Intelligent Classification of Web Pages

Aayushi Agrawal agrawa@ncsu.edu
FNU Vivek vvivek@ncsu.edu
Ravindersingh Rajpal rkrajpal@ncsu.edu
Rithish Koneru rkoneru@ncsu.edu

Abstract

Web page classification is more or less a text classification problem. This project focuses on information extraction from textual content on the web page to correctly classify it to a category. Multiple text based classifiers have been explored such as Support Vector Machine (SVM), Multinomial Naive Bayes(NB) and Deep Neural Network (with Long Short-term Memory cell). The input to SVM and NB are tf-idf vectorized textual content. However, input to LSTM is a pre-trained Word Embedding Layer (GloVe) with a maximum vocabulary size. The classifiers were tuned using Grid Search and cross validated (5-folds) to find the best classifier. The evaluation metrics are accuracy, precision, recall and F1 measure. After exhaustive tuning and evaluation of classifiers it is found that the LSTM-based classifier performs the best on the given dataset.

1 Introduction

1.1 Problem

The amount of textual data accessible on web is evaluated to be in range of one terabyte. This has created challenges for users who want to search web classifying selected web documents into subject. Manual classification of web pages also suffers from the exponential increase in the number of Web documents. This project emphasizes the need for intelligent web page classification using natural language processing on the content with the help of automated classifiers.

1.2 Literature Survey

The authors have proposed a new web-page classification algorithm [3] based on web page summaries generated by human experts. Experimental results have shown that the proposed summarization-based classification algorithm achieves an approximately 8.8 percent improvement as compared to pure text-based classification algorithm. Most Web page classification models typically apply the bag of words (BOW) model to represent the feature space. The original BOW representation, however, is unable to recognize semantic relationships between terms. Both text and the context-features of a web page such as hyperlinks and HTML tags are used to model a SVM classifier to classify the web pages [5]. It was shown that context features consisting of title components and anchor words, improved the classification accuracy significantly. However, the method without using anchor words could not deliver consistently good classification for all the categories. Since web pages have data of high dimensions they need to be preprocessed well to identify their best representative features which better reflect their category.

For advancement in Natural Language Processing (NLP) tasks, deep learning methods have been recently been employed in multiple domains including learning distributed word, sentence and document representation [8] and parsing [9]. Since, a web page classification problem closely lies in the domain of text classification, deep learning methods can be reliably used and so, we explored Sequence Analysis using Long Short-term Memory (LSTM) model.

We implement 3 classifiers in the project. We start from simple probablistic model as baseline and go on to implement complex deep neural network. We ask the following research questions:

- **RQ1**: Do Deep Neural Network perform well in comparison to classical text based classifiers for web page classification?
- **RQ2**: Is accuracy correct metric to evaluate these classifiers on this dataset?

This project can broadly be classified into 4 main components:

1. **Dataset Preprocessing**: In this step we first load the data and then clean it up by removing HTML tags and other noise. The dataset is then divided into training and test sets.
2. **Feature Engineering**: In this step we transform the training data to a form that can be fed as input to our Machine Learning model. We do this by reading and transforming raw dataset to flat features(TF-IDF vectors), the process to accomplish this is explained in section 2.
3. **Building a Model and Training it**: In this step we build 3 classifiers: a) Support Vector Machine b) Multinomial Naive Bayes c) Deep Neural Network - Long Short-term Memory. The definitions and concepts to build each of these classifiers is explained in section 2.
4. **Performance Evaluation**: This is the final step and here we evaluate the performance of each model by first constructing a confusion matrix and then calculating various performance measures like accuracy, precision, recall and F1-Measure. Once the performance of each model is evaluated we choose the best model.

2 Method

We have a corpus of web pages in the data-set that represent one of the the 7 classes i.e. student, faculty, staff, department, course, project or other. We need to represent texts in a format a machine learning classifier would understand. For this purpose we explore Term Frequency-Inverse Document Frequency (TF-IDF). We have explored 3 classifiers for this purpose, Support Vector Machine, Naive Bayes and Long-short Term Memory.

