NLP Coursework : Sentiment analysis

MPhil candidate H801D

1997 words

Preamble

In the context of our NLP course [KSBC20], we apply various NLP techniques to a sentiment analysis problem.

1 Introduction

Given two datasets of 1000 positive and 1000 negative (respectively, 25′000 positive and 25′000 negative[1]) movie reviews, we implement and compare various preexisting techniques to the task of movie review sentiment analysis, hence partially replicating [PLV02] and [LB16].

More specifically, we test and compare the Sentiment Lexicon (SL), the unsmoothed and smoothed Naive Bayes (NB) classifiers (see [PLV02]) and Support Vector Machine (SVM, see [Joa99]) approaches with different choices of features on the smaller database, as well as the smoothed NB, SVM and the Paragraph Vector or Doc2vec (see [LM14]) algorithm on the larger one.

On any given task and dataset, methods can be tuned and combined to achieve increased precision (see [BMMR15]), in a way that often generalizes poorly. This is not our objective here; we wish to compare general ideas, as in [PLV02]. As a result, we do not try to overly optimize parameters[2], though we choose reasonable ones[3].

For linguistic concepts, see [JM08]. Due to strict length constraints, we do not redefine NLP methods but give proper references.

2 Background

Pang et al introduced the movie review sentiment classification task in [PLV02] and found accuracy rates of about 69% for the SL method, between 77% and 81.5% for the smoothed NB, and between 72.8% and 82.9% for the SVM. All those methods use sparse vector representation.

The Doc2vec method was introduced in [LM14] as an improvement on the Word2vec method (see [MSK+13]). Using neural networks, it yields a dense vector representation of both words and paragraphs (the reviews, in our case). Two main versions exist: Distributed Bag of Words (DBOW) and Distributed Memory (DM). One can also choose to either sum, average or concatenate vectors at various steps of the algorithm.

According to [LM14], the DM version is superior, with best results being achieved by com-

[1] From [MDP+11].

[2] By testing dozens of smoothing constants and vector sizes, omitting certain punctuation signs, etc.

[3] *e.g.* similar to those found in the literature.

[4] Including T. Mikolov, one of the authors of [LM14] - see footnote on page 4 of [BMMR15].

bining the two, for an accuracy of about 92.6%. However, others[4] have struggled to replicate those results. Moreover, [LB16] suggests that DBOW, though simpler, is the superior technique (though they do not treat the case of sentiment analysis).

3 Method

We completed the provided Python code, and ran it on an Intel® Core™ i7-8650U Processor under Windows; we provide all code as an annex.

For all statistical comparison between two methods, we use the sign test[5] (see [SC88]) with "both methods are as accurate" as the null hypothesis (which corresponds to the two-tailed version).

We start with the small database. Our sentiment lexicon comes from [WWH05]. We implement both binary and weighted versions, with a weight of 1 for low magnitude words and of 3 for high magnitude ones and the suggested threshold of 8 for both (the weighted score must be greater or equal to 8 for the review to be predicted as positive).

We then compare unsmoothed NB and smoothed NB, with a smoothing coefficient of 1 (as in [PLV02]).

We also run both smoothed NB and SVM, using T. Joachims' SVM implementation (see [Joa99]) with default parameters, on the small dataset with the following features[6]: unigrams - bigrams - trigrams - unigrams, bigrams and trigrams - stemmed unigrams (using Porter Stemmer, see [Por80]) - unigrams and part-of-speech (POS) tags[7] - open-classes (see [JM08]) only unigrams and POS tags. We cross-validate with Round-Robin splitting (10 folds) for increased reliability.

On the larger database, we use the Gensim implementation (see [ŘS10]) of Doc2vec. We train the Doc2vec algorithm on all 50'000 reviews (previously randomly shuffled) with stemmed unigrams[8] as features (as it is unsupervised, we do not differentiate between training and test sets), then use it to infer a vector embedding for each review. We choose not to normalize those vectors, as vector norms also carry information (in particular related to frequency) - see e.g. [SW15]. We then train a SVM on 25′00 reviews, and test it on the 25′000 other reviews (other classifiers could have been used on the embedded vectors). We do not cross-validate, nor do we use the additional 50′000 unlabelled reviews of the second database to further train the Doc2vec models, as both the training and testing set are already very large.

