process then repeated k times, with each of the k subsamples served exactly once as the validation data. The k results were then averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once.

There are 4 universities explicitly categorized: Cornell (248), Texas (256), Washington (266), Wisconsin (321). We utilized 4-fold cross-validation for this project and partitioned data by their universities. Since each university's web pages have their own idiosyncrasies, we trained and tested on pages from different universities. The miscellaneous collection was always added towards three of the universities as the training set, and the holdout became the testing set.

4.3.2 Expected Results

By running the experiment, we intend to statistically compare the performance of different classifiers. We define the performance of the classifiers in terms of their prediction accuracy. Below we listed what we would like to gather from our experiment. These numbers are worthwhile to compare against each other as they enhance our understanding of different text classification algorithms.

- Accuracy of each classifier in each fold
- Accuracy of each classifier on each type of document
- Accuracy of each classifier using the best CV-score model
- Accuracy of the classifier that averages results from each classifier identified in step 3

5 Results

5.1 Cross-Validation Accuracies

We splitted the experiment into weighted and unweighted, and the results are displayed in the two tables beneath.

	Naïve Bayes	MultinomialNB	Maximum Entropy	PCA + KNN
1^{st} fold	0.5161	0.7540	0.7177	0.6573
2^{nd} fold	0.5781	0.8046	0.7851	0.6641
3^{rd} fold	0.4737	0.7594	0.7293	0.6692
4^{th} fold	0.4860	0.7664	0.7726	0.6667

Table 1. Results of cross-validation on unweighted data

This table presents the accuracies of the four models in each fold. These models were trained using unweighted data. Accuracy is calculated as the number of correctly classified instances divided by the total number of instances. As can be seen in table, MultinomialNB and Maximum Entropy performed similarly good, whereas the hand-coded Naïve Bayes was supposed to yield comparable performance as the MultinomialNB but did not.

	Naïve Bayes	MultinomialNB	Maximum Entropy	PCA + KNN
1^{st} fold	0.5161	0.7298	0.7177	0.6613
2^{nd} fold	0.5781	0.7969	0.7813	0.5977
3^{rd} fold	0.4736	0.7519	0.7030	0.6842
4^{th} fold	0.4860	0.7664	0.7882	0.6542

Table 2. Results of cross-validation on weighted data

This table is alike the previous table, except that models were trained using weighted data. The table shows that weighted data generally performed poorly than the unweighted data. One exception is

Maximum Entropy, which benefited from the added weights as these added weights help spread data points further away and maximize entropy more.

This result is counter-intuitive because we thought words putting in the headers and title must hold greater significance. The result proved our latter way of thinking that adding weight to words lets the original rare vocabulary to have an unexpectedly higher rate of occurrence. For example, a project page would have the <h1> header "CAD For VLSI Research Group." If we remove stop words, then the header becomes "CAD VLSI Research Group." Adding weight to words like CAD and VLSI blurs the correct pattern that could be observed without the weightings.

Since training using unweighted data produces a better result, we tested the best CV score models from the unweighted table against all the data, and the overall accuracies are demonstrated below. MultinomialNB scored the highest accuracy of 0.8729, followed by Maximum Entropy (0.7810) and PCA+KNN (0.7009), and lastly, the Naïve Bayes (0.3633).

	Naïve Bayes	MultinomialNB	Maximum Entropy	PCA + KNN
Best CV Model	0.3633	0.8729	0.7810	0.7009

Table 3. Results of cross-validation using best CV-score model

We watched that the hand-coded Naïve Bayes had reliable performance (up to 90% accuracy) when tested using data with shorter vector length. However, as we merged data together and lengthened each vector of tokens, the classifier performed somewhat unfavorably. Considering the next per-class accuracies table, all the files were classified as the Student type. The cause of this poor performance is uncertain. We attribute it to the size of too large features, so when multiplying probabilities, our code did not keep up with the high floating-point precision needed.

Besides, the Maximum Entropy was told to score better than the regular Naïve Bayes. Our results revealed the opposite. Since the maximum entropy algorithm is sensitive to inadequate features, we wonder that it might be the case we did not generate features optimally, which leads to this suboptimal outcome.

5.2 Per-Class Accuracies

	Naïve Bayes	MultinomialNB	Maximum Entropy	PCA + KNN
Course	0	0.9742	0.9645	0.9183
Department	0	0.5824	0	0.7142
Faculty	0	0.9066	0.8612	0.6797
Project	0	0.7817	0.1071	0.1965
Staff	0	0.0368	0	0
Student	1	0.9220	0.9805	0.8038

Apart from the Naïve Bayes explained in the last section, a noticeable thing here is that our classifier suffered from imbalanced input data. Some categories were difficult to be classified due to the lack of samples. For example, these classifiers performed below par in classifying department pages because each university has only one department page. The sample is too small to retrieve information from it.

Moreover, these classifiers also worked unsatisfactorily on staff and project pages. The logic is that such pages did not possess distinct features and might be confused with others. For instance, faculty, staff, and student pages tend to be classified into one type because they all describe a person. Likewise, the project and course pages have analogous contents, so it is easy for classifiers to make the wrong shot.