2.1 Support Vector Machine

Support Vector Machine (SVM) has been known to be one of the best classification algorithms for text classification [10]. In general, SVM works well for linear classification. SVM is great at solving the problems of linear separation. SVM uses the training set to to find support vectors. In the case presented here, we have 7 classes. Each document is a web page associated with one of these classes. The support vectors represent the documents that lie on the boundary of linear separation of these classes. Every class will have corresponding support vectors. SVM classifies a given document based on these support vectors. The formula for binary classification is

$$(x_i, y_i), ..., (x_n, y_n), x \in \mathbb{R}^m, y \in \{+1, -1\}$$

where, $(\mathrm{x}_i, y_i), ..., (x_n, y_n)$ are training samples, n is the total number of training samples and m is the total number of features indication the dimension of the data, and y denotes the separation. In case of linear separation, the new point is classified basis this formula

$$(w.x_i) + b > 0 \; for \; y_i = +1, (w.x_i) + b < 0 \; for \; y_i = -1$$

Sometimes a hyperplane can not be found in the dimension we train SVM in. SVM can be trained in different dimension through the use of kernel trick. Also, if the problem is not linearly separable, SVM can be still used, if an appropriate kernel is established. One thing to note is all calculations are still done the dimension we operate in as it involves only dot product. So, kernel trick does not add to computation complexity. However, linear kernel is known to work best for text classification [10].

2.2 Naive Bayes

Naive Bayes classifier assumes conditional independence among all the features. This is known as "Naive Bayes assumption". Since, the parameters of the features are independently computed, because of the assumption, NB has low computational complexity in case of large number of features.

For web page classification, we generally have very large number of features. There are thousands of words in the vocabulary for text based classification. Naive Bayes has been successfully applied to many document classification researches [6].

We assume that the document are generated using a mixture model θ. The mixture model consists of mixture components $c_j \in C = \{c_1, ..., c_{|C|}\}$ [6]. Each component is parameterized by a disjoint subset of θ. Thus, a document i is created by following equation:

$$P(d_i|\theta) = \sum_{j=1}^{|C|} P(c_j|\theta)P(d_i|c_j;\theta)$$

The following Bayes rule is applied to classify a document by calculating a probability given a test document

$$P(c_j|d_i;\hat{\theta}) = \frac{P(c_j|\hat{\theta})P(d_i|c_j;\hat{\theta}_j)}{P(d_i|\hat{\theta})}$$

Since, the frequency of words can be transformed as conditional probabilities, it is also common to use Naive Bayes (NB) classification model for text based classification. Also, it has been shown that Multinomial model works better on large number of features than Multi-variate Bernoulli model [6]. Two documents are said to be correlated if they belong to the same category specified by the conditional probabilities based on the frequency of words. New document is classified basis Bayes estimates. Hence, we choose Multinomial NB for web page classification.

2.3 Long Short-term Memory

Long Short-term Memory (LSTM) is a recurrent neural network (RNNs) model. RNNs through their neural network architecture can propagate historical information. It looks at both the current input and previous output from the hidden states. So, RNNs has the ability to model a time-series data, it is adapted in sentence modeling. However, standard RNNs are not good at capturing long-term relationships. LSTM was first introduced [4] and reemerged as a successful architecture [2] that obtained remarkable performance in statistical machine translation [7].

