COMP-551-002 Final Project
Effective Use of Word order for Text Categorization

Improving the Baselines Study

Naomie Abecassis
260873070
naomie.abecassis@mcgill.ca

Antonios Valkanas
260672034
antonios.valkanas@mail.mcgill.ca

David Valensi
260873585
david.valensi@mail.mcgill.ca

I. Introduction

Natural language processing (NLP) is a set of techniques aimed at understanding and representing human languages automatically. While understanding human languages is a challenging task various deep learning methods have yielded results with astonishing success in the recent decades. Two classic problems in NLP consist of text sentiment classification and topic categorization. Convolutional neural network (CNN) architectures have long been able to surpass other non deep learning methods however improvements are still being made at a fast pace.

It is a well known fact that the bag of words (BoW) approach is a simple way of vectorizing and mapping the feature space to a vocabulary but it comes at the cost of losing information pertaining to the relative location of words within the sentence [1]. A simple remedy to this is the use of bigram and trigram words as shown in [2] and [3], however this approach is not always effective [4].

In 2015, to mitigate this loss of information, the approach of using the relative location of words in a text as opposed to merely noting the frequency of said words was incorporated in CNN architectures as an alternative to the commonly used bag of words approach [1]. While it might be intuitively true that word order matters when determining the meaning of a sentence there had not been conclusive and quantifiable prior research on the effect of word order in the context of sentiment and topic classification with CNNs.

It is important to note that word order was partially incorporated using a lookup table with good results [5]. The key difference is that the approach of the novel idea proposed here is to make a fully self reliant architecture without using outside help such as word2vec.

In order to better understand the robustness and strength of the model proposed in [Johnson zhang] we will perform a baseline optimization study and show that this proposed approach beats the baselines even after extreme fine tuning. The goal here is twofold; to beat the reported paper baselines and to approach the fully developed CNN model as much as possible.

II. Background

In order to fully understand the nature of our baseline study, background knowledge on both traditional approaches such as the BoW approach coupled with simple baselines such as naive Bayes (NB), support vector machines (SVM), and logistic regression, as well as basic CNN architectures must be studied.

A. Bag of Words & n-gram vectors

The bag-of-words model is one of the most popular representation methods for object categorization. The key idea is to quantize each extracted word from the text into one of the vocabulary words. The dictionary containing all of the vocabulary words can be created by mapping all words in the training examples to entries in the dictionary. This associates each word a specific dictionary index.

We can then transform every text segment to a histogram of word frequencies which is represented by a vector containing the frequency of dictionary vocabulary words.

The basic BoW approach entails using only single word entries in the vocabulary as shown in Fig. 1.

$$\mathbf{r}_0(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \text{don't} \\ \text{hate} \\ \mathbf{I} \\ \text{it} \\ \text{love} \end{matrix} \qquad \mathbf{r}_1(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \text{love} \end{matrix}$$

Fig. 1. Two examples of BoW vectorization of a sentence. [1]

However, in general this is not the case as n-word entries can be used as separate entries in the dictionary called n-grams.

B. Baseline Classifiers

In the studied paper the baselines used to test against the convolutional networks include two methods: SVM and NB. Furthermore, logistic regression was used by our team as a third method to test the performance of the more complex Reuters dataset.

1) Linear Support Vector Machine: An SVM is a supervised learning algorithm used for both classification and regression. It attempts to find the maximum margin hyperplane that divides the set input points of different classes such that the distance between the hyperplane and the nearest point of opposing classes is maximized.

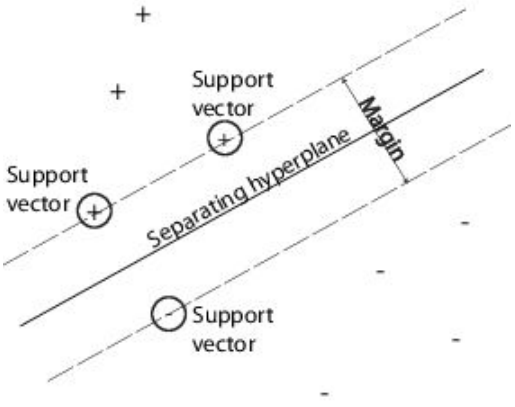

Fig. 2. Linear SVM support vector, margin and decision boundary.

