Indian Institute of Technology
Kanpur

CS671 - Natural Language Processing
Course project

Deep Learning for Document Classification

Amlan Kar
Sanket Jantre

Supervised by
Dr. Amitabha Mukerjee

Contents

1 Abstract

Document Classification techniques have been applied to various tasks, such as automatic tag suggestion, document indexing, sentiment analysis etc. Traditionally, most of these methods involve processes that do not utilize the information in the order of occurence of words, such as BoW models or Tf-Idf techniques to create document vectors. Later, powerful semantic word embeddings emerged, including word2vec and GloVe that have been shown to work well for benchmark sentence classification tasks[1]. Recently, a new semantic sentence embedding method, dubbed Skip-Thoughts[2] has emerged which models sentences as vectors. We intend to explore how a Convolutional Neural Network(CNN) can work with these embeddings to model documents for various classification tasks. We also look at a method suggested in[3] to fine tune pre-trained word vectors to yield much more powerful semantic embeddings.

2 Introduction

In the recent past, deep learning methods have consistently set new benchmarks for a variety of NLP Tasks, such as part-of-speech tagging [4], sentiment classification [5], neural language models [6] and machine translation. These models have been heavily influenced in the recent past by the availability of robust embeddings[1] and a massive increase in computational capabilities with GPUs. Recently, a sentence embedding model, dubbed Skip-Thoughts[2] has emerged, which employs a Gated Recurrent Neural Network based encoder-decoder model to learn generic unsupervised sentence encodings. We attempt to train a convolutional neural network, which given a representation of a document, learns to to perform various Document classification tasks on it. We present the network doing a binary sentiment classification task, but show how other tasks can be easily performed by slightly modifying the network's structure.

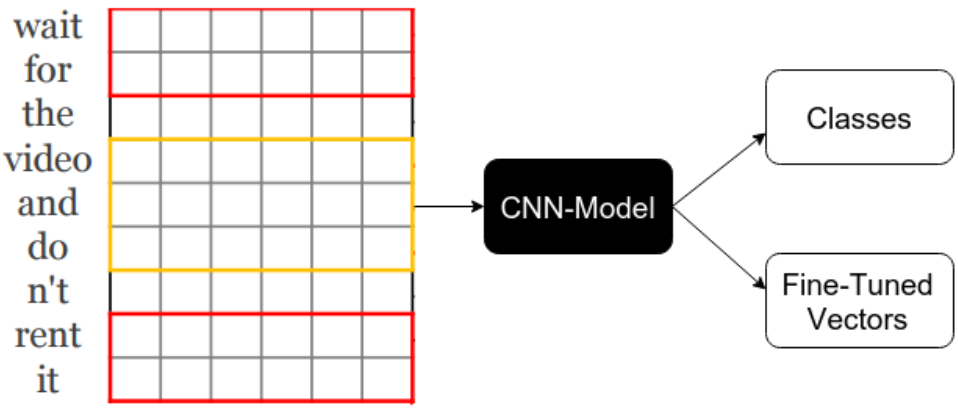


Figure 1: Project Aim

3 Related Work

Our main motivation came from work done in [3] where the author introduces the idea of fine-tuning word vectors and using a convolutional neural network for modeling classification tasks. The work done by Kalchbrenner et al.[7] is another model where sentence representations are being learned within their DCNN(Dynamic Convolutional Neural Network) structure. We use some aspects of their Dynamic Convolutional Network with Y.Kim's model to form a hybrid which beats the state of the art in Sentiment Classification on the Pang-Lee[8] binary sentiment classification dataset.

4 Theory

4.1 Distributed Representations

Distributed Representations are a way to encode data in much lesser space than it would take to store the whole data itself. In a neural network context, we try to learn neurons that learn some features that alone cannot answer a complex question but when activated together, can represent a very intricate concept. It can be seen as a many-to-many relationship between neurons and concepts. So instead of letting each neuron learn each example, we learn a representation of the examples. In NLP, word vectors are exactly distributed representations of words. Each word is represented by a vector in R^k for a decided value of k. The values of different dimensions represent some coarse information, but all dimensions used together represent a succinct encoding of the word such that it retains it's usage such that a certain degree of the trends in the training data can be reproduced. In the case of Mikolov's[1] word vectors, the bag of words model gives each vector a representation that would put it closer to other words that occur in similar contexts.

4.2 Convolutional Neural Networks

A Convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. Convolutional networks were inspired by biological processes and are variations of multilayer perceptrons which are designed to use minimal amounts of preprocessing.[1] They are widely used models for image and video recognition. Recently, these models have been used in Natural Language Processing tasks.

