UNIVERSITY OF CAMBRIDGE

MPhil in Machine Learning and Machine Intelligence 2021-2022

Module Coursework Feedback

Module Title: Natural Language Processing

Module Code: MLMI13

Candidate Number: J902S

Coursework Number: 1

I confirm that this piece of work is my own unaided effort and adheres to the Department of Engineering's guidelines on plagiarism. √

Date Marked: Marker's Name(s):

Marker's Comments:

This piece of work has been completed to the following standard *(Please circle as appropriate)*:

	Distinction			**Pass**			**Fail (C+ - marginal fail)**		
Overall assessment (circle grade)	**Outstanding**	**A+**	**A**	**A-**	**B+**	**B**	**C+**	**C**	**Unsatisfactory**
Guideline mark (%)	90-100	80-89	75-79	70-74	65-69	60-64	**55-59**	**50-54**	**0-49**
Penalties	**10% of mark for each day, or part day, late (Sunday excluded).**								

The assignment grades are given **for information only**; results are provisional and are subject to confirmation at the Final Examiners Meeting and by the Department of Engineering Degree Committee.

Contents

1 Introduction **3**

2 Sentiment Lexicon Baseline **3**

3 Naive Bayes **4**
3.1 Smoothing 4
3.2 Stemming 4
3.3 Unigrams vs Bigrams 5

4 SVM **6**
4.1 Comparing SVM with smoothed Naive Bayes 6
4.2 POS-tags and Discarding closed-class words 7

5 Document Embeddings **7**
5.1 SVM with document-embedding features 7
5.2 Analysis of the `doc2vec` approach to sentiment classification 8

6 Conclusions **10**

Word-count: 1979

1 Introduction

This report provides a comparative evaluation of various approaches to classifying movie reviews by sentiment. To this end, we use a dataset consisting of 2000 labelled IMDB movie reviews, evenly balanced between positive and negative reviews.

A symbolic approach via a sentiment lexicon forms our baseline, after which we explore Naive Bayes and Support Vector Machine (SVM) machine learning approaches. Here we analyse the impact of a variety of feature engineering techniques, such as applying stemming and adding Part-Of-Speech (POS) tags. Finally, we investigate the use of document embeddings for improving classification performance.

2 Sentiment Lexicon Baseline

As a baseline to which to compare machine learning approaches, we first consider a simple approach that uses a sentiment lexicon to decide whether a review contains more positive or negative words, and classify the review accordingly. We also evaluate a more fine-grained approach that weights the counts of positive and negative words by the magnitude of their sentiment.

Table 1 shows that the magnitude approach attains a marginally higher accuracy on our dataset. However, to evaluate whether the magnitude-based predictions provide a *statistically significant* improvement over the predictions from the word-only approach, we apply the sign test [SC88]. We use a p-value of 0.05 as a threshold for determining significance, as per common scientific convention (that is, there is a less than 5% chance that the difference in performance is due to noise). Our observed p-value of 0.687 thus suggests that the performance difference between the two methods is not statistically significant[1].

Accuracy		p-value	Statistically Significant (p <0.05?)
Word-only	Magnitude		
0.677	0.687	0.687	No

Table 1: Classification accuracy on the 2000 reviews in the IMDB dataset using a sentiment lexicon. According to the sign test, weighting word sentiment scores by magnitude does not provide a statistically significant different in performance compared to the unweighted scores.

An accuracy of 0.687 will thus be used as a benchmark accuracy to which to compare the more complex classifiers discussed in the following sections. To justify their added complexity, the machine learning approaches should outperform the sentiment lexicon approach.

[1]All mention of statistical significance throughout the rest of the report is in reference to the sign test.

3 Naive Bayes

In this section we investigate applying a Naive Bayes (NB) classifier with a Bag-of-Words (BoW) representation of the reviews as features. Naive Bayes classifiers are simple probabilistic classifiers that use Bayes' rule to compute the probability of each class given the features, based on a simplifying assumption that all features are conditionally independent given the class.

