Revisiting the Baselines for Distributed Representation of Sentences and Documents

Linck Wei 260796147 **Zhiyue Jiang** 260669404 **Mahad Khan** 260678570

April 2019

Abstract

Machine learning is a domain that is constantly evolving; more and more new models have been introduced in recent years, and they have the potential to outperform traditional approaches. For example, Paragraph Vector [9] is a recently proposed approach for processing text, and it can be used in machine learning tasks such as sentiment analysis and predicting the next words in a sentence. The authors of Paragraph Vector claimed that this new approach has lower error rates than traditional baselines such as n-grams and support vector machine(SVM). In this project, we examined whether this claim holds, by implementing and fine-tuning the traditional baselines mentioned in their publication. Our experiments found that the fine-tuned version of these baselines yields the same accuracy or a slightly higher accuracy than the one reported in the authors' paper. However, Paragraph Vector still has the lowest error rate. Therefore, we can confirm that paragraph vector is an innovative and effective approach.

1 Introduction

Text classification gives us the ability to assign emotions or sentiments to text documents. It can be used in various applications that require human interaction. Another important machine learning task is to predict a word given the other words in a context. Algorithms such as logistic regression and K-means are generally used to approach these Natural Language Processing (NLP) tasks. These algorithms typically require the text input to be fed in as fixed-length vectors. Common approaches to convert text into fixed-length vectors are using bag-of-words or bag-of-n-grams [6]. Although these approaches are simple to implement and provide good accuracy, they have disadvantages too. They have very little sense of the semantics of the words and also lose the order of words.

In their paper [9], Quoc Le and Tomas Mikolov propose Paragraph Vector as an approach to overcome these disadvantages. Paragraph Vector, when trained by the stochastic gradient descent and backpropogation [20], can be applied to texts of variable-length such as phrases, sentences and even large documents. Their research indicates that this method used along with logistic regression has a lower error rate than the more general models like Naïve Bayes, SVMs, Bigram Naïve Bayes and Word Vector Averaging [25]. The results of their experiments show that there is a relative improvement of 16% on the sentiment analysis task when performed on the Stanford sentiment treebank dataset [25] as compared to the baseline models.

The Stanford sentiment treebank (SST) dataset includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 single sentences taken from the movie review site Rotten Tomatoes. Each sentence in this dataset has an associated label that ranges from very negative to very positive in the scale from 0.0 to 1.0. The SST dataset presents new challenges for sentiment compositionality and has become a benchmark for sentiment analysis.

In this project, we reproduce the results of a subset of the baseline models suggested by [9] on

Proceedings of Group 20 for McGill University's 2019 Winter semester COMP 551 - MiniProject 4

the SST dataset. `Scikit-learn` [17], a Python package that aids machine learning tasks is used to build, train and validate the Naïve Bayes, SVM, Bigram Naïve Bayes and Word Vector Averaging models. We then attempt to fine-tune these baselines by doing extensive hyper-parameter tuning.

This report is structured as follows: in Section 2 we discuss some related work to the distributed representations of sentences and documents, in Section 3 and 4, we outline the dataset used and our implementation of the baseline models and finally in Section 5 and 6 we provide and discuss the results of our models and future directions.

2 Related Work

The task of distributed representation for words was first proposed in 1986 [20] and has since become a successful paradigm, especially for statistical language modeling [4]; [1]; [11]. Over the years, we have come to realize that traditional approaches to NLP, such as one-hot encoding and bag-of-words models do not capture information about a word's meaning or context. To provide machines more information about words, word vectors have been used to make technologies like speech recognition and machine translation [31] possible. Word vectors are being used in a variety of NLP applications such as word representation, named entity recognition, word sense disambiguation, parsing and tagging [2]; [27]; [26]; [3]; [23]; [8].

There has been recent work that uses neural networks to learn vector representation of words [1]; [2]; [15]; [26]; [12]; [13]. The general approach to these models is to represent each word by a vector that is either concatenated or averaged with other word vectors in a context, and use the resulting vector to predict other words in the same context. One example is the proposed approach of [1] that predicts the next word. To do so it feeds the concatenation of several previous word vectors to the neural network. Once this model is trained, semantically similar words have similar vector representations in the resulting vector space.