Among other parameters [9], we can choose models DBOW or DM, whether the Doc3vec neural network sums or concatenates vectors if using DM, whether to simultaneously train word vectors if using DBOW, window size $W \in \mathbb{N}$, embedded vector size $d \in \mathbb{N}$, whether to use negative sampling (NS) or hierarchical soft-max (HS) when training the Doc2Vec model, and the number of training iterations $T \in \mathbb{N}$.

We test the following parameters settings[10]:

[5] There are more sophisticated tests available, such as the trinomial test that takes draws into account (see [GMW09]), but the results are clear enough here.

[6] Except for SVM and trigrams or unigrams+bigrams+trigrams, as the number of features becomes too large for SVM to run in a reasonable time.

[7] Using the tags provided with the database.

[8] Both because it seems to slightly improve performance, and to reduce the number of features and hence running times.

[9] See [LM14] for a detailed explanation of the meaning of each parameter.

[10] Those were suggested by our reading of [LM14], [LB16] and the online Gensim documentation [ŘS10]. A fully systematic study would involve testing each reasonable combination of parameters, but running all configurations would take hundred of hours without a better computer and more optimized code.

1. DBOW, $W = 15$, $d = 30$, NS, $T = 50$. Based on [LB16][11], this seems like a reasonable base case.

2. DBOW, $W = 15$, $d = 300$, NS, $T = 50$.

3. DBOW with simultaneous word vectors training, $W = 15$, $d = 30$, NS, $T = 50$.

4. DBOW, $W = 15$, $d = 30$, HS, $T = 50$.

5. DM with sum, $W = 5$, $d = 30$, NS, $T = 100$.

6. DM with concatenation, $W = 5$, $d = 30$, NS, $T = 100$.

We also ignore all words that only appear once. All other parameters (including the linearly decreasing learning rate) are the default ones for the Gensim implementation. We use PCA (see for example [Jol02]) and the Tensorboard Embedding Projector [Goo] to visualize the inferred vector representations.

Additionally, we run smoothed NB and SVM on the large database (also with stemmed unigrams as features) as points of comparison.

4 Results

We consider the results on the small database in Subsection 4.1, and those of the Doc2vec algorithm on the large database in Subsection 4.2. All numerical results (except counts) are approximated.

4.1 Sentiment lexicon, Naive Bayes classifier and SVM

We see in Table 1 that the SL method achieves reasonably good results, in line with the 69% accuracy found in [PLV02]. Though adding weights appears to have improved the accuracy, the difference between regular and weighted SL is statistically insignificant, with $p \cong 0.69$

Sentiment lexicon	Weighted sentiment lexicon
67.4	68.3

Figure 1: Accuracies in % of both standard and weighted SL on the 2000 reviews database, with threshold of 8 and weights of 3.

In Table 2, we see that the non-smoothed NB classifier performs very poorly, with 5% accuracy[12], as it assigns probability 0 of being positive to a review containing a word that has never appeared in a positive review in the training set (and similarly for negative). This model is severely overconfident: new words (in particular names) are bound to appear. The smoothed NB classifier, with 82.5% accuracy, is significantly superior to it, with $p < 10^{-5}$ (using cross-validation to obtain 2000 predictions).

It also significantly outperforms the SL approach ($p < 10^{-4}$). This is not surprising: though both methods are crude and ignore word order (and in particular fail to account for negation, contrast and irony), the smoothed NB classifier learns from the data, while SL does not. Henceforth, we refer to smoothed NB with unigrams as simply NB, and use it as our new baseline.

NB classifier	Smoothed NB classifier
5.0	82.5

Figure 2: Averaged accuracies in % of non-smoothed and smoothed NB classifiers on the 2000 reviews database using cross-validation, with smoothing coefficient of 1.

In Table 3, we compare the mean accuracy and standard deviation (from cross-validation) of the NB and SVM classifiers for different types of

[11] Which in particular states that low vector dimension can improve performance in the special case of sentiment analysis.

[12] The exact accuracy depends on conventions: here, if in our trained model both $P(w|r \text{ is positive}) = 0$ and $P(w|r \text{ is negative}) = 0$ for word w in review r, we consider that the prediction is incorrect.

