2018 5th International Conference on Behavioral, Economic, and Socio-Cultural Computing (BESC)

Latent Semantic Analysis Boosted Convolutional Neural Networks for Document Classification

Eren Gultepe, Mehran Kamkarhaghighi, Masoud Makrehchi
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology, Oshawa, ON, Canada
{eren.gultepe, mehran.kamkarhaghighi, masoud.makrehchi}@uoit.ca

***Abstract*—Convolutional neural networks (CNNs) have been shown to be effective in document classification tasks. CNNs can be setup using various architectures with many different parameter settings, which may make them difficult to implement. For many document classification tasks, data transformed with ngrams (typically using uni, bi, and trigrams) and term-frequency inverse-document-frequency (TFIDF) weighting are still considered effective baseline models when used with linear classifiers such as logistic regression, especially in smaller datasets with less than 500K observations. A parsimonious CNN baseline model for sentiment classification should replicate the easy use of linear methods. In this study, we introduce a Latent Semantic Analysis (LSA) based CNN model, in which natively trained LSA word vectors are used as input into parallel 1-dimensional convolutional layers (1D-CNNs). The LSA word vector model is obtained by applying singular value decomposition (SVD) on the data transformed by a unigram and TFIDF weighting. Thus, the convolutional layers are designed with window sizes that are best suited for LSA word vectors. This parsimonious LSA-based CNN model exceeds the accuracy of all linear classifiers utilizing ngrams with TFIDF on all analyzed datasets, with average improvement of 0.73% by the top performing LSA-based CNN models. This may be due to the fact that CNNs are better adept at capturing word relationships in phrases and sentences that are not necessarily in the training corpus. Furthermore, the LSA-based CNN model exceeds the performance of word2vec-based CNN models as well. Thus, the success of LSA-based CNNs may potentiate their use as a baseline in classification tasks alongside linear models. Also, we provide guiding principles to simplify the application of LSA-based CNNs in document classification tasks.**

I. Introduction

Automated text classification is a classic problem where the objective is to assign a set of categories to documents. The studies in text classification vary from developing a sophisticated document feature representation methodology [1, 2] to implementing the best possible classifiers that use simple document representations [3]. A common approach in text classification is the bag-of-word as ngrams representation, where documents are represented with a vector of words that appear in each document [4]. Although, ngrams are very simple to generate, however, the main challenge in such a presentation is that the resulting vector is very large and sparse. The sparsity and semantically understanding text documents are the major challenges in text categorization [5, 6]. In fact, data sparsity is encountered in ngrams due to the length of generated text and the use of different words with the same meaning.

Recent studies [7] have used 1-dimensional convolutional layers (1D-CNNs) directly with one-hot vectors representation of bag-of-words data. Typically this is not well-suited for convolution networks, wherein dense data is preferred. To mitigate this issue, an embedding layer to densify the one-hot vectors is preferred prior to the convolutional layers [8, 9, 10, 11]. An embedding layer takes a large vocabulary and projects the full representation into smaller dimensional space[8]. Furthermore, in sentiment analysis [12, 13], CNNs with an embedding layer are similar to long short-term memory (LSTM) models [14] that have an embedding layer. However, LSTMs are known to be difficult to train since they are sensitive to hyperparameters such as batch size, hidden dimensions, and etc. [15].

Recently, pretrained word vectors obtained from an unsupervised neural network language model (word2vec[16]) have been successfully used as trainable weights into the embedding layer, prior to the convolutional layers[8]. The word2vec (w2v) word vectors were trained on 100 billion words of Google News. The goal in [8] was similar to [17], which showed that for image classification, pretrained features from a different domain can be fine-tuned to domain specific-tasks. However, if the domain of the document classification task is very different from the pretrained vectors, classification accuracy may not reach its full potential, since the purpose of unsupervised pretraining is to provide relevant features that can improve accuracy [18]. There are limitations for training word2vec models from natively. The first is that it requires a large datasets for training. For satisfactory performance, a minimum of 10 million words should be in the training corpora [19]. For smaller datasets, LSA word vectors were found to provide better performance in semantic similarity tasks [19]. The second is that there are a number of hyperparamters that need to be chosen before successful model training can be implemented [20].

