Mini-Project 2 - Natural Language

Group 60

João Santos 92486

Sara Marques 93342

Models and Experimental Setup

Experiments

While finding a classifier adequate to this problem and dataset, we tested out various types of models on an 80/20 train/dev split of the original provided train.txt file (reported accuracies are calculated using the development set):

- Distance based with a K nearest neighbors classifier, k = 7 (accuracy - 40.3%)
- A Support Vector Machine (accuracy - 46.3%)
- A Decision Tree (accuracy - 42.65%)
- Random Forest, composed of 50 decision trees, each with a max depth set to 5 (accuracy - 39.3%)
- A Multinomial Naive Bayes classifier (accuracy - 46.75%)
- A Voting Ensemble composed of 2 models - a Support Vector Machine and a Multinomial Naive Bayes (accuracy - 47.25%)

We used the classifier objects from scikit-learn to implement these models in our project's code. They were also all used with a CountVectorizer, also from scikit-learn, to encode the review strings of text into vectors usable by the classifiers.

The split was done using scikit-learn's train_test_split function, ensuring a good random sample of the original file to be used as a development test set.

To define the baseline model, we used the best performing out of these, the Voting Ensemble composed of a SVM and a Naive Bayes classifiers, which reached an accuracy of 47.25% on the development test set.

This baseline was fed the review strings without any pre-processing, excluding the pre-processing done by default by the CountVectorizer (e.g. transforming upper case characters into lower case, so good and Good refer to the same word).

Pre-Processing of Reviews

Afterwards, to improve the overall performance of all classifiers, we started testing out various pre-processing functions to apply to each review string before they are fed into the models.

We tested the following pre-processing functions:

- Removing punctuation from the strings
- Removing stop words, using the stopwords list from the nltk corpus whilst adding a few exceptions that are on that list but are useful to the classification of a review

Exceptions: not, no, like, did, dont, didnt, wont, couldnt, cant, but, very, really, just, will, good, great, better, ok, okay, best

- Stemming using the Porter Stemmer made available by the nltk package
- Lemmatization using the WordNetLemmatizer also made available by the nltk package

Final Model

The final model we selected is the same model as the baseline but now using some of the pre-processing functions we experimented on. The ones used are the following:

- Removal of punctuation
- Stemming
- Lemmatization

The voting ensemble classifier, with these pre-processing functions done to the input strings, and a CountVectorizer to encode the input, achieves a final accuracy rate of 48.5% on the previously mentioned development test set, the strongest score of any combination of models and pre-processing we tested.

Results

The accuracy results of all combinations of pre-processing functions on the voting ensemble model (other models always achieved less accuracy).

Individually		Groups			
Ponctuation	47.9	Ponctuation + Stopwords	46.55	Ponctuation + Stopwords + Lemmatization	46.3
Stopwords	46.3	Ponctuation + Stemming	48.45	Stopwords + Stemming	46.95
Stemming	47.75	Ponctuation + Lemmatization	47.1	Stopwords + Lemmatization	46.2
Lemmatization	46.25	Ponctuation + Stopwords + Stemming	47.3	Stopwords + Stemming + Lemmatization	47.1
		Ponctuation + Stemming + Lemmatization	48.5	Stemming + Lemmatization	47.55
		Everything	47.5		

Discussion

The score achieved with the model is very much less than ideal. We can attribute this partially to the inherent subjectivity of the final label, while the strings of the review aren't descriptive enough to separate between a "Good" and "Very Good" label. For example:

Very Good – Wonderful for skin care

There are also some cases where the review string isn't anywhere descriptive enough for even a human to label the review, much less the model. For example:

Very Good – what!? no brush!?

Future Work

To improve the score achieved with the choosen model we could do a deeper analysis of the words in the stopwords list that influence the division between labels. But as previously mentioned this dataset has a big subjectivity and it's going to be very hard to achieve an ideal score.