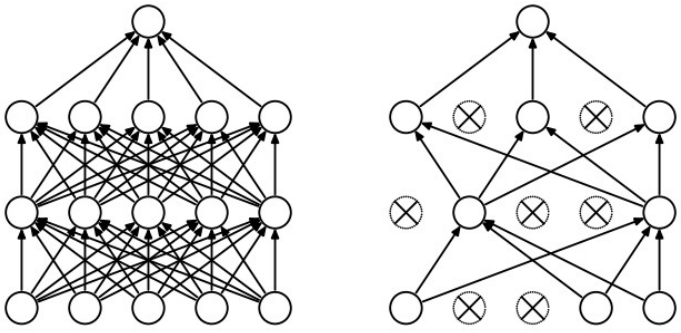
A CNN differs from an ordinary neural network in several ways. First, neurons in a CNN are organized topographically according to dimensions in the input data. So for images, the neurons are laid out on a 2D grid. Similarly in our case, the neurons are laid out in a 2D grid, one dimension representing the word/sentence number in the sentence/document and the second dimension representing the word vector dimension. Second, neurons in a CNN apply local filters and which are centered at the neuron's location in the topographically. This is reasonable for datasets where we expect the dependence of input dimensions to be a decreasing function of distance, which is the case for pixels in natural images. The same might not hold true for Natural Language in all cases, but the concept of n-grams have this implicit assumption. In particular, we expect that useful clues to the identity of the object in an input can

[1]Definition from wikipedia.org

be found by examining small local neighborhoods. Third, all neurons in the input apply the same filter, but as just mentioned, they apply it at different locations in the input. This is reasonable for datasets with roughly stationary statistics, such as natural images, or sentences. We expect that the same kinds of structures can appear at all positions in an input, so it is reasonable to treat all positions equally by filtering them in the same way. In this way, a bank of neurons in a CNN applies a convolution operation to its input. A single layer in a CNN typically has multiple banks of neurons, each performing a convolution with a different filter. These banks of neurons become distinct input channels into the next layer.[2] Non-linearities are applied to the data at the end of a convolution layer. In this project, we employ a ReLU (Rectified Linear Unit) non-linearity which basically is a $max(0, x)$ function. Normally, pooling is also done in CNNs as a sub-sampling method to see larger trends in the local features learnt by the previous layers. Pooling could select the maximum in a neighborhood, average the neighborhood or apply other functions. The first two cases are called max-pooling and average-pooling respectively.

4.3 Dropout

Dropout is a neat method to prevent overfitting in neural networks as shown in [9]. Basically, dropout gives a probability p(generally 0.5) to each node which determines its presence during a training epoch. This ensures that the weights of a neuron do not get too tuned to the data and do not depend a lot on a particular other neuron. This is not applicable during testing, and each node is present normally during testing. Figure 2[3] gives a sketch of the dropout method. This can be seen as an ensemble learner as it is a large combination of different neural networks that give a joint output during testing. In the implementation, the weights of the neural network are multiplied by p as probabilistically, the number of neurons during testing is $1/p$ times the number of neurons during training.



(a) Standard Neural Net (b) After applying dropout.

Figure 2: Dropout

[2]adapted from "Improving neural networks by preventing co-adaptation of feature detectors", Hinton et al.

[3]Image taken from cs231n.github.io

5 Methodology

In this section, we discuss the methodology used in the project. We consider a document/sentence as a 2-D matrix consisting of concatenated vectors of its sentences/words. The size of the input to the convNet is calculated according to the dataset. The height of the 2-D input is set to the maximum length of a document(in sentences)/sentence(in words). Short documents are zero-padded and fed to the Convolutional Neural Network. The ConvNet is trained with 0.5 dropout in the fully connected layers. The adadelta update rule[10] is used to determine the learning rate. The loss function is the negative log-likelihood of the output. In our results, we employ a simple softmax layer to output the probability distribution over the outputs. But since our neural network is small, we do not expect to learn very high level features from the data. This leaves the door open for use of other more complex classifiers such as a Random Forest Classifier to take the pre-final FC layer's activations as inputs for the classification task. Effectively, we would be using the CNN as a feature generator then.

