Reimplementation of *Thumbs up? Sentiment Classification using Machine Learning Techniques*

Andreea-Elena Bacanu
aeb83
Christ's College

Original authors: **Bo Pang**, **Lillian Lee**, and **Shivakumar Vaithyanathan**

Introduction

This work consists of a reimplementation of Pang et al[1] (2002)'s sentiment classification and employs two of the three standard algorithms mentioned in the paper: Naive Bayes classification with Laplace smoothing and support vector machines (SVMs). The data, 1000 positive and 1000 negative movie reviews, was provided in the framework of the Part II NLP unit of assessment.

Background

Being a reimplementation, previous work focuses almost exclusively on Pang et al's sentiment classification using a subset of the Internet Movie Database (IMDb) archive. Apart from the previously mentioned differences, we also explicitly measure the effects stemming of the word tokens has on the overall performance of each system. The programming language used is Python.

Method

The **Naive Bayes** classification follows the equation $c^* = \arg\max_C P(c|d)$ where c^* is the most probable class from the set $C = \{POS, NEG\}$ and every document is represented as a vector $\vec{d} := (n_1(d), n_2(d), \ldots, n_m(d))$ with $n_i(d)$ = the number of occurrences of feature f_i in document d. Using Bayes' rule and assuming the features f_i are independent we get:

$$P_{\text{NB}}(c|d) := \frac{P(c)(\prod_{i=1}^{m} P(f_i|c)^{n_i(d)})}{P(d)}$$

As per Pang et al, we train our model through relative-frequency estimation and employ add-one smoothing.

SVMs are *large-margin* classifiers that find a hyperplane represented by a vector $\vec{w}$ that separates the document vectors in one class from the ones in the other and for which the margin is as large as possible. The trained model then determines which side of the plane each document vector $\vec{d_i}$ in the test set falls on. For this work, we have used Joachim's SVM^{light} package[2], with all parameters set to their default values.

In addition, Pang et al explores the performance differences observed from employing **presence** as opposed to **frequency**. That is, instead of counting the number of occurrences of a word in a document, we simply acknowledge its presence, 1, or absence, 0. We have also replicated this aspect of the paper. Moreover, as per Pang et al, we studied the impact of training on **bigrams** (as opposed to only unigrams) with the hypothesis that it would capture more context.

As an extension, we have used the **Porter Stemmer**[3] to generate stems of the word tokens and analysed the performance of the 2 systems with and without stemming. **Frequency cutoffs** were also employed by excluding features with a frequency less than 3 from the feature vector.

Results

We ran 16 combinations of classifiers and conditions such as presence and stemming. Results are gathered in Table **??**.

The first run confirms Pang et al's observation, namely that Naive Bayes outperforms SVM on unigrams with feature frequencies. The difference is statistically significant under the two-tailed sign test ($p = 0.079\%$). The second run replaces feature frequency with presence. Naive Bayes worsens significantly ($p = 4.41\%$) - contrary to Pang et al's results, whereas SVM improves by an impressive 7.35% ($p = 1.09\%$). Stemming (and presence) doesn't yield a statistically significant difference in either Naive Bayes($p = 14.6\%$) or SVM($p = 8.75\%$). Adding bigrams makes Naive Bayes outperform itself (on unigrams only) by 7.05%($p = 0.17\%$).

Conclusion

This reimplementation agrees with Pang et al's in that SVM performs significantly better than Naive Bayes when frequency is replaced by presence, except when the models are trained on bigrams alone.

Word count: 500[4]

Link to code: https://github.com/andreea794/nlp-part2

[1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan (2002). *Thumbs up? Sentiment Classification using Machine Learning Techniques*. Proceedings of EMNLP

[2] http://svmlight.joachims.org

[3] https://tartarus.org/martin/PorterStemmer/python.tx

[4] Word count performed by the TexCount web interface: https://app.uio.no/ifi/texcount/online.php

features	# of features	frequency or presence	NB	SVM
unigrams	55384	freq.	**81%**	73.45%
unigrams	55384	pres.	76.45%	**80.8%**
unigrams + stemming	34285	freq.	**81.55%**	74.3%
unigrams + stemming	34285	pres.	73.15%	**81.15%**
unigrams + bigrams	539470	pres.	**83.5%**	82.2%
unigrams + bigrams + stemming	444884	pres.	82.8%	**84%**
bigrams	484086	pres.	**84.35%**	79.3%
bigrams + stemming	410599	pres.	**84.95%**	81.1%
unigrams + cutoff 3	27609	freq.	81.65%	73.45%

Table 1: Average 10-fold cross-validation accuracies. Boldface: best performance for a given setting (row).