Bag-of-words methods applied to the 2000 reviews database					
		NB		SVM	
features	# of features	Mean acc.	Std dev.	Mean acc.	Std dev.
unigrams	16'278	81.4	1.8	86.4	2.3
stemmed unigrams	10'924	81.6	1.9	86.4	2.2
bigrams	81'536	84.6	2.2	84.4	3.0
trigrams	132'279	83.7	2.9	-	-
1-,2- and 3-grams	230'274	82.9	1.9	-	-
unigrams + POS	18'130	81.8	1.7	86.7	2.6
OC unigrams + POS	17'537	81.6	1.4	86.8	2.1

Figure 3: Averaged accuracies in % and standard deviations in % of smoothed NB and SVM classifiers on the 2000 reviews database using cross-validation. "1-,2- and 3-grams" means using unigrams, bigrams and trigrams simultaneously as features. "POS" means adding part-of-speech tags. "OC unigrams" stands for open-classes only unigrams (discard all closed-classes words, see [JM08]). "# of features" is the number of considered features in the held-out set.

features, as well as the number of such features present in the held-out set of 200 reviews.

For the NB classifier, no choice of features significantly outperforms using unigrams: comparing NB with unigrams only to NB with bigrams only, the highest accuracy recorded, we get $p \cong 0.158$.

Similarly, no choice of features significantly outperforms unigrams for the SVM classifier (we discuss this in Section 5). However, SVM is significantly more accurate than NB using unigrams, with $p \cong 0.025$. Those results are in line with those from [PLV02].

Though more precise, SVM is considerably slower than NB, in particular for large number of features: running time tends to scale very poorly with the number of features (according to [Joa99]), which kept us from testing trigrams and combinations of unigrams, bigrams and trigrams.

A priori, if V is the number of existing unigrams, there are V^2 possible bigrams and V^3 possible trigrams. In practice, semantic and syntaxic rules ("constitutional banana" and "I you he" never appear) and the size of the corpora strongly limit the number of observed n-grams. Indeed, we see in Table 4 that there are about 5 times more bigrams than unigrams and 8 times more trigrams; moreover, the number of bigrams and trigrams grows faster in the size of the corpus than the number of unigrams[13].

# of reviews	200	2′000	25′000
# of unigrams	16'278	55'379	194'751
# of bigrams	81'536	484'078	2'347'518
# of trigrams	132'279	1'058'186	6'784'843

Figure 4: Evolution of the number of unique unigrams, bigrams and trigrams in three different corpora of reviews.

4.2 Paragraph vectors

In Table 5, we see the accuracies and running times[14] of our different Doc2vec configurations.

[13]More precisely, here the number of unigrams approximately grows as $C_1 k^{0.5}$, the number of bigrams as $C_2 k^{0.75}$ and the number of trigrams as $C_3 k^{0.9}$, where $C_i > 0$ are constants and k is the number of reviews - of course, these relations cannot hold for k very large or small.

[14]The running times can vary slightly from one run to another; they are mostly meant to allow general comparisons.

Model	Accuracy	Time
DBOW, $W = 15$, $d = 30$, NS, $T = 50$	89.0	1793
DBOW, $W = 15$, $d = 300$, NS, $T = 50$	87.7	2315
DBOW, Word vectors training, $W = 15$, $d = 30$, NS, $T = 50$	87.0	5604
DBOW, $W = 15$, $d = 30$, HS, $T = 50$	88.6	2864
DM, Sum, $W = 5$, $d = 30$, HS, $T = 50$	82.5	2592
DM, Concatenate, $W = 5$, $d = 30$, NS, $T = 100$	86.4	7021
Naive Bayes classifier	80.4	15
SVM classifier	88.2	2451

Figure 5: Accuracies in % and running time in seconds of various Doc2vec configurations (see Section 3 for details), as well as a NB and a SVM classifiers, on the large database of 50′000 reviews.

As predicted by [LB16], we fail to replicate the level of precision claimed in [LM14] - our results are closer to those reported in the documentation of Gensim [ŘS10]. Moreover, DBOW with low vector dimension and negative sampling does obtain the best accuracy, at 89%, and in particular significantly beats the best DM accuracy, with $p < 4 \cdot 10^{-5}$, while also being the fastest Doc2vec configuration. It significantly beats the NB classifier as well, with $p < 10^{-5}$ (though it is considerably slower), but does not significantly outperform the SVM classifier alone, with $p \cong 0.21$ (though it is slightly faster).

The only surprise here is that it is claimed in [LB16, Section 5] that running the DBOW model without simultaneously training word vectors severely decreases accuracy. In contrast, it seems to improve it here (as well as running time). It could be due to either the nature of the task ([LB16] does not consider sentiment analysis), an artifact of the datasets, or a subtlety in the Gensim implementation of that option. Further experiments would be needed to conclude.