LSTM has repeated architecture of basic RNN unit. Along with that, output of each unit is controlled through a set of gates. This gate functions on current input and old state. The three types of gates are forget gate, input gate and output gate. Together, they decide the update logic for current memory cell and hidden state. The LSTM transition functions are defined as follows

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$
$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$
$$q_t = tanh(W_q.[h_{t-1}, x_t] + b_q)$$
$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$
$$c_t = \sigma(f_t \odot c_{t-1}, i_t \odot q_t)$$
$$h_t = o_t \odot tanh(c_t)$$

Here, σ is the logistic sigmoid function that has an output [0, 1], *tanh* denotes the hyperbolic tangent function that has an output in [-1, 1] and $\odot$ denotes element wise multiplication. f_t controls how much the cell needs to forget, i_t controls how much new information needs to be stored in the current memory cell and o_t controls what to output based on memory cell c_t. Since, LSTM are great at learning long-term dependencies, we choose LSTM for the purpose of web page classifcation based on textual content.

3 Experiment Setup

This section will be divided in multiple subsections. We will go on to describe baseline classifiers and then our proposed model of IntelliWeb.

3.1 Python environment and packages used

For this study, we have an environment which has python v3.6 installed and other supporting packages are described in the table below:

Table 1: Softwares and Packages

Software/Package	Version
Numpy	v1.15.4
Pandas	v0.23.4
Scikit-learn	v20.0
NLTK	v3.3
BeautifulSoup	v4.6.3
matplotlib	v3.0.1
Keras	v2.2.4
tensorflow	v1.11.0
glove	v1.2

3.2 Data

We have a collection of webpages in the data set named Webkb. This dataset contains webpages from 4 universities and are packaged according to their labels. Basically, these webpages belong to one of 7 different classes. A total of 8282 webpages are present in the dataset. The data is available in 7 types of directories corresponding to each of the classes as shown in Table-2. Each of these directories have 4 sub-directories classifying the webpages which come from different universities namely (i) Cornell, (ii) Miscellaneous, (iii) Texas, (iv) Washington, (v) Wisconsin. Each of these sub-directories contains various webpages from respective universities. As we read these webpages using `pandas` we add a column indication their labels. This initial `raw_df`, raw dataframe is used for further processing.

3.3 Flow of Execution

We have implemented a pipeline of operations in order to perform classification. The block diagram of the pipeline is as follows and each operation will be explained thereafter. Figure-1 shows the pipeline. For baseline classifiers we use TF-IDF vecotrization. TF-IDF is classical way of vectorization. It is discriminative in nature that it penalizes features that occur very frequently. In case of proposed model, *vectorizer* is through *GloVe* embeddings.

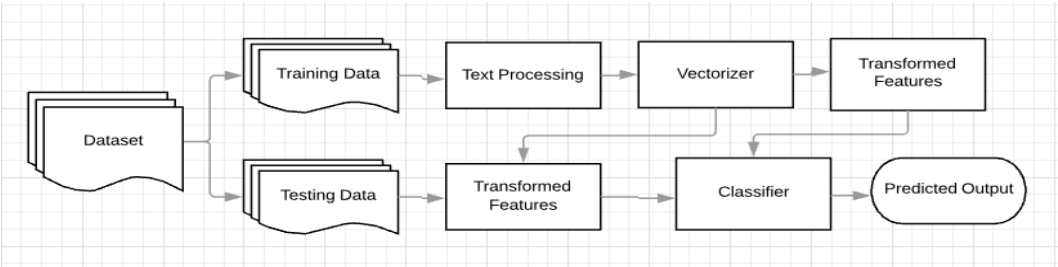

Figure 1: Flow of execution

3.3.1 Extract Web Pages

As mentioned earlier, the webpages are in directories of their entities. So, we need to first fetch all the webpages and combine them as raw data frame. After performing multiple file reads, we have a raw data set which is of the form [['$content_i$', '$class_i$'], ...,['$content_n$', '$class_n$']] where n is total number of web pages in all sub-directory described above and class is labeled as per the directory. Each row contains two fields: (1) Content – the raw html content of the webpage and (2) Class – which describes to which class this document belongs to. After this, we need to process the html tags and extract useful tokens.