Thus, the goal of this study is to develop a straightforward CNN-based baseline for text classification using locally trained word vectors. We use locally trained word vectors from an LSA model, which is easily obtained by SVD on a unigram transformed and term-frequency inverse-document-frequency (TFIDF) weighted data. Our model has the advantage of learning the words embeddings natively, without the onerous hyperpameter search and extensive training time. Thus, in the following sections we highlight how to efficiently setup the LSA-based convolution networks and demonstrate its effec-

978-1-7281-0207-8/18/$31.00 ©2018 IEEE
DOI 10.1109/BESC.2018.00027

Review Dataset	Classes	Train	Test
IMDB Movie	2	25,000	25,000
AG's News	4	120,000	7,600
DBPedia	14	560,000	70,000

TABLE I: Summary of the text document datasets.

tiveness against baseline linear classifiers and pretrained w2v-based CNN models.

II. Materials and Methods

In the following, we describe the implementation of the baseline linear classifier, w2v-based CNN, and LSA-based CNN models. Document datasets for all models used the same preprocessing steps.

A. Datasets

The models were evaluated on three benchmark datasets (IMDB, AGNews, DBPedia). These three datasets were chosen because they were shown have to high classification accuracies using linear classifiers on ngram transformed and TFIDF weighted data [21, 22]. The last two datasets are from [21]. The statistics of the datasets and the size of training and test data are presented in Table I. All datasets contain balanced distribution of classes. The information regarding each of the datasets are described as follows:

IMDB Movie reviews: This dataset contains 25,000 training and 25,000 test samples, wherein the objective is to predict if a given review has either negative or positive sentiment [23].

AGNews: Antonio Gulli's news (AGnews) article corpus contains 496,835 articles from more than 2000 different sources [24]. Only the four largest classes from this corpus, where each document is represented by the title and description fields are used. For each class, 30,000 training and 1900 testing samples are used.

DBPedia: DBpedia is ontology dataset extracted from Wikipedia [25]. It contains 14 non-overlapping classes. Each class contains 40,000 training and 5,000 testing samples. Each document is represented by the title and abstract of each Wikipedia article.

For each review dataset we remove punctuation, hypertext, stopwords, and convert to lowercase.

B. Linear Classifier with ngram and TFIDF

To construct the TFIDF weighted {1,2,3}-ngram text representation, first a bag-of-words (BOW) transformation is performed. The BOW representation is constructed by using the frequency count of each word (unigram), bigram, and trigram. Then for the TFIDF weighting[26], the frequency counts (term-frequency) is divided by the inverse document frequency (IDF). The IDF is the log of the total number of samples divided by the number of samples in the data. The {1,2,3}-ngram size is limited to the top 30K most frequent terms, similar to [7]. TFIDF representation, was shown to generally perform better than the BOW representation [21]. Finally, the TFIDF weighted {1,2,3}-ngram representation was used with multinomial logistic regression with the L_2 regularization parameter $C = 1$ from the LIBLINEAR package [3].

C. CNN Model

CNNs are feed-forward neural networks, where the features generated by the neural networks layers are convolved with each other until a final classification is applied. Typically, the features are extracted from small 2D patches of an image. For instance, for a 32×32 grayscale image, 5×5 regions are extracted from which the features are learned. However, for text documents this 2D region is simplified down to 1D patches.

The core of the LSA-based CNN and w2v-based CNN [8] is a 1D-CNN model [10]. A sentence can be defined by a sequence of n concatenated k-dimensional word vectors, $\mathbf{x}_i \in \mathbb{R}^k$,

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus ... \oplus \mathbf{x}_n.$$

The convolution of words, selected by a window of length h, with a filter $\mathbf{w} \in \mathbb{R}^{h \times k}$ produces the feature mapping c_i given by the formula below,

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b),$$

where $b \in \mathbb{R}$ is the bias term and f is a nonlinear function such as the sigmoid function. Applying the filter c_i to all sentences selected by the window h provides the feature mapping $\mathbf{c} \in \mathbb{R}^{n-h+1}$, where

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}].$$

Then max-over-time pooling is applied to all the features, where the maximum activation value for the filter is obtained. This convolutional process is applied to all filters for a given window of words. The last step in the CNN model is a softmax layer, which is used to obtain the output class of the text document. A schematic workflow of a general word vector based CNN is shown in Figure 1. In the subsections below, the details regarding the number of filters, window sizes, and word vectors is provided.