Researchers have also extended previous word level models to achieve phrase-level or sentence-level representations [14]; [30]; [28]; [5]; [13]). An approach as proposed by ([23] uses matrix-vector operations to combine word vectors in an order given by a parse tree of a sentence. Another simpler approach is using a weighted average of each word in the document. However, both these approaches have weaknesses.

Socher et al. have also done extensive experimenting on distributed representations of phrases and sentences [22]; [24]; [25]. Their methods require more labeled data to perform well because they are supervised. Typically, their proposed methods require parsing and work for sentence-level representations. The findings of [25] on the SST dataset suggest that their Recursive Neural Tensor Network outperforms the bag-of-words model. This can be because of the rather small size of the training set, and also because the semantics of words as well as the short nature of movie reviews play an important role in classifying whether a review is positive or negative.

Paragraph Vectors [9] is another competitive approach to the NLP task of distributed representations of sentences and documents. It is an unsupervised framework that learns continuous distributed vector representations for variable-length pieces of texts such as sentences and documents. It does not require the word weighting functions to be tuned according to the task nor does it rely on parse trees. This model can predict the following word in a context and it does so by concatenating several word vectors from a paragraph with the paragraph vector. Paragraph Vectors capture the semantics of paragraphs well and have the potential to overcome many weaknesses of the bag-of-words model. The word matrix (set of words associated with their word vector) that is used to make predictions is always the same, since all word vectors are learned from the same entire corpus. However, if we want to predict the next words in a paragraph, this model only uses the paragraph vector representing this specific paragraph. The results of using Paragraph Vector for the sentiment

analysis task on the SST dataset as reported in [9] indicate that this approach outperforms all the baselines including recursive networks even though it does not require parsing. As mentioned in their paper, their method has an absolute improvement of 2.4% in terms of error rates on the coarse-grained classification task and this translates to a 16% relative improvement.

3 Datasets

In this section, we briefly discuss the two datasets we will use for our experiments. In Section 5 we will compare our results with the published results based on these datasets.

3.1 Stanford Sentiment Treebank

The SST dataset has now become a benchmark for sentiment analysis. This dataset was first proposed by [16] and was later extended by [25]. The dataset can be downloaded at: `http://nlp.Stanford.edu/sentiment/`. This dataset consists of 11,855 reviews from the movie review site Rotten Tomatoes. The reviews are split across training, development (or validation) and test sets, containing 8,544, 1,101, and 2,210 reviews respectively. The dataset has been annotated for five levels of sentiment: strong negative, negative, neutral, positive, and strong positive. It has been annotated at the clause level, where each node in a binary tree is given a sentiment score, as well as at the sentence level. In total, there are 239,232 labelled phrases in the dataset. For this project, we perform the classification task in two ways and compare the results. We consider a 5-way fine-grained classification task where the reviews are annotated for the five levels of sentiment. We also consider a binary coarse-grained classification task where the labels are {Negative, Positive}.

3.2 IMDB

The Stanford Large Movie Review (IMDB) dataset was first proposed by [10] as a benchmark for sentiment analysis. This dataset consists of 100,000 movie reviews from IMDB. What differentiates this dataset from the SST dataset is that each review in this dataset has several sentences. The dataset can be downloaded at: `http://ai.Stanford.edu/~amaas/data/sentiment/index.html`. 50,000 of the reviews are "highly polar", binary labeled reviews that are split into 25,000 labeled training instances and 25,000 labeled test instances. The dataset also consists of 50,000 unlabeled reviews which may be used for unsupervised learning. The findings of [7] tell us that the reviews averaged 267.9 tokens in length with a standard deviation of 198.8 tokens.