5.3 Ensemble Accuracy

We eliminated hand-coded Naïve Bayes from our ensemble of classifiers as the way it classified all files into the *student* type makes it unreliable. Therefore, our final ensembler consists only of MultinomialNB, Maximum Entropy, and PCA + KNN, and the resulting accuracy is 81.6%, which is mediocre in respect of how the MultinomialNB solely performed. Thus, we conclude that either the ensemble of three classifiers is too biased to inaccurate prediction, or good classifiers ought to compensate for other classifiers' poorness in performance.

5.4 Discussion on Inspirations

Inspirations 1 and 2 illustrate the importance of words to documents for their frequency of occurrence. These two inspirations are confirmed right, as indicated by the high accuracy using MultinomialNB.

According to **Inspiration 3**, the type of webpage is typically determined by the emphasized keywords. However, deviated from our expectation, adding weights to the emphasized words detrimented the performance (except Maximum Entropy). One possible reason is that we are setting weights improperly due to the time constraint to try every combination.

Inspiration 4 assumes pronouns could play a part in differentiating between human and non-human pages. We did not know to what extents this inspiration helps, but we believe it should not hurt as the idea is reasonable. The presence of pronouns should help distinguish human and non-human pages, but it may contribute little when distinguishing among human pages, say faculty, staff, and student, as these three types of pages can all use first-person pronouns and third-person pronouns interchangeably.

5.5 Further Discussion on Maximum Entropy

A single run of Maximum Entropy classifier produced the following output:

```
-0.022 assign==True and label is 'department'
-0.021 syllabu==True and label is 'project'
-0.020 syllabu==True and label is 'department'
-0.020 exam==True and label is 'project'
-0.020 resum==True and label is 'course'
-0.020 final==True and label is 'department'
-0.020 fall==True and label is 'staff'
-0.019 professor==True and label is 'staff'
-0.019 code==True and label is 'department'
-0.019 solut==True and label is 'department'
```

The features were sorted by absolute values, which corresponded to the relevance for the classification. These values were weights to multiply to each feature. Negative weights here would mean to decrease the file's likelihood to be a particular type when the feature shows. For example, when seeing the word "assign," the classifier gains more confidence to not mark the page as the department type. Our Maximum Entropy classifier does the elimination of choices. Since most features are about eliminating Department, Project, and Staff, the classifier learns little knowledge about these types themselves and therefore classified them poorly compared to the other three.

6 Conclusions

The purpose of this project is to build a learning classifier that learns from labeled web pages and predicts new ones. Three candidate algorithms were considered: Naïve Bayes, Maximum Entropy, and K-Nearest Neighbors with Principal Component Analysis as its pre-fitting step. From the Results section, we discovered that MultinomialNB, a variation of Naïve Bayes for discrete features, attained the highest accuracy, followed by Maximum Entropy. The paper Using Maximum Entropy for Text Classification (4) concludes that Maximum Entropy outperforms the regular Naïve Bayes both with and without prior, but our experiment results failed to affirm such conclusion.

Throughout the project, we learned that building a good classifier involves taking care of details at every stage. First of all, much of our effort should go to feature engineering, for its vital importance to the project's end success. Our feature engineering was done by extracting stemmed word counts from HTML files. Consequently, there were over 20 thousands unique stemmed words, and many of them have no specific meaning. It would be better to strip off these words so that our input features stay clean and relevant. Secondly, it is crucial to set the right parameters for algorithms, such as the k we set for KNN. Because of the time constraint, we only tried out some values for each parameter and chose the ones we felt reasonable. To improve the performance of our classifier, we must set each parameter to its optimal value. Third, when our results did not go as we expected, we have to

judge the results' validity and continue looking for bugs. Taking care of these details ensures a good classifier to be built.

Many areas of future work remain. Our classifier trained on imbalanced sample data, which resulted in some of the classes' mistaken classification. If possible, future researchers could acquire samples that have each type of web page equally represented. Secondly, the reason that the hand-coded Naïve Bayes ruined its predictions over aggregated data worths further investigation. One could try to compress the length of every vector of tokens and use a higher-precisioned data type for probability. Lastly, the cross-validation we proposed is not perfect. Future researchers should devise a more systematic way to cross-validate models. For example, one can deploy CVGridSearch from sklearn's model selection package to automate the tuning process.

References

[1] M. Craven, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, artd C.Y. Quek. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie Mellon University, January 1997. http://www.cs.cmu.edu/~webkb/

[2] Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17. doi: 10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2. http://i.stanford.edu/~ullman/mmds/ch1.pdf

[3] World Wide Knowledge Base (Web->KB) project. Carnegie Mellon School of Computer Science. http://www.cs.cmu.edu/~webkb/

[4] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. IJCAI-99 Workshop on Machine Learning for Information Filtering, pages 61–67, 1999. http://www.kamalnigam.com/papers/maxent-ijcaiws99.pdf

[5] NLTK Tutorial: Text Classification. https://lost-contact.mit.edu/afs/cs.pitt.edu/projects/nltk/docs/tutorial/classifying/nochunks.html#maxent