D. Embedding Layer

The word vectors in the CNN model are represented by the embedding layer $\mathbf{W}_E \in \mathbb{R}^{d \times v}$, where d is the dimension of the embedding layer and v is the size of the vocabulary. The input text layer is a set of one-hot (binary) vectors representing the words used in documents from the vocabulary v (as in Figure 1). The goal of the embedding layer is to map a discrete and sparse representation of words to continuous and dense values, which are best suited for CNNs. Therefore in the CNN model, the ngrams are modeled by a continuous feature space representation where similar words are located close to each other. This transformation provides the semantic understanding of words to 1D-CNN models. In the training of the CNN model, word vectors are fine-tuned to fit the classification task of the documents.

In this study, three types of word vectors are used to initialize the embedding matrix. First, as a baseline, random word vectors from $\mathbf{W}_{rand} \in \mathbb{R}^{d \times v}$ are used, which are initialized by the uniform distribution within the range [-0.05, 0.05]. Second, the LSA word vectors are formed by using

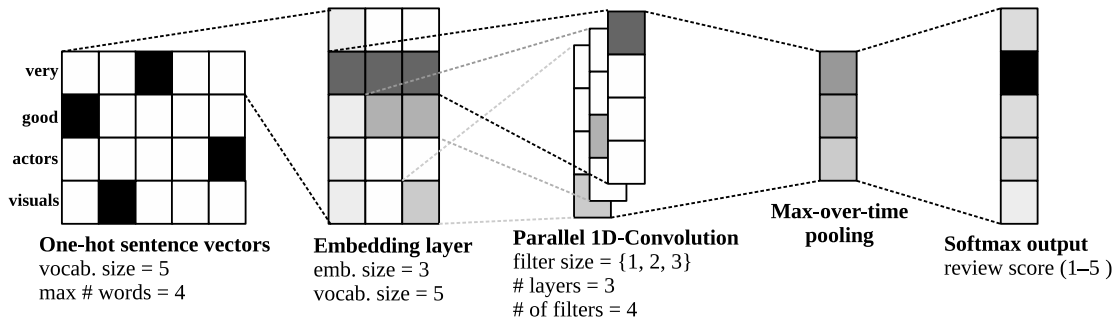

Fig. 1: Schematic of the proposed model. In this example, a single sentence of a preprocessed movie review is input into a model where there is a vocabulary size of 5 in the dictionary. The word vector embedding layer reduces the dimensionality of the dictionary to 3. Then 3 parallel 1D-convolutions are applied, using 3 different filter sizes. on which L_2 reguralization is performed. Max-over-time pooling is applied, which takes the single best feature per feature map. Finally, the review is classified with a score from 1 to 5.

SVD on a unigram transformed and TFIDF weighted data. Specifically, using only the top d singular values, SVD is performed on $\mathbf{X} \in \mathbb{R}^{v \times n}$, where v is the size of the vocabulary and n is the number of documents, giving

$$\mathbf{X}_d = \mathbf{U}_d \mathbf{S}_d \mathbf{V}_d^T.$$

Then the LSA word vectors are defined as the rows of

$$\mathbf{W}_{LSA} = \mathbf{U}_d \mathbf{S}_d.$$

Also, according to Levy et al.[20], the word vectors are row normalized, which has been shown to improve representative accuracy.