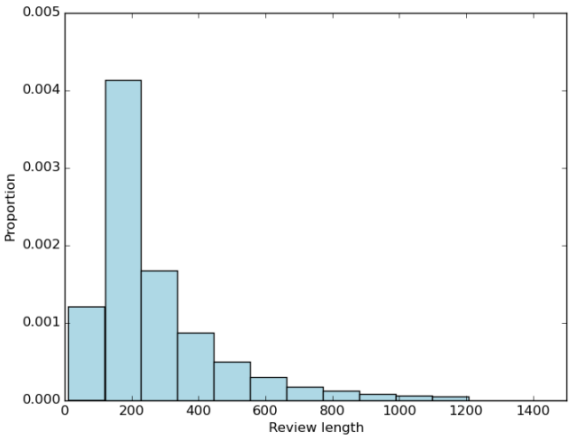

Figure 1: The distribution of length of reviews in the IMDB dataset

4 Models

4.1 Features

Before moving on to the classifiers, we should first introduce the feature extraction method over text. As known to us, text is a sequence of strings, which cannot be fed into the classifiers directly. So, there are 3 popular ways:

- BoW(Bag-of-Words): it represents a text as a d-dimensional vector with the entries indicating how many times that a token appears or not (so, each entry ranges 0 or a positive integer),

in which d is the number of the unique tokens that appear in the dataset. Note that a token can be a word (i.e., uni-gram model) or multiple words (i.e., bi-gram or even n-gram model).

- One-hot: it is the special case of BoW, which binarize the feature derived from the BoW methods. In other words, it just focuses on whether a token appears or not.

- Word vectors: as we can see, if the vocabulary of the dataset is large, i.e., d is large, then it will costs a lot of memory. To overcome this problem, word vectors use a unsupervised model to learn a distributed representation of a word. So a text can be represented as the average of the word vectors.

In the following, we will use Naïve Bayes and SVM over One-hot features, and SGD classifier over the averaging word vectors. Note that, in the field of text processing, especially facing to sentiment classification problem, the text shows a strong linear characteristic. So, it is reasonable to use One-hot feature and linear kernel SVM models.

4.2 Naïve Bayes

Naïve Bayes is a simple baseline. It assumes that all features are independent from each other [19]. Due this assumption, all features can be learned independently, and this makes computation easy and fast. However, this assumption is often not true. For example, if the words in a sentence are related to each other, the accuracy of this model can be impacted by this wrong assumption.

Therefore, Naïve Bayes would intuitively give a low accuracy in tasks such as word prediction. However, there are some ways to improve it. For example, bi-gram Naïve Bayes reduces the negative effect of the independence assumption since every pair of words are considered [18], so at least the words in the pair are not assumed to be independent from each other.

Naïve Bayes can be fine-tuned by introducing the smoothing feature. The likelihood can be calculated from a smoothed version of maximum likelihood, shown in the following equation:

$$\theta_{yi} = (N_{yi} + \alpha)/(N_y + \alpha n)$$

In this equation, N_{yi} is the number of times the feature i appears in a sample belonging to the class y, and N_y is the size of that sample. α is the parameter for smoothing. If $\alpha = 1$, it is called Laplace smoothing, and if $\alpha < 1$, it is called Lidstone smoothing. If $\alpha = 0$, it is equivalent to no smoothing. [29] Smoothing can prevent this equation to yield zero, even if the feature is not present in the sample.

For example, using bi-gram Naïve Bayes (coarse) in the sentiment analysis task with the SST dataset, we tested 50 values for α, starting from 0, and increasing by 0.02 until it reaches 1, and at the end we selected the best value that minimized the error rate. The final error rate is reduced to 0.181, which is approximately the same as than the error rate of 0.182 reported in the original paper about paragraph vector [9]. Therefore we found that the authors of paragraph vector already employed smoothing.

Another point which needs to be addressed is the distribution form of the features. In the most common case, Naïve Bayes assumes that each feature is substitute to a Gaussian distribution. However, as we use the One-hot feature over words, the feature vectors are either 0 or 1. So, we turn into Multi-nomial distribution, which is suitable for multiple binary random variables.

We will re-produce and improve Multi-nomial Naïve Bayes with One-hot features over uni/bi-grams. Please refer to Section 5 for details.

