Report On
Employing Linear Classifiers in Sentiment Analysis and Categorization

Basem Rizk
University of Southern California
brizk@usc.edu

Abstract

This paper examines a combination of classification tasks over sentences and paragraphs of different data-sets, classifying by categories, and by general sentiment. A number of features ideas has been experimented, from presence and counting, to contribution priority based on words order. The results reflect some thoughts over the underlying effect of vocabulary size on Naive Bayes in contrast with Perceptron, and support some and disclaim some of the intuitions behind some contribution heuristics. The insights provided by the experiments drives further investigation of more complex models that might be able to learn words order contribution relation.

1 Introduction

As the amount of human generated data grows further, the list of potential natural language processing (NLP) application expands. The paper examines linear classification on four different types of data-sets in different domains. They include hotels reviews, E-Commmerce websites feed backs, News headlines, and contextual sort of hashed questions. All of these domains are rich and NLP-Compatible, meaning that the potential monetary value of utilizing NLP, the potential moral value, and the potential user-experience value is high and that is capable of accepting more research for the sake of doing better and serving these different applications better. Through this paper, the experiment, employs Naive-Bayes, and Perceptron, classifying these different shapes and types of text from independent sources, labelled in terms of categories and of general sentiment (Pang et al., 2002). Furthermore, I test out a number of features from bare presence of words to priority of words contribution per sentence, over the variable domains of our data sources.

2 Data source

My experiment is conducted over 4 different data-sets, each of which comes from a different domain in separate data sources, however, we may group then into different groups based on different aspects either as sentiment analysis vs. topic categorization, as English vs. foreign language or as relatively short vs. long text, whichever division suitable for the course of discussion, and type of experiments conducted.

2.1 4dim

$4dim$ is a data-set that has been pulled from Google Maps 1,560 reviews of a number of hotels. They are labelled in 4 different classes, however, the 4 classes could also be viewed independently as two binary classes. The two binary classes go as follows a class of positive vs. negative sentiment, and a class of truth vs. deception. So the data points are labelled as follows: { Deceiving positive, True positive, Deceiving negative, True negative }.

Positive and negative labels were pulled based on stars accompanying each of the reviews. On the other hand, the truthiness labels might have been taken based on spam, expired emails, or a sort of copy-pasting of merged pieces of existing text detectors. The two classes of labels then were merged respectively.

2.2 products

$products$ is another general sentiment labeled data-set of similarly variable length 32,592 data points however of a set of products of an e-commerce website. The labels just as ‘4dim‘ were taken from the context of the stars provided accompanying the review on the website.

2.3 questions

$questions$ is a data-set of 4089 questions pulled from a textual context labelled in terms of 6 con-

textual classes: {abbreviation, entity, description, human, location, numeric value}

The language source here is not clear, but a hypothesis is that a transformation in some form of encoding or hashing have been applied to the original text.

2.4 odiya

$odiya$ is a data-set of 15200 different sentences pulled out of the Odiya language news headlines on topics of different contexts, which the corresponding labels point to. The categories are entertainment, sports, business.

3 Machine Learning Models

3.1 Naive-Bayes

Applying Naive-Bayes methodology is an approach to create an empirical model for the implicit rules that underlies the structure that connects/maps and the pieces of text of data-point to their labels whether general sentiment, or categories of interest.

Driven from conditional probabilities, and Bayes' rule in the context of mapping needed for our task of classification, while making the naive assumption that probability of observing any particular set of words in particular sentence corresponding to a particular label is independent.

$$P(CLASS|WORD) = \frac{P(WORD|CLASS)P(CLASS)}{P(WORD)}$$

Hence from a list of bags of words, each made of the words of a sentence of a data-point, while each bag is assigned to a label, we can build a Naive-Bayes model that does observe probabilistic connection between each bag and its respective label, and accordingly infer new guesses of labels on unseen bags of words assuming that the data-set, which would be building the model on, is independently and identically distributed.

For practicality, Laplace (+1) smoothing is applied to avoid out of vocab errors, producing tiny probabilities, and under-flowing in fraction is taken care of addition over log of fractions instead of multiplication.