In Figure 6, the paragraph vectors of a subset of 4′000 reviews obtained using DBOW with simultaneous word vectors training are represented in $3D$ using PCA; we can see that negative reviews (in red) and positive reviews (in blue) tend to cluster, though they fail to be properly separated in this projection[15].

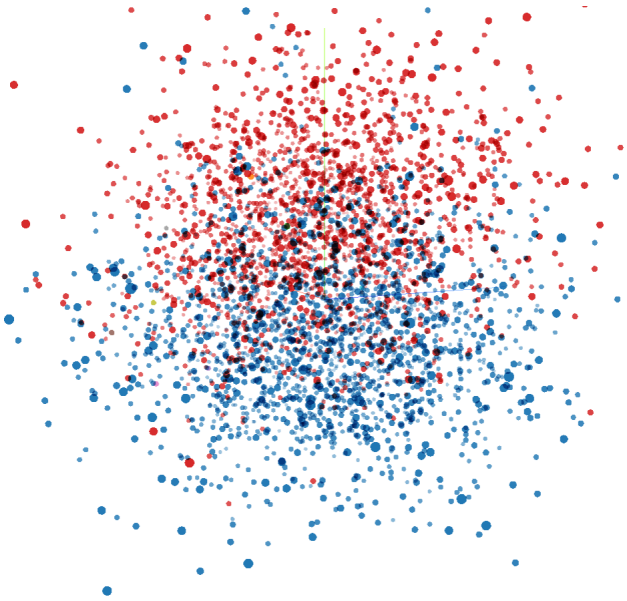

Figure 6: $3D$ representation using PAC of paragraph vectors for 4′000 reviews. Positive reviews are blue, negative ones are red.

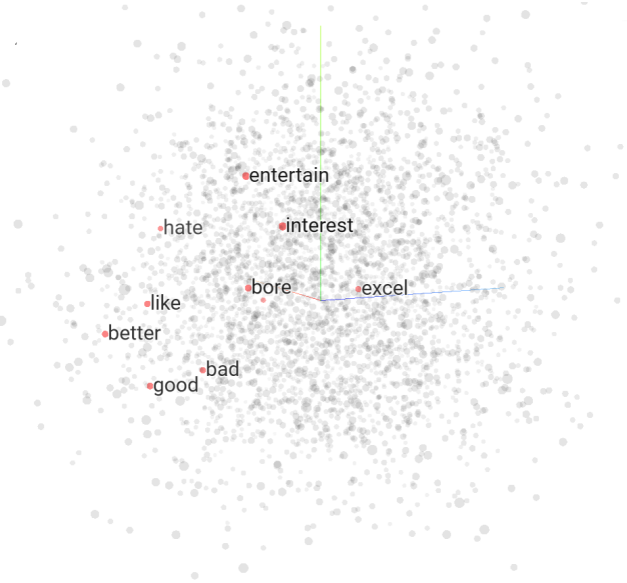

Figure 7: Vector representations of positive and negative words in the same model and from the same point of view as Figure 6.

[15]Note that results depend greatly on the projection method used - one can reach almost any conclusion by selecting Tensorboard parameters carefully enough.

Figure 7 is taken from the exact same point of view; it additionally represents the vectors of a few positive or negative stemmed words. We see that though reviews are loosely organised by sentiment, it is not the case for words.

However, this does not mean that they are randomly scattered: as illustrated in Figure 8 (from another point of view), antonyms appear close to each other, and there is even a degree of *compositionality* of meaning: "like - hate ≅ better - worst". See [MSK+13] and [LCHJ16] for more on visualization of NLP models.

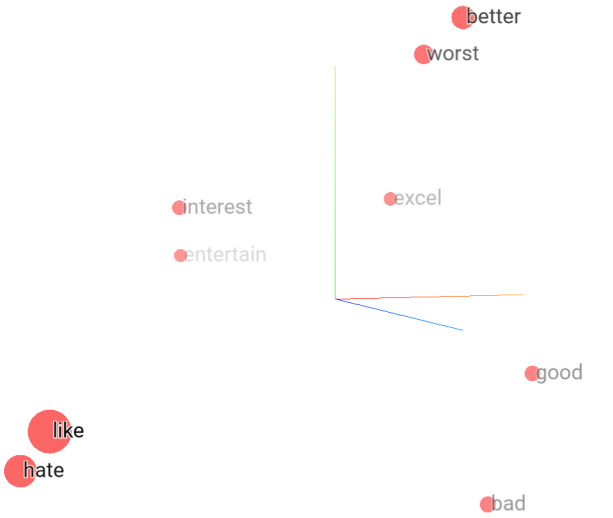

Figure 8: Vector representations of positive and negative stemmed words in the same model from another point of view than in Figure 7.