4.3 Support Vector Machine(SVM)

Conceptually, SVM is a method that tries to separate data points using hyper-planes into categories such that the error is minimized. Since not all data points can be linearly separable, soft SVM is employed, and

some wrongly classified data points are tolerated. [21]

In this model, there are some hyper-parameters that can be fine-tuned. For example, the penalty term decides how strongly we punish wrongly classified data points. Theoretically, a very small penalty term does not punish wrongly classified data points enough, so the model tends to maximize the margin of SVM at the cost of allowing some wrong classifications. On the other hand, if the penalty term is too large, there will be less wrong classifications, but the margin of SVM will be smaller.[21] Therefore, what is important in soft SVM is to find the best value for the penalty term such that there is an optimal balance between the size of the SVM margin and the tolerance for wrong classification.

We tested nine values for the penalty term, ranging from 0.01 to 10. We found that, in general, the error rate is reduced if we increase the value from 0.01 to 0.2, but the error rate is increased if the value goes over 0.2.

For example, in the sentiment analysis task with the SST dataset, the SMV(coarse) with the penalty term = 0.2 yields an error rate of 0.2037, which is slightly better than the error rate reported in the paper about pargraph vector.

4.4 Word Vector

Word vector is a method in which every word is associated with a vector representation. In the prediction task, the vectors of the existing words in a sentence are concatenated or averaged in order to produce a combined vector, which is then passed to a multi-classifier in order to predict the next word, or the sentiment of the sentence, or any target of prediction. [10]

Word vectors can be initialized randomly, or in any method. At the end of learning, vectors that represent the words of similar meanings will be mathematically similar with each other in the vector space. Thus, the semantics of words are taken into account. For example, the words "strong" and "powerful" will have similar word vectors. [10]

A disadvantage of word vector is that prediction is only made using the words from the same sentence, thus the context in which the sentence has been written is lost. Other surrounding sentences in the same paragraph may also contain useful information, but they are ignored.

One way to optimize this model is to fine-tune the multi-classifier in which we pass the concatenated word vector. In our experiment, we used the SDG classifier with L2 regularization. The penalty term in L2 regulation is multiplied by a constant λ. We adjusted this λ in order to adjust how strongly we penalize the model if the weights of the SDG classifier are too large.

For example, we tried 10 values for λ, ranging from 0.001 to 0.01, we found that 0.001 is the best value in the sentiment analysis task performed on the SST dataset(coarse). Our error rate was better than the error rate in the paper about paragraph vector.

5 Results

5.1 SST

Following the baselines in [9], we re-produce and report our results here:

Table 1: baselines for the SST dataset. Note that the results from [9] is also shown in parentheses for fair comparison.

Model	Error rate (Pos/Neg)	Error rate (Fine-grained)
Naïve Bayes	18.07%(18.2%)	59.54%(59.0%)
SVMs	20.37%(20.6%)	59.05%(59.3%)
Bigram Naïve Bayes	17.63%(16.9%)	60.00%(58.1%)
Word Vector Averaging	23.83%(19.9%)	61.17%(67.3%)

A large part of our re-implementation obtains an

improvement than before, which proves that our re-implementation is correct. According to the feature extraction methods and parameter tuning procedure, we can conclude that the separator of the dataset is nearly linear. However, there also exists some hard examples which cannot be modelled with simple baselines.

5.2 IMDB

Following the baselines in [9], we re-produce and report our results here:

Table 2: baselines for the IMDB dataset. Note that the results from [9] is also shown in parentheses for fair comparison.

Model	Error rate
MNB-uni	15.65%(16.45%)
MNB-bi	13.38%(13.41%)
SVM-uni	12.39%(13.05%)
SVM-bi	10.76%(10.84%)

All of our re-implementation obtains an improvement than before, which proves that our re-implementation is correct. Note that IMDB dataset has not development dataset, so the parameter is set to be default. However, the performance is still higher than before, indicating that our understanding of the dataset is correct.

6 Discussion and Conclusion

In this report, we re-implement the baselines on two datasets and try our best to improve the performance of these simple models. As a conclusion, most of the baselines can be improved to obtain a lower error rate. However, simple models cannot capture some higher-order relationship among features and labels. So, there is also urgent need to develop some modern models.

We optimized Naive Bayes, support vector machine, and word vector models using smoothing, penalty term for misclassification, and L2 regularization respectively. While some of these optimized baselines achieved a slightly lower error rate than the one reported in the original paper about paragraph vector , the paragraph vector model still has the lowest error rate. Therefore, paragraph vector is an effective approach.