3.2 Perceptron

Perceptron is another empirical model that is purely based on simple linear algebra, and aims to achieve the same goal of Naive-Bayes, however, underlying no assumption of Independence, and no deterministic way of learning its implicit definitions of the connections between the words and higher chance corresponding labels. It works in trial an error fashion, to create a fit set of weights that implicitly defines probabilities in some sense.

The weights can be viewed as DxC matrix in this case where D corresponds to the dimension size of our features and C is the number of classes (labels), and by works then by updating the weights corresponding to incorrectly labelled data-point of interest at a time until becomes fit enough, meaning learning the underlying probabilities of the data connection to labels, so it can infer labels of unseen new data-points.

4 Features Engineering

4.1 Embedding

Embedding of the data-points, the text, and their labels from our data-set, is required in some scenarios such as to deal with the purely linear algebraic modeling of Perceptron. To unify the framework, and make my modeling flexible for further feature engineering, the data-set is always encoded in numerical format, and that is done as follows: 1- Mapping every unique words in the data-set to unique values, simply using a counter. 2- Similar to 1, mapping is applied as well for the labels.

4.2 Features Vectors

4.2.1 Word count per label with Naive-Bayes

The upcoming step then differs based on the machine-learning model used. In the case of Naive-Bayes, the model simply treats the data-points as bags of words, by counting up numbers, which are needed for calculating the needed probabilities, of the occurrences of words, but in this case it would be their embedding.

4.2.2 Word Presence

On the other hand, for the Perceptron model, one necessity here is to have an identical size feature vector for every data-point, that is done using the encoding mentioned previously and using those as index of defined size feature vector based on the number of vocabulary used. if a word occurs at this data point, its index (based on the mapping) is set to 1, hence implicitly treating the data-point as bag of words as well here.

4.2.3 Word Occurrence Count

Another featurization methodology applied in my experiments is a similar vector to (Word presence)

but placing the number of occurrences of words at its mapped index in this encoded feature vector instead of just values of 1 indicating presence, careless of how much does it occur.

4.2.4 Bias Assumption Toward End

Assuming that the contribution of a word is valued more at some position in a sentence, the heuristics backed by this assumption are considered in four different forms:

- **First Best**: Meaning the earlier the occurrence of a word the more it is valued, and this intuitively backed possibly in questions better identify a question type, and in the case of reviews, arguably people who aim to write a bad review would be charged up at the beginning to actually decide on writing one down.

- **Last Best**: Meaning the later the occurrence of a word the less it is valued, this would come from also based on the intuition of an object of short sentence given possibly at the end, possibly directly related to a type of category, and in the domain of reviews or long text in general is that people aim to summarize and state clearer their point of view by the end.

- **End Best**: Meaning valuing more the words at either end and devaluing the words in the center. Supported by merging both intuition from First Best and Last Best where an introduction and conclusion to a piece of text might convey information for classification, or subject and objective typical locations in the case of short sentence/question,

- **Center Best**: Meaning valuing the contextual information, despite the intuitions driving the above featurization concepts, arguably they might relate to typical professionally written text, it might differ with the way people regularly text, talk, or write down reviews. Valuing the words right in the center of the text in the shape of normally distributed contribution is applied to test whether the words in the middle of a piece of text have contribute more to the class of interest (given label) than the ones at either end.

5 Experimental Results

5.1 Vocabulary size Effect

Table 1 lists accuracy on the 4 different data-set unseen test portion employing Naive Bayes (NB), and Perceptron (PP) in 20 epochs, using different fixed number/percentage of vocabulary included in the training data, and applying two different featurization methods in the case of Perceptron. These two particular featurization methods {word count, and word presence} are put against the Naive Bayes in this table because they are the closet resemblance to the way the features are dealt with in Naive Bayes algorithm implemented in my code. Moreover, the comparison exposes different points that might be insightful for the sake of optimization and taking this another step further with more complex models.