5.1 Flowchart

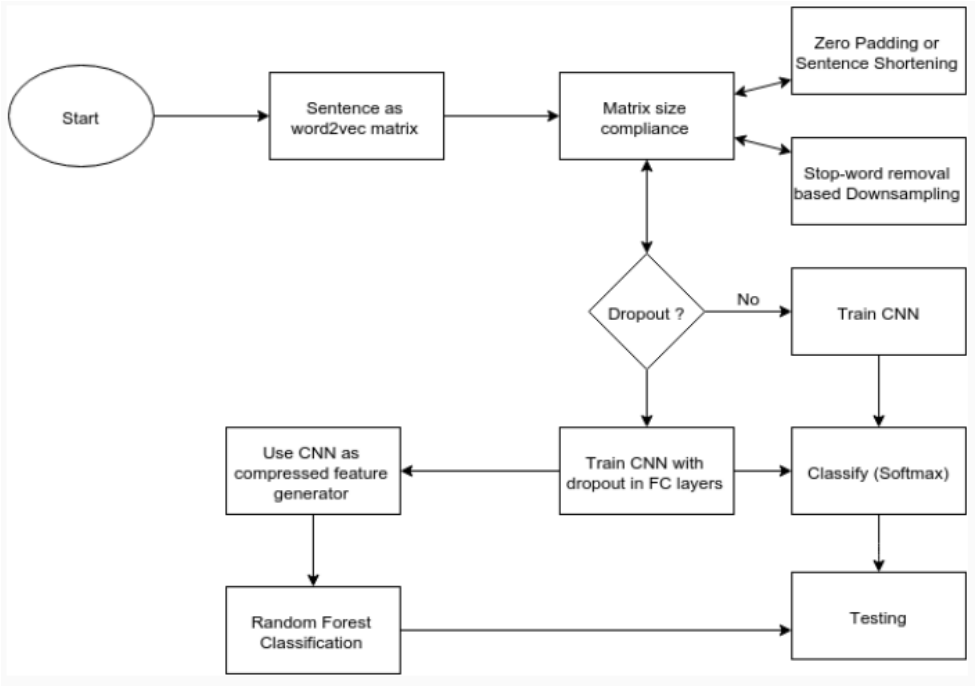


Figure 3: Basic Flowchart

5.2 ConvNet Structure

The Convolutional Neural Network used here is a slight variation of the network proposed in [3] which is based on the structure proposed in [4]. It contains a single convolution layer full-width filters. For our experiment, we use 100 feature maps each of filter heights 3,4 and 5. This gives us a total of 300 filters which are essentially

learning features on 3,4 and 5-grams. We employ wide convolution as suggested in [7]. Wide convolution ensures that all weights in the filters reach all the words/sentences in our input document. This in turn ensures that we do not exclude features that might occur at the edge of a document. Intuitively, for the movie review sentiment classification, it is a trend for reviews to have a short ending or introduction giving most of the sentiment describing data. After the convolution, we employ a k-max pooling of each channel. This gives us an idea of the occurence of a certain feature of a document within 4 chances. This in theory should work better for a larger value of k as we would only include more information with increasing k. But larger values of k also introduce redundancy as there are only a maximum number of times we expect a feature to be seen in a document. In our experiments, we use 4 for the value of k. The k-max pooling layer gives us our feature representation which is then put through a softmax layer to predict the output. We can extend this model to various classification tasks by changing the size of the ouput layer and suitably choosing a loss function for learning. This CNN can also be used as a feature generator to train other classifiers for learning.

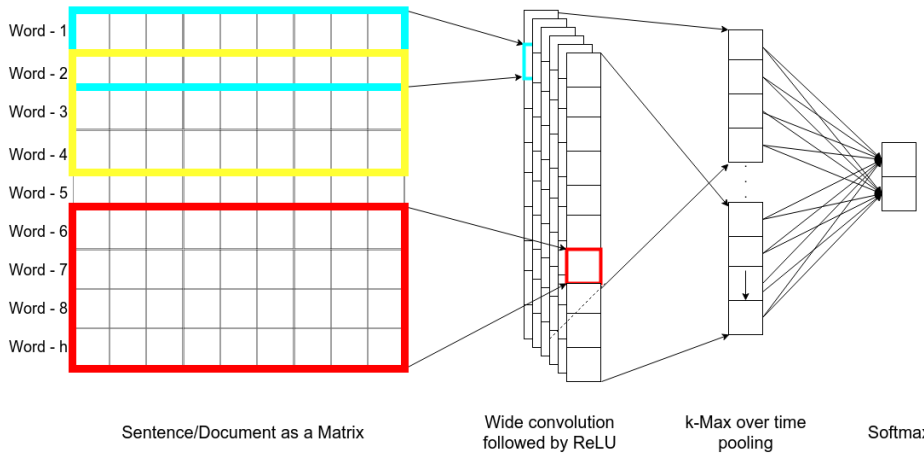


Figure 4: ConvNet Structure

5.3 Datasets

- **Pang-Lee Movie-Review Dataset** [8]

  The Pang-Lee Movie Review dataset is a collection of 10,000 sentences from movie reviews found on rotten tomatoes. These sentences are classified to be positive or negative reviews. The dataset is roughly balanced and has been used as a standard dataset for sentiment classification tasks since 2002.

- **Hindi700 Movie Review Dataset** [11]

  The Hindi movie review dataset was collected by Pranjal Singh for his M.Tech thesis. This dataset contains movie reviews extracted from the Jagran and NavBharat. It contains around 16,000 lines which around 10,000 positive and 6,000 lines of negative reviews.