3.1 Smoothing

We start by training a vanilla Naive Bayes classifier on 1800 reviews. Testing on the remaining 200 reviews, we obtain a test classification accuracy of just 0.51. We would expect to achieve an accuracy of 0.5 by guessing at random, so this suggests the model has essentially failed to learn anything useful. The reason for this poor performance, is that almost all the test documents have some word that was not seen in the training data for each class. This results in a probability of 0 in the Naive Bayes class-conditional probability computation, which results in that test review being undecidable. The predicted class then is an arbitrary implementation-choice, which explains why the observed accuracy is akin to guessing.

To overcome this problem, we can apply smoothing, whereby we assume we have already seen κ pseudo-counts for each word in each class in addition to the training data. Using add-one smoothing with $\kappa = 1$, we then observe a statistically significant increase in test accuracy to 0.81 ($p < 0.001$).

Going forward, we will use cross-validation to ensure we do not overfit to test performance when changing model features and hyperparameters. This is implemented using 10-fold Round-Robin splitting to avoid potential sequential document correlations[2]. The cross-validation accuracy and standard deviation with and without add-one smoothing is shown in Table 2, and yields the same conclusions about the significant benefit of smoothing.

	Accuracy	Standard-deviation
Add-one smoothing	0.810	0.021
No smoothing	0.516	0.008

Table 2: Naive Bayes cross-validation performance with and without add-one smoothing

3.2 Stemming

One method we might consider to improve our classifier is to use stemming, whereby all words get mapped to their 'stem' word (e.g. 'running' to 'run'). The rationale

[2]All future accuracies are reported as 10-fold Round-Robin cross-validation results.

here is that stemming reduces the size of the feature-space and results in more counts per feature, which could improve the generalizability of our model and reduce overfitting. One potential disadvantage of stemming for classification, however, is the concern that some potentially helpful semantic information could be lost (such as word tenses). A sensible way to evaluate this trade-off is to test it empirically. We use this approach for many of the following experiments in the report – test whether applying particular feature-engineering techniques empirically yield statistically significant results, to support their theoretical justifications.

Table 3 shows that using stemmed features results in a marginal increase in accuracy of 0.004 over the smoothed Naive Bayes classifier, which is not statistically significant ($p = 0.893$). However, we notice that the vocabulary size is substantially reduced by almost 40% from 52,555 to 32,404, which could be useful savings if we had memory constraints for our classifier. Whilst stemming does not appear to provide significant benefit here, we do also note that this is possibly only due to the small size of our dataset, and that stemming could add more predictive power given more data.

	Accuracy	Vocabulary size
Stemming	0.814	32,404
No stemming	0.810	52,555

Table 3: Comparing the smoothed Naive Bayes cross-validation accuracy with and without stemming. The vocabulary size for stemmed features is almost half the size of the vocabulary size without applying stemming.

3.3 Unigrams vs Bigrams

To retain some sequential context information that is lost by a BoW unigrams representation of the reviews, we now consider the impact of adding bigrams and trigrams as features.

If we have n features for unigrams, we expect in the order of a linear increase in the number of features when adding bigrams (if each word is followed by on average k different distinct words, there will be $n \times k$ more features when we add bigrams). We similarly expect a further linear increase with the addition of trigrams, but at a smaller rate. This is observed empirically in Table 4 where the vocabulary size increases linearly by a factor of $k \approx 10$ from 52,555 to 500,086 when bigrams are added, but then only by a further $k \approx 3$ to 1,462,605 with the addition of trigrams.

Empirically, we see in Table 4 that whilst the cross-validation accuracy does increase by adding bigrams and trigrams, these increases are not statistically significant ($p = 0.371$ and $p = 0.434$ respectively).