A Statement of Contributions

Linck Wei - Research about models, Analyze Results, Writing report
Zhiyue Jiang - Programming models, Optimizing models, Writing report
Mahad Khan - Research about models, Analyze Results, Writing report

References

[1] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. A neural probabilistic language model. *J. Mach. Learn. Res. 3* (Mar. 2003), 1137–1155.

[2] Collobert, R., and Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning* (New York, NY, USA, 2008), ICML '08, ACM, pp. 160–167.

[3] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res. 12* (Nov. 2011), 2493–2537.

[4] Elman, J. L. Finding structure in time. *COGNITIVE SCIENCE 14*, 2 (1990), 179–211.

[5] Grefenstette, E., Dinu, G., Zhang, Y.-Z., Sadrzadeh, M., and Baroni, M. Multi-step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers* (Potsdam, Germany, Mar. 2013), Association for Computational Linguistics, pp. 131–142.

[6] Harris, Z. Distributional structure. *Word 10*, 23 (1954), 146–162.

[7] Hong, J., and Fang, M. Sentiment analysis with deeply learned distributed representations of variable length texts.

[8] Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1* (Stroudsburg, PA, USA, 2012), ACL '12, Association for Computational Linguistics, pp. 873–882.

[9] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (2014), ICML'14, JMLR.org, pp. II–1188–II–1196.

[10] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (Stroudsburg, PA, USA, 2011), HLT '11, Association for Computational Linguistics, pp. 142–150.

[11] Mikolov, T. *Statistical language models based on neural networks*. PhD thesis, Brno University of Technology, 2012.

[12] Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. Efficient estimation of word representations in vector space, 2013.

[13] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (USA, 2013c), NIPS'13, Curran Associates Inc., pp. 3111–3119.

[14] Mitchell, J., and Lapata, M. Composition in distributional models of semantics. *Cognitive Science 34*, 8 (2010), 1388–1429.

[15] Mnih, A., and Hinton, G. A scalable hierarchical distributed language model. In *In NIPS* (2008).

[16] Pang, B., and Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)* (Ann Arbor, Michigan, June 2005), Association for Computational Linguistics, pp. 115–124.

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[18] Peng, F., and Schuurmans, D. Combining naive bayes and n-gram language models for text classification. In *In 25th European Conference on Information Retrieval Research (ECIR* (2003), Springer-Verlag, pp. 335–350.

[19] Rish, I. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (2001), vol. 3, IBM New York, pp. 41–46.

[20] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323* (Oct. 1986), 533–.

[21] Schölkopf, B., and Smola, A. J. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002.

[22] Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., and Manning, C. D. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems* (USA, 2011a), NIPS'11, Curran Associates Inc., pp. 801–809.

[23] Socher, R., Lin, C. C.-Y., Ng, A. Y., and Manning, C. D. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (USA, 2011b), ICML'11, Omnipress, pp. 129–136.

[24] Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2011c), EMNLP '11, Association for Computational Linguistics, pp. 151–161.

[25] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (Seattle, Washington, USA, Oct. 2013b), Association for Computational Linguistics, pp. 1631–1642.

[26] Turian, J., Ratinov, L., and Bengio, Y. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (Stroudsburg, PA, USA, 2010), ACL '10, Association for Computational Linguistics, pp. 384–394.

[27] Turney, P. D., and Pantel, P. From frequency to meaning: Vector space models of semantics. *CoRR abs/1003.1141* (2010).

[28] Yessenalina, A., and Cardie, C. Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (Edinburgh, Scotland, UK., July 2011), Association for Computational Linguistics, pp. 172–182.

[29] Yuan, Q., Cong, G., and Thalmann, N. M. Enhancing naive bayes with various smoothing methods for short text classification. In *Proceedings of the 21st International Conference on World Wide Web* (New York, NY, USA, 2012), WWW '12 Companion, ACM, pp. 645–646.

[30] Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., and Manandhar, S. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)* (Beijing, China, Aug. 2010), Coling 2010 Organizing Committee, pp. 1263–1271.

[31] Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (Seattle, Washington, USA, Oct. 2013), Association for Computational Linguistics, pp. 1393–1398.