The hyperparameter C is used to optimize the fit of the line to data and penalize the amount of samples inside the margin. With a lower C, samples inside the margin are penalized less whereas with a large C, samples are penalized more. SVMs can have different error functions with the most common being hinge loss and squared hinge loss.

2) *Naive Bayes*: Bayesian classifiers assign the most likely class to a given example described by its feature vector. Learning such classifiers can be greatly simplified by assuming that features are independent given class, that is, where is a feature vector and is a class. Despite this unrealistic assumption, the resulting classifier known as naive Bayes is remarkably successful in practice, often competing with much more sophisticated techniques. The method works by making use of Bayes theorem:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

where the naive assumption comes from the class conditional independence:

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$$

3) *Logistic Regression*: The underlying mathematical concept behind logistic regression is the natural logarithm of the odds ratio known as logit. This can be seen in the error function that logistic regression fitting minimizes as shown below. In this baseline study multiple logistic regression methods are used including liblinear, the (LBFGS) Broyden–Fletcher–Goldfarb–Shanno algorithm, Newton conjugate gradient and SAG. All these methods allow the use of L2 regularization to reduce overfitting bia increasing the model bias and reducing the variance.

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$$

C. Convolutional Neural Networks

CNNs are known for their unrivaled performance on image classification [6]. One of the reasons for their success is that CNNs are able to exploit the underlying image structure via local connectivity [7]. While also taking advantage of parameter sharing and local

receptive fields, CNNs can learn robust representations from image data.

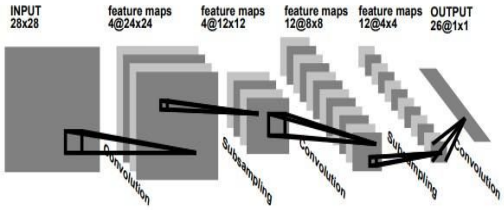

Fig.3. Example of CNN Architecture used for digit recognition, with rigid structure[6].

In the context of NLP, for small fixed n-gram values, CNNs in conjunction with BoW lose the ability to exploit the local ordering of words for each segment of text due to the transformation of text from a string to a frequency histogram vector [1]. However, a great advantage of CNNs is that they are able to learn embeddings of text regions useful for the intended task [1]. As a result, for higher-order n-grams the BoW convolution layer learns the appropriate n-grams that give the best predictive ability. As shown in the paper we are studying, the network assigns more weight to short highly descriptive phrases such as “I love it” rather than longer ones despite having other higher order n-gram options and even though it had not seen the particular phrase in training.

While the CNN with the simplest network architecture, such as the one in Fig. 3, might be the easiest implement, there are benefits to intelligently increasing the complexity. One such example is the use of parallel CNN architectures. The main idea of parallel CNNs is to introduce more convolutional layers and place them in parallel. As Fig. 4 shows, these independent convolutional layers complement each other and improve accuracy.

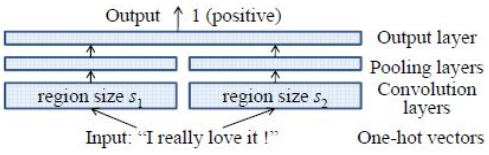

Fig.4. Example of parallel CNN architecture. [1]

III. Improving the Baselines

A. Methodology

To test the assertions of the authors, that using word order information can improve sentiment and topic classification 3 datasets were used. The study is conducted on the following datasets: IMDB: movie reviews; Elec: electronics product reviews from the amazon dataset; RCV1: topic categorization from the Reuters Dataset including both the single label as well as the multilabel version. The datasets are summarized in Table 1:

Dataset	Train Data	Test Data
IMDB	25,000	25,000
Amazon Elec.	25,000	25,000
RCV1 single	15,564	49,838
RCV1 multi	23,149	781,265

Table 1: Summary of datasets used.

It is important to note that for the IMDB and Amazon datasets we followed the same procedure as the authors of [1] to make the sure that both the training set and the test set contain an equal amount of positive and negative reviews. The RCV1 dataset was used as is from the authors repository provided in footnote 12 on page 6 of [1].

Our methodology was to first reproduce the performances reported in the paper, then to extensively fine-tune the hyperparameters of the models. Finally, we were meant to explore different simple algorithms combined with the previous models to outperform the results.

To do the hyperparameter fine tuning we designed the some simple experiments described in a later section. Additionally, we designed a simple experiment to find if any other simple baseline method outside the reported ones could improve the baseline performance.

