HW1: Linear Classifiers

Zhivar Sourati Hassan Zadeh
University of Southern California
souratih@usc.edu

Abstract

Text classification with its applications in different domains and contexts has experienced a surge due to the large amount of textual information getting produced everyday. In this work, we try to explore the extent to which different linear classifiers with both general and more dataset-tailored features can succeed in this task. Despite their simplicity and also the minimum amount of resources they need, our results show that linear classifiers can be great tools for fast, automatic, and cheap classifiers. Nevertheless we found that not all the methods alongside all the features perform at the same level and they seem to outperform each other in different settings.

1 Introduction

Considering the abundant resources of online information in social media as well as the other resources like scientific papers or news articles, organizing all these documents in a structured and well defined format needs much effort from human. There are so many ways that we can categorize these documents according to where they come from, but by and large all of them can be framed as multi-class classification problems. The ability to automatically label these documents would be beneficial from various angles among which, the minimization of the cost and fast pace of the whole process can be mentioned. We could name a few examples to elaborate more on the topic.

Identifying whether a posted article in social media by a person expresses depression or any kind of sadness can help prevent events that otherwise would be unavoidable given the number of people active in social media and few experts in the area who can monitor people's behaviours. Plus, having millions of articles about different topics in resources we use everyday can make the job of recommending the right article to the right person arduous. These situations would call out for fast automatic classification models that can deal with data from any arbitrary domain.

There are multiple ways to frame the learning process of the classifiers as well as many featurization techniques that can be utilized and might be suitable only for some datasets. Having these different considerations in mind, the outline of our work is as follows: In Section 2, we discuss the datasets used briefly. Section 3 will discuss the methods we used for training the model followed by a short remark on the details on the experiments we did in Section 4. Finally, the results will be presented and further insights will be given for future works respectively in Section 5 and Section 6.

2 Dataset

Four datasets have been used, namely, "products", "questions", "4dim", and "odiya" each having their distinct characteristics. The summary of the datasets used is demonstrated in Table 1

dataset	language	# records	# labels
products	English	32592	2
questions	English (encrypted)	4089	6
4dim	English	1560	4
odiya	Oriya	15200	3

Table 1: Summary of the datasets

As we can frame all the datasets in a multi-class classification task, the statistical features based on the number of words like bag-of-words (BOW) can be applied on all the datasets, even on the encrypted dataset as the statistical features of language will be preserved through Caesar Cipher which is the case for "questions" dataset, but further, to better suit to the needs of every dataset, we chose to incorporate more features. To incorporate the happiness level of the words we used Hedonometer [1] (Dodds et al., 2014) which provides us with the happiness

[1] https://hedonometer.org/timeseries/en_all/

scores from zero to ten for different languages and for each word. We chose this feature in order to capture the sentiment of the words specifically for "products", and "4dim" datasets. In addition, to identify the important keywords in each document dealing with "questions" and "odiya" datasets to recognize the category they belong to, we used TF-IDF (Ramos). Another feature that we used is the combination of average word length and number of the words in a sentence which we denote it by "len" when getting utilized in order to capture length-related features. Lastly our justification for using Word2Vec as it is mentioned in Section 5.2 is to capture richer semantic attributes of English words.

3 Methods

Our aim in this work was to explore the strength of different methods and the settings in which they might demonstrate better performance. particularly, we have focused on Naive Bayes classifier (Hand and Yu, 2001), Perceptron model (Rosenblatt, 1958), and Logistic Regression model (Wright, 1995). Apart from the mentioned methods, we have also inspected the Weighted Naive Bayes classifier (Lee-Kien Foo, 2022) which tries to improve upon the base model integrating some notation of weight for each word.

3.1 Naive Bayes

Abstractly, Naive Bayes is a conditional probability model, which tries to label the individual records taking into account the posterior probability of seeing the specific labels given the words in a sentence and the prior probability of seeing the label which is formulated in Eq. 1. Furthermore, based on the chain rule and the simplifying assumption of occurrences of words independently from each other, we can get to the Eq. 2. We used bag-of-words (BOW) to extract features from each of the N sentences d_i with class $c_i \in C$ containing words w_j^i with counts $count(w_j^i)$.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{1}$$

$$P(c|d) = P(c|w_1, \ldots, w_n) \propto P(c) \prod_i^n P(w_i|c) \tag{2}$$

As in the classification stage we try to find the label which maximizes $P(c|d)$ in Eq. 2, we only need to find the probabilities $P(w_i|c)$ in our training stage. Although these models have naive design and apparently oversimplified assumptions, Naive Bayes classifiers have worked quite well in many complex real-world situations (Lewis, 1998).