3.3.2 Text-processing using NLTK

Alhough these are web pages, they represent information about a university. Therefore, they are in natural language. And moreover, they are English sentences. The useful tokens are the ones without

html tags and unique to the web pages. Since, the raw dataset contains various unwanted characters and tags, and repetitive words as well. We preprocessed the content by stripping of html tags first and splitting using word-punctuation tokenizer. In any english sentence, the punctuations and the stopwords don't add much meaning or help in inference, so we also removed those stop words and punctuations with help of `english` stopwords of `nltk`. Each token is also lemmatized, to bring the word to original word root. So, at this point we have processed and filtered tokens with respect to the webpage content and their corresponding class labels.

3.3.3 Vectorization - Feature Generation

The list of tokens for each webpage content cannot be directly passed to the classifier. For passing the content, we need to first form feature matrix so as to be used within the classifier. The feature generation is different for our baseline classifiers and proposed deep learning IntelliWeb model.

- **Baseline classifiers**: We have used TF-IDF vectorization for transforming words to feature vectors corresponding to our tokens. Term frequency – inverse document frequency is a numerical statistic that is intended to reflect how important a word is to the webpage. It is discriminative in nature, as it gives more weight to words which are infrequent among documents but frequent within document.
- **IntelliWeb**: To feed LSTM, we generally use word embeddings. We can either use trainable word embedding or pre-trained. Trainable word embeddings adds complexity to deep neural network. So we have used GloVe embeddings for generating feature vectors. GloVe is unsupervised learning algorithm to obtain word vector representations [1]. Training is performed on aggregated global word-word co-occurrence statistics from a corpus. Pre-trained word embedding are available and we use `glove.6B.100d.txt` available as open source through stanford website.

3.3.4 Machine Learners - Classifiers

After generating vectorized features, we pass it on to the classifier to train on. We have 3 classifiers as follows:

1. Baeline 1: Naive Bayes (NB)
2. Baeline 2: Support Vector Machine (SVM)
3. IntelliWeb : Deep Neural Network (LSTM)

Our implementation of NB with TF-IDF and SVM with TF-IDF relies on scikit-learn's implementation of the above models. The IntelliWeb model is implemented in keras. Prior to feeding data to classifier we split the input data to training and testing set.

3.3.5 Evaluate the results

Our evluation metrics are as follows:

- Precision: Defined as number of true labels as a fraction of total truly predicted labels, it says, number of true positives of all the positive predicted labels. Higher precision is preferred.
$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$
- Recall: Defined as number of predicted true labels as a fraction of total true labels, it says, number of true positives of all the positive labels. Higher recall is better.
$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$
- F1-measure: F1 measure is the harmonic mean of precision and recall. For both precision and recall to be high, we need to have high F1 score.
$$F1 = \frac{2 * Precision * Precision}{Precision + Recall}$$

- Accuracy: Accuracy is defined as the number of predicted true positives and true negatives among all the predictions.

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

LSTM-module can be trained on only accuracy as a metric, because it is trained in batches. So, true metrics to observe is accuracy. However, precision and recall is more important in our case, because the labels are not uniformly distributed as can be seen from Table-2. So, our target evaluation metric is high F1 score.

4 Results

The data set in consideration for this project has a total of 7 classes. The distribution of data is as follows: After performing experiments using 3 different types of classifier, the summarized results is

Table 2: Distribution of web pages across data set

Entity	student	faculty	staff	department	course	project	other
web pages	1641	1124	137	182	930	504	3764

explained below.

4.1 Naive Bayes

As explained in before, after performing classification through Naive Bayes Classifier using the feature matrix we prepared, the accuracy, precision and f1 score with respect to every class is mentioned in Table-3.

Based on the results shown in Table-3, we can see that precision and F1 score for 'Staff' turns out to be nan. It occurred due to less data of web pages for the respective class available in the data set. However, we see relatively good scores for other metrics.