B. Result Reproduction

Initially, we aimed to reproduce the reported baseline accuracies for the three datasets. To achieve this we first followed the preprocessing

steps outlined by the authors. Specifically, we tokenized the text and emoticons as well as converting all characters to lower case. For the Reuters dataset we did not have to do much preprocessing as the author made the preprocessed dataset available via their website so all we had to do was to match the indices of the preprocessed files the authors provided to the general RCV1 multilabel dataset to obtain the data.

Following the paper's directions we implemented an SVM classifier as well as a Naive Bayes and a basic voting classifier combining the two. The result was that we were able to approach but not quite reach the reported paper results. This clearly proved that the authors had performed some hyperparameter fine tuning.

C. Improving the Baselines

methods	IMDB	Elec	RCV1
SVM bow3 (30K)	10.14	9.16	10.68
SVM bow1 (all)	11.36	11.71	10.76
SVM bow2 (all)	9.74	9.05	10.59
SVM bow3 (all)	9.42	8.71	10.69
NN bow3 (all)	9.17	8.48	10.67
NB-LM bow3 (all)	8.13	8.11	13.97
bow-CNN	8.66	8.39	**9.33**
seq-CNN	8.39	7.64	9.96
seq2-CNN	8.04	7.48	–
seq2-bow$n$-CNN	**7.67**	**7.14**	–

Table 2: Error rate (%) comparison with bag-of-n-gram-based methods. Sentiment classification on IMDB and Elec (25K training documents) and 55-way topic categorization on RCV1 (16K training documents). '(30K)' indicates that the 30K most frequent n-grams were used, and '(all)' indicates that all the n-grams (up to 5M) were used. CNN used the 30K most frequent words.

To improve the baseline performances achieved by the paper as shown in Table 2, we explored three main methods:

1. Analyze and improve the preprocessing
2. Fine-tune the hyperparameters
3. Use ensemble methods to vote on results with several fine-tuned models like: bagging, boosting.

As observed in the previous section, the authors had already done some baseline optimization via hyperparameter tuning since our results using a NB and an SVM with default hyperparameters were inferior. They tuned hyperparameters for each one of the models [1] (sec. 3.3) and have done a preprocessing [1] (sec. 3.4) which is, in one hand very simple, but in the other hand, which might not be the best one, for the given datasets.

Our first hypothesis was that the preprocessing was suboptimal. To prove this we designed the following experiment: Keeping everything else the same compare the performance of the SVM and NB classifiers with input features that pass from different preprocessing pipelines to find the best preprocessing method for these datasets. Our various preprocessing pipelines consisted of all the previous steps such as text tokenization, emoticon replacement by tokens and lower-case conversion of all characters in addition to the following:

- removing stopwords
- stemming or lemmatization
- weighting BoW using TF-IDF

To determine which preprocessing pipeline was the best we run the basic SVM each time with a different set of preprocessing that contained some or all of the methods discussed and evaluated the results from the accuracy metric.

Our second hypothesis was that the baseline hyperparameter tuning could be improved. The reason for this is that the probability that the authors chose the perfect parameters such that changing them in any direction will decrease performance is very small. Therefore, the experiment to find the optimal parameters is an extensive grid search. Specifically, the SVM loss function and its misclassification parameter C can be varied until a global minimum is found. Similarly, the grid search approach can help us find the correct additive (Laplace/Lidstone) smoothing parameter (alpha) for Naive Bayes.

Finally, our third main hypothesis that we tested was to verify if a voting classifier comprised of 2 or more classifiers could further improve the performance of our methods. To verify this experimentally we first got the results for the tuned independent baseline methods and combined used a weighted voting system. We also tested a third classifier based on logistic regression following the same approach that we did for Naive Bayes and SVM.

IV. Results

A. Code Replication Results:

Using the provided baseline models, we reproduced the experiments. Our performances are shown in Table 3 below.

methods	IMDB	Elec	RCV1
SVM bow1	88.3%	88.1%	89.3%
SVM bow2	89.4%	90.15%	89.6%
SVM bow3	90.1%	90.16%	90.0%
NB bow2	83.05%	87/87.2%	88.3%
NB bow3	84.15%	87.9/88.3%	88.7%

Table 3: Summary of attempted replication accuracy results.When NB has two accuracies: for Bernoulli NB and Multinomial NB

B. Baseline Hyperparameter Tuning

Since extensive tuning experiments were already conducted by the authors [1], we attempt similar experiments to the authors, as described in the methodology section, after incorporating an improved data preprocessing. As it turns out, our first hypothesis was correct. In fact, that the TF-IDF weighting leads to an improved representation of the data (as shown in Table 5), while removing the stopwords reduces our accuracy which made us remove it. We found that lemmatization or stemming reduced the dimensionality but did not particularly affect the quality of the model prediction.