Finally, the word2vec word vectors pretrained on 100 billion words of Google News are used to form $\mathbf{W}_{w2v} \in \mathbb{R}^{d \times v}$. The word2vec model is a two-layer unsupervised neural network that produces vector representation of words, similar to LSA. In the model, the objective function maximizes the log probability of a context word (w_O), given its input words (w_I). The output is a n-dimensional vector for each word in the vocabulary that represents the weights of the hidden layers. Words that are similar in context to each other in the training corpus are located close to each other in the vector space representation. In Figure 2, two different w2v model architectures are shown, the Continuous Bag of Words (CBOW) and Skip-gram models. The CBOW architecture, predicts a word based on the surrounding context words. The Skip-gram architecture uses the current word to predict the surrounding words in a fixed-size window. For infrequent words, the Skip-gram architecture works better, whereas the CBOW model works faster than the Skip-gram model.

The $\mathbf{W}_{rand}$ initializing weights are used to test the efficacy of the vectors $\mathbf{W}_{LSA}$ and $\mathbf{W}_{w2v}$ initializing weights. For the subsequent CNN models presented below, the following name convention is used, "{ngram range}-$\mathbf{W}_E$ type-regularization-CNN", where "{ngram range}" indicates the window size used on the word vectors, "$\mathbf{W}_E$" takes on either $\mathbf{W}_{rand}$, $\mathbf{W}_{LSA}$ or $\mathbf{W}_{w2v}$, and "regularization" is either L_2 weight decay (wt) or dropout (dp) based. The "{ngram range}" and "regularization" are dependent on the model architectures defined below.

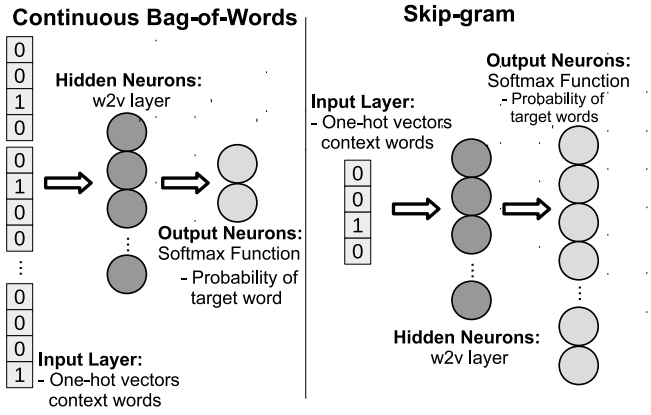

Fig. 2: Training of word2vec word vectors. The two models of word2vec architectures are shown here.

E. LSA-based CNN Model

To successfully apply LSA in an CNN model, two different architectures were developed to take advantage of the LSA word vectors. Specifically, networks with ngram filter sizes = {1,2,3} and {1,2,3,4} are implemented. Each ngram filter size pertains to a separate and parallel convolutional layer in the model. These layers are finally concatenated with each other in the max-over-time pooling layer, prior to the softmax classification layer. The following are the parameters used for both network architectures:

- **Filters:** The number of filters is set to 128 for each of the ngram filters, which is a common size in convolutional network models [27].
- **Emedding:** The embedding dimension size was set to 300 to match the word2vector dimensionality [16].
- **Regularization:** The regularization was applied to each of the weights in convolutional layers using an L_2 parameter weight decay of of $10e^{-5}$ [28, 7].
- **Activation:** Rectified linear units (ReLu) were used as an activation function, as it has been shown to be sufficient in other studies [29] and also to enable comparison to the w2v-based CNN model specified in Kim [8].

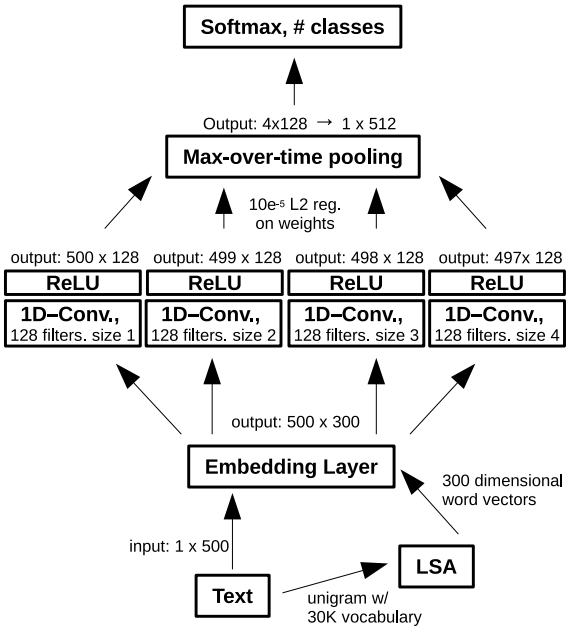

Fig. 3: Sample CNN model architecture. The above setup shows the architecture and parameters for the top performing LSA-based CNN model.

