Project for CSC 583 - Text Retrieval and Web Search Course

Sina Ehsani

Abstract

This document is for the course project of CSC 583 - Text Retrieval and Web Search.

Please find the code at the following GitHub repository: `https://github.com/sinaehsani6/TextMiningFinalPoject`

1 Introduction

This document is for the course project of CSC 583 - Text Retrieval and Web Search. For my project, I worked on recognizing textual entailment using the Stanford Natural Language Inference (SNLI) Corpus. (Bowman et al., 2015). This corpus contains 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference (NLI), also known as recognizing textual entailment (RTE).

For each data point we have the following: 1) sentence 1: Which is the main text. 2) sentence 2: this sentence is the hypothesis written by a human for the sentence 1. 3) gold-label: This is the chosen label that was arrived at by human annotators for the given hypothesis and premise. The label is marked as "gold-label" when the majority of annotators agree on one label. These will be either of NEUTRAL, ENTAILMENT or CONTRADICTION. (if they do not come out into an agreement the gold-label is marked as '-'). You can see the examples in Figure 1. The corpus is divided into three sections: train, dev, and test.

The top models of published results for this project have used deep-learning techniques to solve this problem. However, we wanted to use simpler and cheaper models to see who they can be compared to more expensive models. In particular, the contributions of this work are:

1. Using simple machine learning models to solve a complicated problem. For this project, Naive Bayes is used. Naive Bayes is very fast compared to other machine learning algorithm, easy to compute, and has relatively good performance in many cases.

2. Standardizing data to achieve higher performance. In this project we use standardization techniques, such as lower-casing, removing stop words from sentences, stemming and lemmatization, to improve our model. In addition to these standardizations, we also evaluated the results of using n-gram models. You can see the effect of these techniques in the results section.

3. Feature selection is used to handle the overfitting of the data to the training subset. we used mutual info feature selection to achieve this goal.

Sentence 1: *"a woman within an orchestra is playing a violin"*
Sentence 2: *"a man is looking in a telescope"*
Gold Label: CONTRADICTION

Sentence 1: *"three men one holding pipes another holding a large object above his head and one resting against the pipe bed on the truck are looking at the camera"*
Sentence 2: *"three men going to work"*
Gold Label: NEUTRAL

Sentence 1: *"six or seven people are standing on a pier with a table and a pair of glasses in the foreground"*
Sentence 2: *"six or seven people are standing on a pier"*
Gold Label: ENTAILMENT

Figure 1: Examples SNLI data, and the chosen hand-written labels (by majority of annotators)

2 Related Works

In this section, some of the top results of the published paper are shown. We are not going to go through how the published papers did their work (it is not in the scope of the project). We are just going to show some of the results. It is noteworthy to mention that only two results published by the SNLI website have not used some sort of deep-learning models.

Model	train	test
Unlexicalized	49.4	50.4
Unigrams Only	93.1	71.6
Lexicalized	**99.7**	78.2
300D LSTM encoders	83.9	80.6
SJRC (BERT-Large +SRL)	95.7	**91.3**

Table 1: Showing the accuracy of the models in both training and test sets, the first three models are feature based models. As you can see they highly over-fit on the training set. The LSTM model, is a simple neural network model with 3 million tuned parameters, and the BERT model has 308 million tuned parameters

As you can see from Table 1, the deep-learning models work better, but they need millions of tuned parameters, and therefore they are more expensive to compute. However, the BERT model seems to mark this model as a solved problem. (however, its results on other similar datasets should be evaluated)

3 Model

In this project, we used Naive Bayes and k-nearest neighbors algorithm. the details of these algorithms can be found in the course book. (Manning et al., 2010)

For all models used in this project scikit-learn (Pedregosa et al., 2011), a machine learning library in Python was used.

3.1 Data & Preprocessing

As mentioned before for this project the SNLI corpus was used. For preprocessing the data, the two sentences (sentence 1 and sentence 2) and the gold-label for each data point was extracted. The sentences then were tokenized, lowercased, and all the punctuation were removed. In addition, we also stemmed, and removed the stop words for each sentence and computed the results of these two changes on our model. For the gold-label, the data that the gold-label was not decided were removed (the ones marked as '-').