	Accuracy	Vocabulary size
Unigrams	0.810	52,555
Unigrams+Bigrams	0.828	500,086
Unigrams+Bigrams+Trigrams	0.831	1,462,605

Table 4: Comparing smoothed Naive Bayes cross-validation accuracies with unigrams, unigrams+bigrams and unigrams+bigrams+trigrams as features. The vocabulary size increases linearly with the addition of bigrams, and linearly again with the further addition of trigrams.

4 SVM

One weakness of the Naive Bayes classifier is its simplifying assumption that all features are class-conditionally independent, which does not hold true in practice for sequential text data. An alternative model that is widely applied in text classification is a Support Vector Machine (SVM). The training procedure for SVMs involves finding a separating hyperplane that best classifies the observations in the feature space.

4.1 Comparing SVM with smoothed Naive Bayes

For the features for our SVM, we consider only feature presence (rather than feature frequency) and length-normalize the feature vector. This follows the recommendations made by Pang et al. who showed that this feature engineering substantially improves the performance of SVM classification of movie reviews [PLV02]. Moreover, we use a linear kernel since the dimensionality of our features (≈ 50000) is much greater than that of our data (≈ 2000).

Table 5 shows that the SVM attains an accuracy of 0.865, much higher than the smoothed Naive Bayes's accuracy of 0.810. It is a statistically significant difference ($p = 0.016 < 0.005$). This improvement can be explained by the SVM's ability to consider dependence between features, which allows for more flexibility and a better fit to the data.

	Accuracy
SVM	0.865
Smoothed NB	0.810

Table 5: Comparing the cross-validation classification accuracy of a linear-kernel SVM to the smoothed Naive Bayes classifier.

4.2 POS-tags and Discarding closed-class words

We now evaluate the impact of adding linguistic information by including the POS-tag for each word when constructing features. The idea here is that POS-tags could assist the model in disambiguating word sense. For example, the word "love" is usually associated with positive reviews when used as a verb, but is not typically indicative of sentiment when used as a noun (such as in "a love story").

We also consider whether discarding all closed-class words (everything except nouns, verbs, adjectives and adverbs) improves performance. The rationale behind this is that closed-class words (e.g. "the", "and" etc.) may not carry information useful for sentiment classification, so removing them may help the model to generalize better and reduce its ability to overfit.

Table 6 shows the classification accuracy from the above two considerations. In both cases, the marginal improvements are not statistically significant (p-values of 0.893 and 0.964 respectively). It seems that because there are very few scenarios where differing word-senses have different sentiment, adding POS-information does not help performance significantly. Moreover, given closed-class words tend to occur similarly frequently for both sentiments, removing them does not appear to have an effect on the generalizability of the model.

	Accuracy
SVM: words-only	0.865
SVM: words + POS-tags	0.867
SVM: open-class words + POS-tags	0.866

Table 6: Comparing SVM accuracies after (a) including POS-tags in input features and (b) discarding all closed-class words.

5 Document Embeddings

In this section we analyse whether we can improve the above systems by training `doc2vec` models that produce document embeddings for use as classification features.

At a high-level, a `doc2vec` document embedding is a dense vector trained to be a useful feature for softmax-classification of the words in that document. Intuitively, one can thus think of a document embedding as a vector that represents the core content of the document.

5.1 SVM with document-embedding features

We train various `doc2vec` models using the 100,000 documents in the IMDB movie review dataset [Maa+11]. The models considered include both training algorithms

used in Le and Mikolov's original `doc2vec` paper [LM14] (PV-DBOW and PV-DM) across varying dimensionalities for the document embedding vectors. Each model is then used to produce a feature vector for each review in our dataset as input to the SVM classifier discussed previously. The resulting cross-validation accuracies are displayed in Figure 1.