Using this improved preprocessing we were able to move on to an extensive hyperparameter tuning. As suggested in the methodology we tuned the SVM's C and loss parameters as well as the NB alpha. Table 4 shows the optimized results.

methods	IMDB	Elec	RCV1
SVM bow1	90.0%	88.6%	90.2%
SVM bow2	90.2%	91.2%	90.4%
SVM bow3	90.7%	91.5%	91.1%
NB bow2	84.8%	86.8/88.5%	88.6%
NB bow3	86.9%	87.8/89.5%	89.4%

Table 4: Summary of fine tuned baselines accuracy results. The NB has two accuracies: for Bernoulli NB and Multinomial NB

Secondly, we tested a strong baseline based on NB+SVM similar to the one reported in [WM12]. This is in fact a voting classifier, with a Naive Bayes, Linear SVM and also a logistic regression (it appeared that adding it improved the performances)

Methods	Elec dataset
nb+svm bow1	88.7%
nb+svm bow2	91.3%
nb+svm bow3	91.4%

Table 5: Summary of ensembling strategy on the Amazon Elec Reviews

The ensembling strategy doesn't always help in as we expected it to but offers a simple way to incorporate many classifiers which are good at predicting different sample classes.

Finally, we also tested a bagging strategy (Bootstrap Aggregation) on the Linear SVM model with *n-grams.* Although this reduces variance, the performances are not better using this method.

C. Attempt to beat deep learning model.

As shown in Table 5 our best performance occurred when using a consensus-based method

that combined simple classifiers. This was further improved upon when using an additional classifier based on logistic regression. In fact, many different logistic regression were tested as and as described in the background all of them employed L2 regularization to reduce overfitting. A particularly interesting result was the fact that logistic regression was able achieve very high accuracy in the RCV1 dataset. This can be seen in Fig. 5 and Fig. 6.

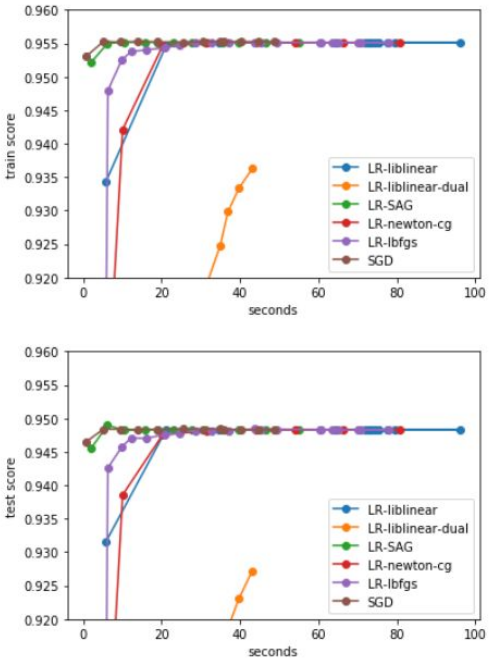

Fig.5. Logistic Regression fit for various log. reg. types for RCV1..

Referring back to Fig. 2 we see that liblinear regression surpasses all non deep learning methods and in fact approaches the sequential CNN error rate. In fact, given enough examples the method is able to surpass the CNN result reported in the paper. Of course this only occurs when we use a lot more training data (in the order of 10 times more) to train our model compared to the authors which is obviously not a fair comparison, as for an equal amount of training examples the CNN outperforms our model. Nevertheless, this goes to show that logistic regression is an excellent non-deep learning method of solving the problem without need for GPUs or long training time.

It is well known that removing the stop words might not be effective in certain tasks since it also removes some crucial information in some BoWs. In a sense, removing stop words is partially throwing away the word order in some cases and obviously it's not what we want. Moreover, lemmatization or stemming may be useful for creating common patterns between resembling expressions (the tenses are no longer differentiated, and inflected forms of the same word are considered the same) but it may also lead to some mistakes since the stemming, for example, is so crude that it can confuse between two expressions which tell the contrary. Finally, we note that TF-IDF weighting is helpful to filter stop words or too frequent words in the BoW that cause noise while not deleting them and thus, preserving the word order [2].