A summary of the architecture is shown for the {1,2,3,4}-LSA-wt-CNN model in Figure 3. The {1,2,3}-LSA-wt-CNN model follows the same setup except for the absence of the quadgram filters.

F. W2V-based CNN Model

The 1D-CNN architecture from Kim [8] was also used to compare against the LSA-based CNN architecture. The w2v-based CNN architecture uses larger filter sizes since the word2vectors are designed to capture long distance relationships among words [16]. The w2v-based CNN model was shown to be an effective classifier compared to linear and other state-of-the-art classifiers such as RNN's and LSTM's [8]. For the w2v-based CNN architecture, the following parameters as given in [8] were used: filter sizes = {3,4,5}; embedding dimension size = 300; and dropout [30] used for regularization. Only the number of filters for each filter size was increased from 100 to 128, to enable fair comparison to the CNN model architecture developed for LSA word vectors.

G. Experimental Setup

For all three datasets, the vocabulary size of the training corpus was limited to 30K words for the linear and CNN based models. To test the effectiveness of the LSA and w2v specific CNN architectures, the $\mathbf{W}_{rand}$, $\mathbf{W}_{LSA}$ or $\mathbf{W}_{w2v}$ word vectors were input into both types of architectures. The Adadelta optimizer [31] was used to trained the models for 10 epochs as it was shown to reach convergence quickly [8]. For all datasets, we report the accuracy of document classification on the testing samples.

Model	IMDB	AGNews	DBP
{1,2,3}-TFIDF-Logistic Reg.	0.8942	0.9142	0.9797
{1,2,3}-$\mathbf{W}_{rand}$-wt-CNN	0.8957	0.9201	0.9847
{1,2,3}-$\mathbf{W}_{w2v}$-wt-CNN	0.8996	0.9201	0.9854
{1,2,3}-$\mathbf{W}_{LSA}$-wt-CNN	0.9001	0.9209	0.9858
{1,2,3,4}-$\mathbf{W}_{rand}$-wt-CNN	0.8961	0.9195	0.9849
{1,2,3,4}-$\mathbf{W}_{w2v}$-wt-CNN	0.8989	0.9201	0.9859
{1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN	**0.9010**	**0.9213**	**0.9862**
{3,4,5}-$\mathbf{W}_{rand}$-dp-CNN	0.8886	0.9159	0.9842
{3,4,5}-$\mathbf{W}_{w2v}$-dp-CNN	0.8965	0.9196	0.9859
{3,4,5}-$\mathbf{W}_{LSA}$-dp-CNN	0.8980	0.9195	0.9855

TABLE II: Classification accuracy for text documents. Bold face indicates best accuracy for a dataset. The following name convention is used, "{ngram range}-$\mathbf{W}_E$ type-regularization-CNN".

III. Results

Table II presents the experimental results of the proposed CNN architecture for LSA word vectors against the baseline linear model and the CNN architecture designed for w2v word vectors. The best performing CNN architecture and word vector type across all datasets was the {1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN model. It had a higher accuracy than all other models. For the IMDB and AGNews datasets, the $\mathbf{W}_{w2v}$ word vectors reached their maximum accuracy using the {1,2,3}-$\mathbf{W}_{w2v}$-wt-CNN models instead of the {3,4,5}-$\mathbf{W}_{w2v}$-dp-CNN model, which was specifically for $\mathbf{W}_{w2v}$. Moreover, on the AGNews dataset, $\mathbf{W}_{w2v}$ word vectors were tied for accuracy on the {1,2,3}-$\mathbf{W}_{w2v}$-wt-CNN and {1,2,3,4}-$\mathbf{W}_{w2v}$-wt-CNN models. On the DBPedia dataset, $\mathbf{W}_{w2v}$ word vectors reached its peak accuracy using the {3,4,5}-$\mathbf{W}_{w2v}$-dp-CNN model. Even then, it only tied the {1,2,3,4}-$\mathbf{W}_{w2v}$-wt-CNN model.