3.2 Weighted Naive Bayes

In real application of classification, the assumption of conditional independence between words cannot always get satisfied. One approach to alleviate the independence assumption is to incorporate attribute weights into the Naive Bayes. In (Lee-Kien Foo, 2022) authors propose to incorporate the weight a_i of attribute w_i into the Naive Bayes classifier as in Eq. 3. In Section 4, the further details of attributes a_i used are discussed.

$$c = \underset{c}{\operatorname{argmax}}\, P(c) \prod_i^n P(w_i|c)^{exp(-a_i)} \tag{3}$$

3.3 Perceptron

Perceptron is an algorithm for online multi-class classification. This algorithm, like most Perceptron algorithms is based on the biological model of a neuron with an activation for each class represented in Eq. 4 and the winner is basically the neuron with highest activation magnitude. As an important remark on the differences between Perceptron and Naive Bayes, we can say that Naive Bayes is a generative algorithm while Perceptron is a discriminative algorithm. Note that the X in Eq. 4 can be any set of features and W as the weight vectors for all different classes concatenated which means W_c corresponds to the weight vector of the class c. Also note that the true label for each document can be shown as Y_i.

$$c = \underset{c}{\operatorname{argmax}}\, X^T W_c \tag{4}$$

The updating process would also be carried out based on the differences between the true labels and labels that the model returns which results in the Loss function presented in Eq. 5.

$$Loss = \frac{1}{N} \sum_{i=1}^{N} [(\max_c X^T W_c) - X^T W_{C[Y_i]}] \tag{5}$$

3.4 Logistic Regression

As a more advanced version of Perceptron model discussed in Section 3.3, Logistic regression tries to

return the probability of the true class by applying a Softmax function (see Eq. 6) on top of the Eq. 4 which results in a smoother process for training because of the Softmax function being differentiable as opposed to the step function used in Perceptron.

$$P_c = \frac{e^{X^T W_c}}{\sum_{c'}^{C} e^{X^T W_{c'}}} \quad (6)$$

4 Experiments

In this section, we will point out the manipulations that we applied on the data to prepare it to get fed to the models.

As in all the datasets, there was no splits given to us, we did a train validation split of 80% and 20% as well as testing on the blind test set on Vocareum[2] given to us.

Regarding the pre-processing stage of our analysis, except for one of the datasets named as "questions", which we decrypted first using Caesar Cipher (Luciano and Prichett, 1987) to be able to apply Word2Vec Featurization on it; we only did a simple punctuation removal, followed by lowering all the words and removing the stopwords. Also, as a step to further normalize the features, we removed the words occurring less than ten times and also did the same when vectorizing based on TF-IDF (Ramos).

Moreover, to tackle the problem of unknown words in the Naive Bayes method, we exploited Laplace smoothing (Chen and Goodman, 1998) with α parameter assigned to one. Laplace smoothing is a smoothing technique that helps tackle the problem of zero probability in the Naive Bayes method by assuming that all the words have appeared in our dataset at least α number of times.

Tracking the progress of the training being of vital importance, we chose Weights & Biases (Biewald, 2020) to both check the metrics on train and validation dataset when training to prevent overfitting as well as a mean to visualize the results tracked through it.

Finally, to train the models, we have used gradient descent for the logistic regression using all the data points, and used the individual data points in the Perceptron training stage to update the weights in each epoch for 30 epochs using Logistic Regression model and 30 epochs using Perceptron model as we observed that after those points, no further improvements would be gained, and the model starts to overfit on the train split.

[2]https://www.vocareum.com/

5 Results

In this section, we will present our results and discuss different combination of features and methods that are used.

features/dataset	products	questions	4dim	odiya
unigram (val)	82	75	82	92
uni+bigram (val)	**84**	79	82	92
uni+bi+trigram (val)	82	80	82	92
Weighted BOW (val)	82	**81**	80	92
uni+bi+trigram (test)	83	84	92	92

Table 2: Weighted-F1 % of Naive Bayes Classifier with different features on four datasets

features/dataset	products	questions	4dim	odiya
unigram (val)	83	75	78	88
uni+bigram (val)	85	76	79	88
uni+bi+trigram (val)	85	76	80	88
uni+bi+trigram+len (val)	**85**	**76**	**81**	**89**
TF_IDF (val)	50	61	74	71
Word2Vec (val)	44	47	43	28
uni+bi+trigram (test)	85	82	97	88

Table 3: Weighted-F1 % of Logistic Regression Classifier with different features on four datasets

features/dataset	products	questions	4dim	odiya
unigram (val)	79	66	75	83
uni+bigram (val)	81	69	75	84
uni+bi+trigram (val)	81	70	74	84
uni+bi+trigram + len (val)	80	**71**	73	**85**
TF_IDF (val)	**82**	70	**77**	76
Word2Vec (val)	77	53	70	28
uni+bi+trigram (test)	81	80	82	83

Table 4: Weighted-F1 % of Perceptron Classifier with different features on four datasets

5.1 n-grams

In this study, unigrams, bigrams, and trigrams were used and their performance was investigated on different datasets. As our intuition suggests, we can incorporate more context in our features by looking not only at individual words, but at their neighboring words as well. Completely aligned with our hunch, we can also see in the results (see Tables 2, 4, 3) that trigrams perform better than bigrams and in the same way bigrams better than unigrams.