V. Discussion

A. Preprocess Improvement

In our results, we have shown that as a part of the preprocessing, removing stopword doesn't improve the performances of the baselines. Indeed, for some tasks, removing stopwords can be counterproductive because it removes important data. For example, the 112th word of the NLTK's list of english stopwords is "not" thus when we remove these kind of words the meaning of a sentence completely changes. As an example from the dataset, we can cite the sentence "This movie was not good": if we remove the word "not", the meaning of the sentence changes from negative to positive. Thus, removing stopwords in this case won't help to get better performance. Maybe, a way to improve that would be to remove only certain stopwords, such as "a" or "the", only neutral stopwords. Moreover, in this project we are trying to evaluate the influence of keeping the order of the words on the accuracy, and by removing stopwords, we sometimes remove the link between adjacent words in the text.

On the other hand, we found that using TF-IDF improved our results. The reason that we expect this to happen is that using the logarithm of the frequencies still allows more noise from high frequency words to enter our vector

representation compared to TF-IDF.

B. N-Gram observations

In our task of improving the performances of the baselines we tried several models of N-grams, for different values of N. As we can observe in the results, the performance of the algorithm tends to increase when we increase N, but only until a certain N. Indeed, for N<4 increasing the value of N results in better performances but when N>4 the performance cease to increase. At one point, using a N-grams model with big N becomes counterproductive. The use of N-grams for keeping the order of words in a text is a simple solution but not really efficient. Indeed, the complexity of the algorithm grows exponentially with N and generally, each N-gram ignores the fact that other N-grams share constituent words. Using the CNN method to solve the issue of the order of words is an efficient way to get around the issues caused by the N-grams method. Indeed, even if N-gram provides a information on the order of the words, this information is limited and sometime, a N-gram will take words from other sentences that are not related to a current sentence.

C. Baseline Improvement

Using parameter grid we performed extensive hyperparameter tuning for our classifiers. This resulted in improving the performance of the models. Another major source of improvement to the baselines was trying to use other baseline methods such as logistic regression and making ensemble classifiers. The results showed that logistic regression is an excellent choice for a basic classifier as it approached the deep learning method closer than any other classifier proposed in [1].

D. Additional Comments

It is important to note that we did not have access to a GPU for this project so we could not complete the section where the authors benchmark their CNN performance over time (see [1] - Figure 5 from Johnson Zhang) as they used very strong hardware.

VI. CONCLUSION

Although the authors of the paper provided a lot of the code required to reproduce their results, this was very hard to without access to computationally expensive resources due to the large size of the datasets, especially RCV1. As such, our efforts in this final project were concentrated on reproducing the baseline performances cited by the authors of the paper. We found that the authors had done a relatively good job of tuning their hyperparameters as small gains were achieved by just tuning our models. On the other hand, looking further in depth we noticed that the preprocessing of the authors was not optimal and improved upon it to further boost our results. Finally, we introduced new baseline models which were better at approximating the true predictions than the original baseline models. In conclusion we achieved our goal to improve the performance of the paper's baselines.

VII. STATEMENT OF CONTRIBUTIONS

For this project, we all brainstormed together about the ways to solve the problem. We then separated the tasks such that Antonios and David wrote the code and Naomie the spotlight. After that we all wrote together the report.

REFERENCES

[1] Rie Johnson, Tong Zhang, "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks"

[2] http://www.tfidf.com/

[3] Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification.

[4] NLP Stanford https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915, 2016.

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[7] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 1995.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[9] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation.

In CVPR, 2014.
[11] D. Lowe, “Object recognition from local scale-invariant features,” Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999.
[12] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In ICML, 2010. [11] C. Lin Deformed Convolutional Networks in PyTorch goo.gl/fBZv7a
[12] S. Hu, D. Yu, J. Dias, COMP551-Project4: Find the Largest Digit, COMP551, McGill University, 2018.
[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inceptionv4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261, 2016.
[14] P. Getreuer, “Linear Methods for Image Interpolation,” Image Processing On Line, vol. 1, 2011.

TODO: FIX REFERENCES IN TEXT!