5 Discussion

Our findings are as expected based on [PLV02] for SL, NB and SVM on the small dataset, as well as for the Doc2vec algorithm on the large dataset based on [LB16], except that adding word vectors computations to the DBOW algorithm decreases precision instead of increasing it. Fine-tuning parameters and combining models, as in [BMMR15], would likely allow to gain a few percents of accuracy.

We see that NB classifiers outperform SL classifiers, surely because they learn from the data. SVM themselves are more accurate than NB classifiers, perhaps because they do not work under an assumption of features independence. SVM and NB classifiers are both limited by their use of bag-of-words representations that do not account for word order - any grammatical structure goes undetected.

Improving upon that seems difficult. Using more sophisticated features does not seem to help, perhaps because better features provide better local information, while we actually need better global information (on the scale of the sentence, or the entire document) to capture key-phenomena such as negation, opposition or contrast effects (the "thwarted expectations" mentioned in [PLV02]).

Doc2vec models are much better at capturing word meanings and semantic relations, even allowing for "meaning algebra" (see [MSK+13]), as mentioned before. However, it only translates to a small improvement at best in accuracy in sentiment analysis; they accurately represent words meanings, which is very useful to identify what a text is about (as demonstrated in [DOL15]), but the only relevant topics here are roughly speaking "positive" and "negative", which NB classifiers already identify accurately; what matters is how these topics are articulated in the text. In other words, Doc2vec might also fail to capture global phenomena[16].

Though understanding all subtleties (such as irony) of a review is most likely AI-complete, adding constituency or dependency parsing to our models to endow them with an increased understanding of the interactions between words might allow for some progress.

[16]This idea is supported by the fact that Doc2vec DM models do not outperform DBOW models, even though the latter discard local word order, which indicates that whatever additional (and necessarily local) relations that DM models capture might not be relevant.

References

[BMMR15] Y. Bengio, G. Mesnil, T. Mikolov, and M. Ranzato. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *International Conference on Learning Representations*, 2015. `https://arxiv.org/pdf/1412.5335`.

[DOL15] A. Dai, C. Olah, and Q. Le. Document embedding with paragraph vectors, 2015. `https://arxiv.org/pdf/1507.07998.pdf`.

[GMW09] B. Guorui, M. McAleer, and W. Wong. A trinomial test for paired data when there are many ties. *Mathematics and Computers in Simulation*, 81, 2009.

[Goo] Google. `https://projector.tensorflow.org/`.

[JM08] D. Jurafsky and J. Martin. *Speech and Language Processing, 2nd edition.* Prentice Hall, 2008.

[Joa99] T. Joachims. Making large-scale SVM learning practical. *Advances in kernel methods: support vector learning*, 1999. `http://svmlight.joachims.org/`.

[Jol02] I. Jolliffe. *Principal Component Analysis.* Springer Series in Statistics, 2002.

[KSBC20] K. Knill, W. Sun, P. Buttery, and A. Caines. Natural language processing course, 2020.

[LB16] J. H. Lau and T. Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016.

[LCHJ16] J. L, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in nlp. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, 2016.

[LM14] Q. Le and T. Mikolov. Distributed representations of sentences and documents. *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196, 2014.

[MDP+11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[MSK+13] T. Mikolov, I. Sutskever, C. Kai, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 2013.

[PLV02] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? senti- ment classification using machine learning techniques. *Proceedings of EMNLP*, pages 79–86, 2002.

[Por80] M. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 1980. `https://tartarus.org/martin/PorterStemmer/`.

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.

[SC88] S. Siegel and N. Castellan. *Nonparametric Statistics for the Behavioural Sciences*. McGraw-Hill, 1988.

[SW15] A. Schakel and B. Wilson. Measuring word significance using distributed representations of words, 2015. `https://arxiv.org/abs/1508.02297`.

[WWH05] T. Wilson, J. Wiebe, and P. Homann. Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of HLT-EMNLP*, 2005.