When the {3,4,5}-$\mathbf{W}_E$ type-dp-CNN architecture was used, $\mathbf{W}_{w2v}$ performed better than $\mathbf{W}_{LSA}$ and $\mathbf{W}_{rand}$ on the AGNews and DBP datasets, but not on the IMDB dataset, demonstrating that this architecture is w2v specific. Nonetheless, this architecture never achieved the maximum performance. Also, on all of the CNN architectures, $\mathbf{W}_{rand}$ performed better than the linear ngram models. However, the $\mathbf{W}_{rand}$ word vectors still performed worse than $\mathbf{W}_{LSA}$ word vectors on all datasets, but tied the $\mathbf{W}_{w2v}$ word vectors on the AGNews dataset.

kimThe average accuracy across all datasets was higher for the {1,2,3,4}-$\mathbf{W}_E$ type-wt-CNN (mean = 0.9349) and {1,2,3}-$\mathbf{W}_E$ type-wt-CNN (mean = 0.9347) architectures than the {3, 4, 5}-$\mathbf{W}_E$ type-dp-CNN (mean = 0.9326) architecture. Although, the {1,2,3,4}-$\mathbf{W}_E$ type-wt-CNN and {1,2,3}-$\mathbf{W}_E$ type-wt-CNN models have a small difference in mean performance, the extra quadgram in the former model helped to provide the individual maximum classification accuracy with $\mathbf{W}_{LSA}$, which the trigram model never achieved.

We also compared the {1,2,3}-$\mathbf{W}_E$ type-wt-CNN model to state-of-the-art character-based CNN classifiers used in document classification in Table III. These CNN models use alphanumeric characters to embed the documents rather than words. The idea is to learn the semantic relationships of the words in higher convolutional layers in a similar way to image classification tasks used for CNNs, where the input data are only pixels [32, 21]. The models mainly differ in the number of convolutional layers and the filters sizes in CNN architecture.

Model	AGNews	DBP
{1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN	**0.9213**	0.9862
6-$\mathbf{W}_{char}$-CNN [21]	0.9145	0.9842
9-$\mathbf{W}_{char}$-CNN [32]	0.9083	0.9834
29-$\mathbf{W}_{char}$-CNN [32]	0.9133	**0.9871**

TABLE III: Comparison of classification accuracies of the LSA-based CNN to character-based CNN models. Accuracy values of the character-based CNNs are taken from their respective studies. Bold face indicates best accuracy for a dataset.

The model of the character-based CNN by Zhang et al. [21] uses 6 convolutional layers with filter size of 7 (on the first two convolutional layers) and 3 (on the remaining convolutional layers), which is designated as 6-$\mathbf{W}_{char}$-CNN. In Conneau et al. [32], either 9 convolutional layers or 29 convolutional layers with a filter size of 3 was used, which are designated as 9-$\mathbf{W}_{char}$-CNN and 29-$\mathbf{W}_{char}$-CNN, respectively. Compared to these models, our best performing {1,2,3,4}-$\mathbf{W}_{LSA}$-wt-CNN model had the highest accuracy on the AGNews and second highest accuracy on the DBPedia datasets. Comparison results for the IMDB dataset is not available since [21, 32] did not use it for analysis.

IV. Discussion

This study has shown that natively trained LSA word vectors can be used as an alternative to pretrained w2v word vectors in a CNN model for text document classification. Also we have introduced a novel architecture for CNN classification of text documents, using small word window sizes. The LSA-based CNN model has also been shown to perform the better than all other character-based CNN models on the AGNews dataset and second best on the DBPedia dataset.

Most CNN architecture for document classifications have used combinations of window sizes ranging from 3, 5 and 7 words[32]. Although, the goal was to model short and long distance relationships among words [32], smaller window sizes of 1 or 2 words have not been analyzed, as in this current study. As shown in Johnson and Zhang [7], effective linear classification with {1,2,3}-ngram is heavily dependent on unigrams rather than bigram and trigrams. The larger window sizes may better at modeling semantic similarity tasks, but the short distances may be better suited for classification.