3.2 Building index

We used submodules, which gather utilities to build feature vectors from text documents. For our project, we tested the following:
Count Matrix: This indexing module converts a collection of text documents to a matrix of token counts.
tf-idf Vectorizer : This indexing module converts a collection of raw documents to a matrix of tf-idf features.
N-gram: For the above indexing, we also tried indexing over n-grams and checked the performance.

3.3 Naive Bayes

For the Naive Bayes model, we used the multinomial Naive Bayes classifier. This classifier is suitable for classification with discrete features (as well as with tf-idf weights)

3.4 k-Nearest Neighbors

We also implemented the k-nearest neighbors vote for this project. In this model, the classification is computed from a majority vote of the nearest neighbors of each point.

4 Improved model

To improve our model, we tried to add a feature selection model to the code. For the feature selection, we used the mutual information model, and top frequent terms.

4.0.1 Mutual information

Mutual information between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency. Then we choose the top k values as the model's features.

4.0.2 Frequency

Select the most frequent terms is also an easy but powerful feature selection, that is used in this paper.

5 Results

The results of the machine learning models used in this project are shown in Table 2. We can see that adding more features as n-grams improve the model. The training accuracy has been calculated with using 5-fold validation model. The KNN model does not seem to work as well as the NB for this dataset.

	train	test
NB Model		
Count Matrix	61	62
tf-idf Matrix	61	63
+ stem	61.7	63
+ lemma	61.5	63
+ lemma&stem	61.4	63
- stop words	59.9	61
+ bi-grams only	61.3	63
+ n-gram (1,3)	64	65
+ n-gram (1,5)	64	**66**
KNN Model		
n=5	56	57
n=20	56	57
n=50	54	57

Table 2: This table shows the accuracy of our results, as you can see changing the data dose not effect the model very much. However as you shown including n-grams improves the results.

6 Error Analysis

As you can see from Table 3, in the Naive Bayes model all of the labels perform evenly. I believe the reason for this behavior is the fact that the data is distributed evenly for all labels. (you can see the count of the labels in the support section.)

NB+tfidf	P	R	f1	support
contradiction	63	61	62	3278
neutral	65	60	62	3235
entailment	61	67	64	3329
micro avg	63	63	63	9842
macro avg	63	63	63	9842
weighted avg	63	63	63	9842

Table 3: This table is the result of Naive Base model with both stemming and lemmatization. the model have preformed evenly for all of the labels.

We also tried to see the effect of lemmatization, stemming and stop words on the results. As shown in Table 2, adding lemmatization or stemming does not have a significant effect on the performance. This might be because the vocabulary size is not that big for this dataset. In addition, as we can see removing the stop-words have decreased the accuracy. The reason for this might be the fact that the tf-idf matrix will take care of the stop words itself, and removing them before indexing will have the effect of the values of other words tf-idf scores.

7 Improving the Model!!

To improve our model, we thought the vast number of features might have a bad effect on the naive bayes model. Therefore, we implemented and tested to different feature selection models. However, as you can see from Table 4 they did not have a positive effect.

NB Model	train	test
NB	61	63
+ MI-500	56.4	57.4
+ MI-50	33.5	33.8
+ F-100	33.5	34

Table 4: This table shows the accuracy of our results when adding feature selection. the MI stands for mutual information, and F stands for the high term frequency feature selection. The number in front of each represents the top N selected features

To further analyze the result of the feature selection algorithm, we provided detailed values for each of the classes in Table 5. As seen in the table the recall for the ”entailment” class is 100% which means the model has predicted most of the examples as ”entailment” class.

NB+F-100	P	R	f1	support
contradiction	59	0	1	3278
neutral	53	0	0	3235
entailment	34	100	51	3329
micro avg	34	34	34	9842
macro avg	49	33	17	9842
weighted avg	49	34	18	9842

Table 5: This table is the result of Naive Base model with top 100 term frequency terms as the features.

8 Conclusion

We developed a simple model for recognizing textual entailment. The use of tf-idf indexing for naive Bayes showed improved result compared to regular term-count indexing. In the other hand, we saw that using feature selection decreased the model performance. The reason for this decrease might due to the relativity small number of features (vocabulary) for this task. As we can see from Table 3, the model does not overfit on the training set, and that is why feature selection does not have a positive effect on the model.

References

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.