Based on the scores given, it seems that Naive Bayes is more prone to either over-fitting, or convergence when the vocabulary size is bigger, or in other words, when the features dimension is higher. An example that show this clearly is questions where the accuracy increases tremendously when as we cut down the vocabulary size. In contrast, you can see the PP Count and PP presence doing either slightly better or about the same with higher percentage of existing vocabulary.

5.2 Count Vs. Presence Vectors

Vectorizing the sentence into a bag of words rather than a set of words, in the sense that the redundant occurrence of a word does not add value, does not in fact add any value to the classification training in the case of Perceptron, according to these scores at table 1; In fact, the presence does show better performance on average here. Not sure if this has to do with the nature of the Perceptron, where higher numbers indicator in the encoded vector confuses the model, as it can not fit the spanning shape.

5.3 Contribution Heuristics

Table 2 shows a list of accuracy over a number of experiments with different featurization method from the priority/order contribution heuristics mentioned in the earlier section where F, L, E, C, correspond to First Best, Last Best, End Best, Center Best respectively, and different vocabulary sizes as found applicable and number of epochs in each particular execution on the 4 different data-sets.

On average First Best seem to do better, although it seems to over-fit with questions where possibly

4dim

VOCAB	NB	PP Count	PP Presence
Full	0.9	0.875	0.9
25%	0.925	0.775	0.875
10%	0.925	0.775	0.85
5%	0.9	0.675	0.85
5000	0.925	0.85	0.85

products

VOCAB	NB	PP Count	PP Presence
Full	0.812	?	?
25%	0.83	?	?
10%	0.821	0.761	0.791
5%	0.825	0.819	0.795
5000	0.825	0.757	0.814

odiya

VOCAB	NB	PP Count	PP Presence
Full	0.925	?	?
25%	0.916	0.879	0.891
10%	0.897	0.852	0.872
5%	0.875	0.848	0.842
5000	0.914	0.867	0.887

questions

VOCAB	NB	PP Count	PP Presence
Full	0.27	0.806	0.848
25%	0.646	0.834	0.784
10%	0.698	0.798	0.806
5%	0.706	0.766	0.8
5000	0.554	0.814	0.844

Table 1: Vocab size against Word Count/Word Presence Based Features using 20 epochs of Perceptron and using Naive-Bayes Classifiers

it is memorizing the question type and first few common words, which does not seem to apply to the validation nor the test set.

Moreover, Center best does not seem to do any good except being possibly acceptable with *odiya* which is quite interesting as it has the shortest sentence of them all, which either then that it is affect does not do much and pretty cool to considering each pure presence or that the object of classification is exactly mentioned in the middle as part of the grammar of this language. Worth noting also is that End Best performing significantly better than Center best in most times on the 4 different data-sets.

FT-VOC-EP	*4dim*	*products*	*odiya*	*questions*
F-5000-20	0.85	0.763	0.826	0.802
F-5000-100	0.9	0.723	0.828	0.768
L-FULL-20	0.875	?	?	0.7
L-5000-20	0.8	0.777	0.863	0.684
L-5000-30	0.875	0.799	0.848	0.73
L-5000-100	0.85	0.792	0.864	0.724
E-FULL-30	0.85	?	?	0.766
E-5000-100	0.825	0.797	0.863	0.776
C-FULL-20	0.7	?	?	0.584
C-5000-50	0.775	0.772	0.848	0.704

Table 2: A number of FT-VOC-EP (Featurization Method - Vocabulary Size - Number of Epochs) against each data-set

6 Conclusion

The experiments serve as good guide toward the direction of other experiments with linear classifiers, and testing dimensional size with other machine learning models as well, might be a good step forward.

The results regarding the contribution heuristics show potential due to the noticeable differences between the results over the different domains and different contribution intuitions, but these methodologies need further investigation; reformulating the underlying math behind implementation, as well as testing other models that are more capable of learning higher number of parameters.

Acknowledgements

Thanks for insightful discussion recently with a friend on embedding, and text vectorization through the work of RNN, and appreciation for the insightful post by Mina and Hirona on CSCI 662 Piazza discussion board that I used to build on my understanding on implementing Perceptron.

References

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.