4.2 Support Vector Machines

We performed two different experiments with SVM. Since, we got pretty good result after using Naive Bayes, we explored some ways to improve our feature matrix by inclusion of hyperlinks and title of every web page along side initial feature matrix which was used in Naive Bayes. First experiment was with respect to original feature matrix (Model 1) and second experiment was with respect to updated feature matrix as described above (Model 2). The results are shown in Table-3. Comparing results from Table-3, we can see that the new feature matrix performed fairly nearly the original feature matrix but is still performing a bit less than what they were intended to do.

4.3 Long Short Term Memory

While implementing LSTM, we thought of experimenting again on the feature matrix. Both experiments consists of GloVe embeddings, but the dataset slightly differs. First experiment was done on the original feature matrix (Model 1). In the second experiment, we observed that we have a lot of web pages under the category 'Other'(3764) which behave as irrelevant data as they can be separated from the original matrix (Model 2). The results achieved are shown in Table-3

We can clearly see improvement in accuracy, precision and F1 scores for respective classes, in Table-3.

5 Conclusions

RQ1*: Does Deep Neural Network perform well in comparison to classical text based classifiers for web page classification?*

Table 3: Results

Classifiers	Metric	Class						
		student	faculty	staff	department	course	project	other
Naive Bayes	Accuracy	0.89	0.90	0.98	0.98	0.93	0.94	0.80
	Precision	0.73	0.61	nan	0.9	0.79	0.74	0.74
	F1 Score	0.73	0.69	nan	0.38	0.67	0.31	0.79
SVM (Model 1)	Accuracy	0.90	0.92	0.98	0.99	0.95	0.94	0.83
	Precision	0.74	0.70	0.33	0.83	0.80	0.53	0.83
	F1 Score	0.77	0.74	0.26	0.82	0.78	0.56	0.82
SVM (Model 2)	Accuracy	0.89	0.90	0.98	0.98	0.94	0.94	0.84
	Precision	0.71	0.64	nan	0.74	0.78	0.58	0.80
	F1 Score	0.73	0.65	nan	0.72	0.73	0.53	0.83
LSTM (Model 1)	Accuracy	0.93	0.95	0.98	0.98	0.95	0.95	0.89
	Precision	0.85	0.81	0.62	0.64	0.76	0.55	0.86
	F1 Score	0.84	0.81	0.25	0.70	0.74	0.57	0.89
LSTM (Model 2)	Accuracy	0.91	0.92	0.97	0.97	0.97	0.96	
	Precision	0.87	0.81	0.5	0.63	0.94	0.82	
	F1 Score	0.88	0.84	0.12	0.72	0.93	0.81	

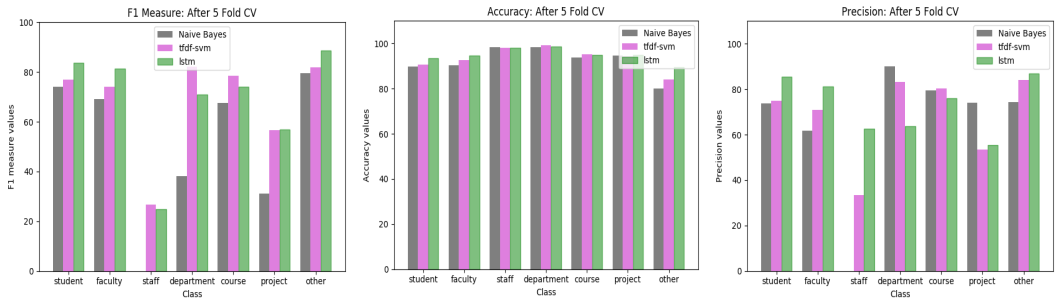

Figure 2: Evaluation metrics for each classifier for entities

From the results which we got after running 3 different classifiers, we can clearly see that SVM performs better than Naive Bayes. We can derive this from the fact that Naive Bayes works on probabilities and in our context, probabilities between words to find relations between their occurrence among various web pages and SVM works on creating hyper planes and differentiating classes on the basis of one against all structure which performs better. Comparing SVM and LSTM, LSTM performs better than SVM and that is clear from the results. Since, LSTM is deep learning and works with pre-trained embeddings on a very large corpus, training the model with deeper network results in deriving relations between context of the web page resulting in better classification.