6 Results

6.1 Classification

6.1.1 Pang-Lee MR

Method	Accuracy
Socher2012	79.0
Dong2014	79.5
Kim2014	81.5
This Method	**81.8**

Table 1: Sentiment Classification on Pang-Lee dataset

6.1.2 Hindi-700

Method	Accuracy
Pranjal2014	**0.91**
Kalchbrenner2014[4]	0.71
This method	0.70

Table 2: Sentiment Classification on Hindi-700

6.2 Fine-Tuning

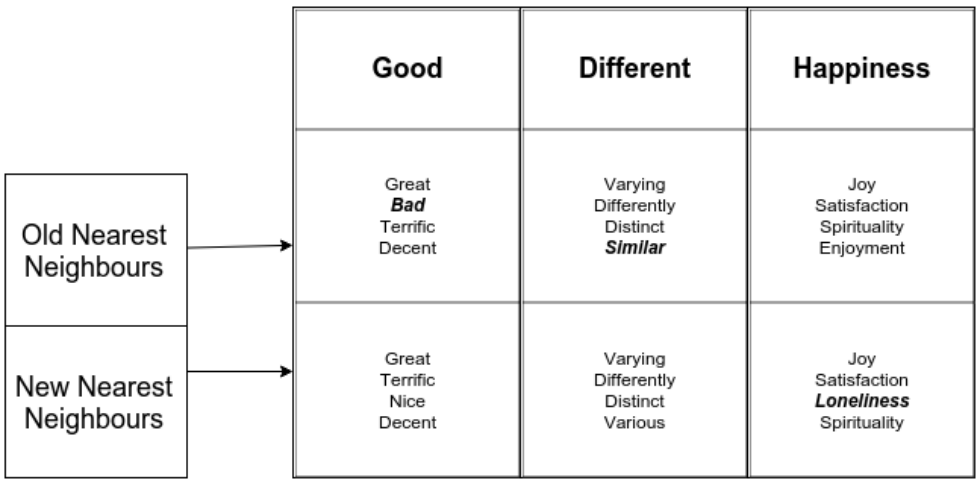


Figure 5: Fine Tuning

6.3 Discussion

The classification accuracy beats Kim's method. We think this is due to the introduction of wide convolution and a k-max pooling layer to the network. As discussed before, the wide convolution ensures that all the inputs get all the values of the filters. Especially in movie reviews, one tends to find small portions in the beginning and the end that contain most of the pertinent information regarding the sentiment being conveyed.

81% for a binary classification task is very low by modern standards. This can be explained to be the property of the dataset itself. The data was collected as many complete movie reviews which were classified as positive or negative. Then each sentence of these reviews was extracted to make an input instance with the same class as the parent document. Some cleaning was done, but this shows the high probability of the data set being skewed and irregularly aligned with respect to the output classes.

The very low classification accuracy on the hindi dataset can be attributed to the weak word vectors that were used in the experiment. 1 in 3 words of the input vocabulary was not even present in the word vectors and these words had to be randomly initialized. The randomly initialized words in the english case were only 1 in 20.

Fine-Tuning does not always fine tune towards more semantic similarity. It fine tunes the word vector to make it work better for the given task. The gradient will make the word vector go closer to other vectors that would co-occur with it in similar examples in the data. As we have shown in the $3rd$ example in the fine tuning results, Happiness being close to Loneliness is clearly a mistake, but these two words seem to be together more in positive reviews of movies. This could be attributed to the large number of inspirational new-age movies and their reviews.

7 Conclusions

- A simple convolutional neural network shows the capacity to learn good features for classification tasks in NLP. This goes in line with the deep learning revolution, where feature learning within the learning framework greatly increases performance over hand-crafted features.
- Accuracies in the testing results obtained from the simple convolutional neural network are surprisingly high. So it is highly expected that when a deeper CNN is would be trained using much bigger dataset then results will be better than we see them now.
- The low performance on the Hindi classification task may be due to lack of availability of robust word vectors like English-300 word vectors. We obtained 100 length Word Vectors [5] trained on a 32GB dataset of hindi sentences. But still, we found that one third of the vocabulary of the Hindi-700 dataset was missing in the word vector vocabulary, which is definitely a big reason for the low performance of the model.

8 Future work

- Use this approach further for multi-class document classification task.
- Fine tune word vectors by training the CNN over a huge corpus to generate more semantically rich word vectors.
- Use this model as a feature generator to train other standard classifiers for various tasks to test the power of the CNN as a robust feature generator.
- Use skip-thought vectors i.e. sentence vectors instead of word vectors in this model to test multi-sentence document level classification tasks.

[5] Thanks to Nirbhay Modhe and Drishti Wali, IIT Kanpur

9 References

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[2] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

[3] Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP 2014*, 2014.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

[5] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[6] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[8] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[10] Matthew D Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[11] Amitabha Mukerjee Pranjal Singh. Decompositional semantics for document embedding. 2015.