Sentimental Analysis on Tweets

Iro Moumoulidou

Technical University Of Crete

June 23, 2016

Introduction

- Main Object: Classify tweets as positive/negative
- Build a Classifier with Supervised Learning

Our Data Set

- Data set retrieved from an in-class competition organized at Kaggle
- Set of tweets already classified as negative or positive
- 84.400 of them were used to train our classifier, 8.420 to test it

Tweet Representation

- Every Tweet is a **Bag of Words** (positive or negative respectively)
- Union of same sentiment *bags of words*, form the terms of each class
- However, do we need every single one of these terms?

Detection Of Uninformative Terms

- Characteristics of tweets: @username, #feelings , urls
- Stopwords
- What about terms that appear in both classes?

Detection Of Uninformative Terms

- For a term x of both classes a metric is defined as:

$$ratio = \frac{min(P_x, N_x)}{max(P_x, N_x)}$$

- If this ratio is close to one, occurrences of the term are almost equal
- A good threshold should be found for discarding these terms

Detection Of Uninformative Terms

- Another idea: Select the most representative terms-words for each class
- Metric used : tf as in Information Retrieval
- Term-frequency(tf) in a class is defined as:

$$tf_c = \frac{frequency\ of\ term\ in\ class\ c}{maximum\ frequency\ of\ class\ c}$$

- Again a good threshold should be found for discarding irrelative terms

Multinomial Naive Bayes

- Model used to train our classifier is Multinomial Naive Bayes
- Given a tweet t, the classifier should choose the class that maximizes the probability $p(c|t)$
- Formally we get:

$$\begin{aligned} c_{NB} &= \operatorname*{argmax}_c p(c|t) \\ &= \operatorname*{argmax}_c \frac{p(t|c)p(c)}{p(t)} \\ &= \operatorname*{argmax}_c p(t|c)p(c) \end{aligned}$$

Multinomial Naive Bayes

- However a tweet is a set of independent terms, so we get:

$$\begin{aligned} c_{NB} &= \operatorname*{argmax}_c p(t|c)p(c) \\ &= \operatorname*{argmax}_c p(t_1, t_2, \ldots, t_n|c)p(c) \\ &= \operatorname*{argmax}_c p(t_1|c)p(t_2|c)\ldots p(t_n|c)p(c) \end{aligned}$$

- The probability of each term t_i is defined as:

$$p(t_i|c) = \frac{frequency\ of\ term\ t_i\ in\ class\ c + 1}{sum\ of\ all\ frequencies\ in\ class\ c + |D|}$$

- $|D|$: number of distinct words in both classes

Multinomial Naive Bayes

- Finally, the a-priori probability of each class is defined as:

$$p(c) = \frac{num\ of\ processed\ tweets\ in\ class\ c}{total\ num\ of\ processed\ tweets}$$

Preprocessing Of Test Tweets

Every test-tweet has undergone the same preprocessing(stopword removal etc) as the training tweets.

Experiments

- Quick overview of our two thresholds:

$$ratio = \frac{min(P_x, N_x)}{max(P_x, N_x)}$$

$$tf_c = \frac{frequency\ of\ term\ in\ class\ c}{maximum\ frequency\ of\ class\ c}$$

- First, we set a threshold for ratio metric and discard every term above it.
- Then, among the remaining tweets in each class those with tf less than the second threshold are discarded

Experiments

- The error results of our classifier for various thresholds are:

$t1 \setminus tf$	0.1	0.2	0.5	0.7
0.9	0.32	0.36	0.46	0.464
0.7	0.33	0.37	0.43	0.464
0.5	0.34	0.38	0.45	0.458

Table 1: Errors for Various Thresholds

- As we decrease the t1, we increase the number of terms chosen for deletion according to the first filter.
- We observe that the error in this case does not change much.

Experiments

- As we increase the tf threshold, the error deteriorates as we choose to keep less and less terms.
- So the tf thresholds seems to determine the final error, and the more terms are included in the feature vector of each class, the better perfomance we get.


Thank You!