We also performed two additional experiments. One experiment was with SVM with inclusion of titles of web page and hyperlinks. Hyperlinks tend to provide extra information about correlations between web pages and help to derive similarity and theoretically should improve the classification. We included hyperlinks, but on the contrary, the results weren't that good. It could be the case that we did not shape the feature matrix or did not pre-process the data well which resulted in contradicting results. The second experiment was exclusion of web pages with class 'Other'. The thought behind removal of those web pages seems plausible as the data was in abundance and somehow irrelevant with respect to the classification we were aiming at during the whole experiment. The results did turn out to be better as expected.

To conclude, LSTM turns out to be the better model in comparison to all other classifiers.

RQ2:*Is accuracy correct metric to evaluate these classifiers on this dataset?*

As evident from Figure-2 we see that all the classifiers perform really well on accuracy. LSTM does the best, but not by much. However, when we look at Precision and F1 score, there are many differences. This happens because in our data set, there are very high number of *Others* classes, almost 45% of the data. In general, across all classifiers, LSTM does the best. However, for LSTM to be really effective, we need equitable distribution of classes. Therefore, in case of LSTM with only 6 classes, excluding the webpages classified as *Others*, we see better performance as evident from Table-3.

6 Limitations and Future Work

IntelliWeb consists of Deep Learning module, which has a very high computation requirements. Moreover, IntelliWeb was trained on WebKb dataset which has only 8282 rows. In practice, DNN models require much larger dataset. IntelliWeb as an automated web page classifier works well on WebKb, but it can be trianed on evaluated on larger datasets and compute heavy cluster.

We have evaluated our results by performing grid search on SVM classifier. There are many hyper-parameters in LSTM model e.g. number of hidden units, number of hidden layers, number of iterations, etc. Hyper-parameter optimization of LSTM model can be done.

We did report the metrics for SVM (Model 2) classifier that trains on relationship between the web pages. Other relationships can be established basis HTML tags in the webpages. A graph based approach can be established. However, we need to evaluate how it will perform in comparison to Deep Neural Network based IntelliWeb model that takes benefit of natural language in the web page content.

References

[1] Pennington, J., Socher, R., & Manning, C.D., (2014) Glove:Global Vectors for Word Representation. *In EMNLP*, Vol. 14., pp:1532–1543

[2] Ilya, S., Oriol, V. & Quoc, L. (2014) Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems 27*, pp:3104- 3112.

[3] Shen, D., Chen, Z., Yang, Q., Zeng, H., Zhang, B., Lu, Y., Ma, W. (2004), Web-page classification through summarization. *Proceedings of the 27th annual international ACM SIGIR 04 conference on Research and Development in Information Retrieval*, New York, ACM Press, pp:242- 249.

[4] Hochreiter, S. & Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*. *MIT Press*. 9(8):1735–1780

[5] Sun, A. , Lim E. & Ng W. (2002) Web classification using support vector machine. *Proceedings of the 4th international workshop on Web Information and Data Management*, New York, ACM Press, p: 96-99.

[6] McCallum, A. & Nigam, K. (1998) A Comparison of Event Models for Naive Bayes Text Classification *AAAI Workshop*, *Workshop on Learning for Text Categorization*

[7] Zhou, C., Sun, C., Liu, Z. & Lau, F., (2015) A C-LSTM neural network for text classification. *arXiv preprint* arXiv:1511.08630.

[8] Mikolov, T., Chen, K., Corrado, G. & Dean, J., (2013) Efficient Estimation of Word Representations in Vector Space. *arXiv preprint* arXiv:1301.3781.

[9] Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., & Potts, C., (2013) Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of EMNLP*, pp:1631- 1642.

[10] Sebastiani, F. (2002) Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Github Link: https://github.com/rvsingh31/IntelliWeb