5.2 Word2Vec

As an attempt to capture the semantic information of the words in our datasets, we used Word2Vec pretrained models (Mikolov et al., 2013) to get the word representations and applied an average pooling layer on top of it to get the sentence embeddings. We explored the use of these features in Logistic Regression and Perceptron models and observed that they do not perform as well as the other features. These observations are totally explainable for Odiya dataset as it is not in English, however for the other datasets too, we cannot see metrics comparable to the performance of other features. There might be two reasons behind this, one being the average pooling layer on top of the feature vectors and the other one being the density of the features. It seems like that the model is able to learn better from more sparse matrices of features than small dense and richer-in-semantic ones. Furthermore, we can observe that the Perceptron model can use the Word2Vec features better than Logistic Regression model by a great margin, which its reason can be explored in future works.

5.3 Happiness Scores

Following the definition of weighted Naive Bayes and the intuition behind that which was discussed earlier in Section 3.2, we tried to use the happiness scores (Dodds et al., 2014) of the words as their weights when classifying data points. As you can see in Table 2, this feature is only nudging up the performance of our Naive Bayes classifier for "questions" dataset, nevertheless its presence is not significantly detrimental to the performance of Naive Bayes on other datasets, even though in some datasets sentiment seems not to play any role in determining its labels, at least from a general-knowledge perspective. This can be an evidence that emotion features do not necessarily have to get used in domains where labels are related to emotions.

5.4 TF-IDF

In information retrieval, TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus (Ramos). The reason behind using this feature was to give the model richer features to learn about dealing with datasets that contain categorical labels, namely, "odiya" and "questions". Doing so, you can see in Table 4 that the results would exceed the metrics we get using combination of n-grams, which similarly to Section 5.3, encourages that TF-IDF features can be useful in any context and in any multi-class classification problems.

To summarize our findings based on the used features and also the methods we used as well as our results on the blind test dataset, we can see that all the models perform on par with each other since all of them are able to capture the statistical features that is needed to classify the data points correctly. Still, so many questions can be asked about their differences, one can be the ability of the model to perform well on the settings with less data available.

5.5 Length of Sentences and Words

Although this feature is pretty simple and easy to use, we can see its benefits through the nudge it gives to the model on top of BOW features. The intuition behind its efficacy could be differences that latently exist in our datasets between the data points of different categories. For instance, looking at the results for "odiya" dataset, we can argue that different categories of business, sports, and entertainment articles posses distinct characteristics in terms of the length of the articles and the model is succeeding at capturing them through this feature.

6 Conclusion

We analyze the capabilities of different methods all fall under linear classifiers, alongside various features and also test our results on four datasets associated with distinct domains. Although the methods and features we used are pretty simple, they demonstrate reasonable learning potentials, and due to their generalizability can be used in any other context and domain. We also illustrate the effect of features specifically used to capture certain characteristics of the data and argue that even in the presumably wrong context, they might be useful rather than detrimental. While this work tries to have a broad overview of methods and features, further work can be done to focus on the nuances of the combination of features, methods, and datasets.

References

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling.

P. S. Dodds, E. M. Clark, S. Desu, M. R. Frank, A. J. Reagan, J. R. Williams, L. Mitchell, K. D. Harris, I. M. Kloumann, J. P. Bagrow, K. Megerdoomian, M. T. McMahon, B. F. Tivnan, and C. M. Danforth. 2014. Human language reveals a universal positivity bias. Preprint available at http://arxiv.org/abs/1406.3855.

David J. Hand and Keming Yu. 2001. Idiot's bayes: Not so stupid after all? *International Statistical Review / Revue Internationale de Statistique*, 69(3):385–398.

Neveen Ibrahim Lee-Kien Foo, Sook-Ling Chua. 2022. Attribute weighted naïve bayes classifier. *Computers, Materials & Continua*, 71(1):1945–1957.

David D. Lewis. 1998. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML*.

Dennis Luciano and Gordon Prichett. 1987. Cryptology: From caesar ciphers to public-key cryptosystems. *The College Mathematics Journal*, 18(1):2–17.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Juan Ramos. Using tf-idf to determine word relevance in document queries.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

Raymond E. Wright. 1995. *Logistic regression.*, Reading and understanding multivariate statistics., pages 217–244. American Psychological Association, Washington, DC, US.