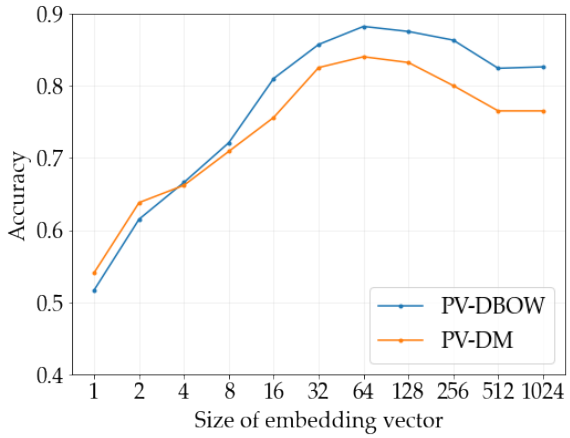

Figure 1: SVM cross-validation accuracy using document embedding features for both `doc2vec` training algorithms (PV-DBOW and PV-DM) across varying vector embedding dimensionalities. Note that the x-axis is shown on a log-scale, since varying the embedding sizes linearly in log-space allows a broader range of dimensionalities to be explored.

Firstly, we notice that the PV-DBOW models achieve higher accuracy than PV-DM across most vector embedding sizes. Interestingly, we note whilst accuracy increases initially with embedding dimensionality (as expected, since very low-dimensions are unable to represent a document sufficiently), accuracy also starts to decrease when the dimensionality is 128 or larger. This can be explained by large dimensionalities reducing the generalizability of the classifier, leading to overfitting.

The highest accuracy of 0.883 is attained using feature vectors from the PV-DBOW model with 64-dimensional vector embeddings. This is the highest out of all models we have considered, and is substantially higher than that of the smoothed Naive Bayes classifier which attained only 0.810 (a statistically significant improvement, with a very small p-value of 0.001). This demonstrates how effective document embeddings are as features for sentiment classification.

5.2 Analysis of the `doc2vec` approach to sentiment classification

We now analyse why the `doc2vec` approach succeeds (and where it may struggle) with sentiment classification[3].

[3]The PV-DBOW model with 64-dimensional embeddings is used for this analysis.

To gain intuition into what advantage embeddings have over bag-of-words features, we can look at the word embeddings learned alongside the document embeddings during `doc2vec` training. Table 7 demonstrates that word embeddings are similar in `doc2vec` space to the embeddings for other semantically similar words (e.g. "brilliant" is most similar to "fantastic" and "superb"). In a similar way, document embeddings are able to represent that a review like "brilliant movie" is semantically more similar to "fantastic movie" than "terrible movie" (as seen in Figure 2). In contrast, these three reviews are equidistant in bag-of-words feature space, which makes it more difficult for the SVM to construct its separating hyperplane.

Word	Cosine Similarity
fantastic	0.910
superb	0.894
excellent	0.887
great	0.879
amazing	0.873

a) "brilliant"

Word	Cosine Similarity
horrible	0.963
awful	0.958
bad	0.927
lousy	0.916
horrendous	0.907

b) "terrible"

Table 7: Words with the most similar embeddings to (a) "brilliant" and (b) "terrible" respectively, measured by cosine similarity. This illustrates how semantically-similar words are similar in `doc2vec` space.

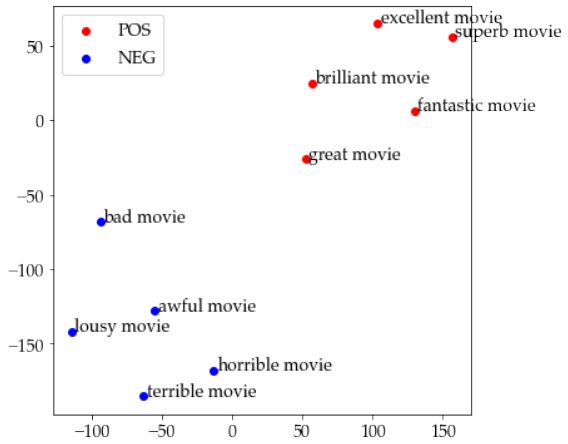

Figure 2: A two-dimensional visualization of document embeddings for short review phrases in `doc2vec` space using t-SNE [VH08]. Whilst all these reviews are equally distant in bag-of-words feature space, the distinct semantic differences between documents are well-represented in `doc2vec` space.