Thus, the success of the LSA-based CNN model architecture could be attributed to the use of the smaller filter sizes together with the LSA word vectors. Since the LSA word vectors are extracted using unigrams, the LSA only encodes short distance relationships among words. Thus, using filter sizes larger than quadgrams would not be able to capture meaningful semantic relationships among words. Whereas the w2v word vectors perform better in a CNN with larger filter sizes, since w2v word vectors are trained with large windows sizes ranging from 5 and 10 words in length [20]. Also the results show that carefully tuning traditional methods, such as LSA word vectors, gives equivalent results to deep methods, such as w2v, as shown in Levy et al. [33].

A limitation of linear models is that they can not use ngram representations that are not present in the training dataset [7]. CNNs are able to find and use ngrams that are not wholly in the training set, which is why CNN based ngram models perform better than linear ngram models. For instance, CNN based ngrams can learn a general trigram "best *X-positive* ever", where *X-positive* represents a positive word. This general trigram can then be used to classify the testing data as long as it follows the general form [7].

One limitation of the LSA-based CNN model and the related small word window architecture, can be that in datasets with long documents (i.e., many and long sentences per document), the model may begin to perform worse against models that take advantage of larger window sizes. In this case, CNN architecture based on w2v word vectors or character-based models may perform better, since they are more likely to capture long distance semantic relationships among words.

V. Conclusion

In this study we have shown that LSA word vectors using CNNs with small window sizes can be used as a baseline classifier for document classification. Moreover, the LSA word vectors out performed pretrained w2v word vectors in all CNN models presented in this study. One reason for this is that the LSA word vectors are more domain specific than pretrained w2v word vectors. Furthermore, with the LSA-based CNN models, LSA word vector dimensionality can be easily adjusted to any size. Whereas the pretrained w2v word vectors are set to a dimensionality of 300, unless a costly pretraining process is undertaken. For future studies, we would like to analyze the effect of using different LSA word vector dimensions. Also, we would like to implement other types of traditional word vectors, which could have an effect on larger text datasets.

References

[1] A. J. J. Yepes, L. Plaza, J. Carrillo-de Albornoz, J. G. Mork, and A. R. Aronson, "Feature engineering for medline citation categorization with mesh," *BMC bioinformatics*, vol. 16, no. 1, p. 1, 2015.

[2] K. Javed, S. Maruf, and H. A. Babri, "A two-stage markov blanket based feature selection algorithm for text classification," *Neurocomputing*, vol. 157, pp. 91–104, 2015.

[3] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[4] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive datasets*. Cambridge university press, 2014.

[5] H. Kim, P. Howland, and H. Park, "Dimension reduction in text classification with support vector machines," in *Journal of Machine Learning Research*, 2005, pp. 37–53.

[6] C. Silva, U. Lotrič, B. Ribeiro, and A. Dobnikar, "Distributed text classification with an ensemble kernel-based learning approach," *Systems, Man, and Cybernetics, Part*

C: Applications and Reviews, IEEE Transactions on, vol. 40, no. 3, pp. 287–297, 2010.

[7] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014.

[8] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[9] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using lstm for region embeddings," *arXiv preprint arXiv:1602.02373*, 2016.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[11] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, pp. 806–814, 2016.

[12] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[13] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv:1211.5063*, 2012.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[17] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 512–519.

[18] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[19] E. Altszyler, M. Sigman, S. Ribeiro, and D. F. Slezak, "Comparative study of lsa vs word2vec embeddings in small corpora: a case study in dreams database," *arXiv preprint arXiv:1610.01520*, 2016.

[20] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

[21] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[22] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 90–94.

[23] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[24] "Ag's corpus of news articles," https://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, (Accessed on 05/06/2018).

[25] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[26] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[27] "Cs231n convolutional neural networks for visual recognition," http://cs231n.github.io/convolutional-networks/, (Accessed on 05/06/2018).

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[29] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[32] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, 2016.

[33] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2177–2185.