We now consider whether document embeddings are close in space to their most critical words, and how this relates to sentiment classification. Figure 3 shows documents embeddings for two of the shortest movie reviews in our data, along with the

word embeddings. t-SNE is used to map the high-dimensional embeddings into two dimensions for visualization purposes [VH08].

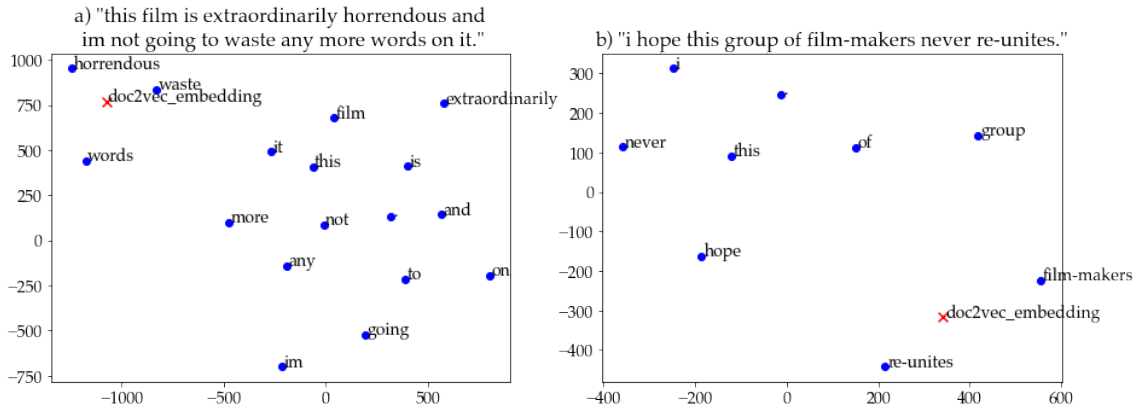

Figure 3: Document and word embeddings from a PV-DBOW model visualized using t-SNE for two short movie reviews.

In subplot (a), we notice that the review embedding is close in space to the word embeddings for 'horrendous' and 'waste', and far from the non-content words like 'on' and 'and'. The document embedding is thus close to the words that best-capture both the content and sentiment of the review, which understandably is beneficial for sentiment classification.

Subplot (b), however, gives an example of how document embeddings don't necessarily map to regions in space that are close to the words that convey the most *sentiment* information. The review embedding is close to the word embeddings for 're-unites' and 'film-makers', which are critical content words for the review. However, these words do not necessarily give insight into the review's negative sentiment. This helps illustrate one potential weakness of the `doc2vec` approach to sentiment classification – that embeddings learned are not optimized to be most useful for determining the *sentiment* of a document, but rather in an unsupervised manner to represent the key content of the document. Classification thus might be more difficult in cases when a test review is contextually similar to reviews in the training data, but opposite in sentiment.

6 Conclusions

We have discussed various approaches to classifying movie reviews by sentiment. Naive Bayes and SVM classification approaches with bag-of-words features significantly outperform a simple lexicon-based baseline approach. However, the best classification performance is attained by using document embeddings as input features into an SVM.

References

[LM14] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. PMLR. 2014, pp. 1188–1196.

[Maa+11] Andrew Maas et al. "Learning word vectors for sentiment analysis". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 2011, pp. 142–150.

[PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques". In: *arXiv preprint cs/0205070* (2002).

[SC88] S. Siegel and N.J. Castellan. *Nonparametric statistics for the behavioral sciences*. Second. McGraw–Hill, Inc., 1988.

[VH08] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).