Convolutional Neural Network for Modeling Sentences and Sentiment Analysis

Jayesh Kumar Gupta
jayeshkg@iitk.ac.in, 11337

Arpit Shrivastava
shriap@iitk.ac.in, 12161

April 18, 2015

Supervised by Dr. Amitabha Mukerjee

Abstract

Language understanding is the central problem in natural language processing. Critical to this understanding is accurate representation of sentences. We use a novel architecture for neural networks dubbed the Dynamic Convolutional Neural Network (DCNN) for this semantic modeling of sentences. This allows us to handle sentences of varying lengths and capture short and long-range relations. The network is language agnostic as it does not rely on any parse tree. We use this model on the classic NLP problem of sentiment analysis of sentences. We apply this technique to analyze sentiment of labeled Hindi sentences and compare our results with existing methods.

Acknowledgment

The authors wish to thank Prof. Amitabha Mukerjee for giving us the opportunity to work on the project and multiple insights on the way. We would also like to thank Mr. Pranjal Singh for providing us with the Hindi sentence dataset and for the conversations we had with him during the course of our project.

Jayesh Kumar Gupta
jayeshkg@iitk.ac.in
Arpit Shrivastava
shriap@iitk.ac.in

Contents

Chapter 1

Motivation and related work

To perform the classic NLP tasks of sentiment analysis, paraphrase detection, summarization etc. it is important to represent the semantic content of a sentence. We want our feature function to model the sentence in terms of features extracted from the words and n-grams.

Currently the literature is replete with multiple models of meaning. Methods based on composition obtain vectors for longer phrases using the co-occurrence statistics of the vector representations of word meanings. [1] [2]. In other cases, the sentence meaning can be represented by extracted logical forms [3].

Some of the most popular techniques among them are those based on neural networks. These include basic neural bag-of-words [4], recursive neural networks [5], time-delay neural operations [6] etc. These have a number of advantages. Neural networks are especially good at generating generic vectors for words and phrases by taking into account their context [7]. We can then use supervised backpropagation techniques to fine-tune these vectors to perform specific tasks. The model obtained using these techniques is powerful enough to generate sentences word by word [8] [9] [6].

Chapter 2

Background

To understand the introduced neural network model called Dynamic Convolution Neural Network, henceforth referred as DCNN, we need to review related neural sentence models, one-dimensional convolution and pooling operation.

2.1 Convolution

Convolution of two functions is given by the following relation and can be seen as the area overlap between the two.

$$(f * g)(t) = \sum_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.1}$$

In a neural network the convolutional layer contains feature nodes computed for sets of consecutive nodes from the previous layer using a weight matrix. This layer is then fed into a fully-connected layer. This is shown in Figure 2.1.

The features obtained are given by:

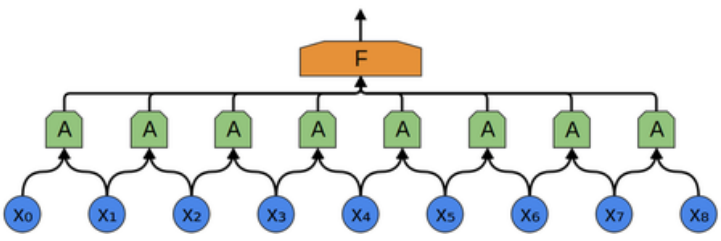

Figure 2.1: Convolution Layer [10]

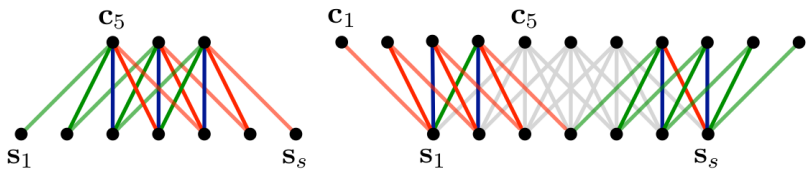

Figure 2.2: Narrow and wide types of convolution. The filter m has size m = 5 [7]

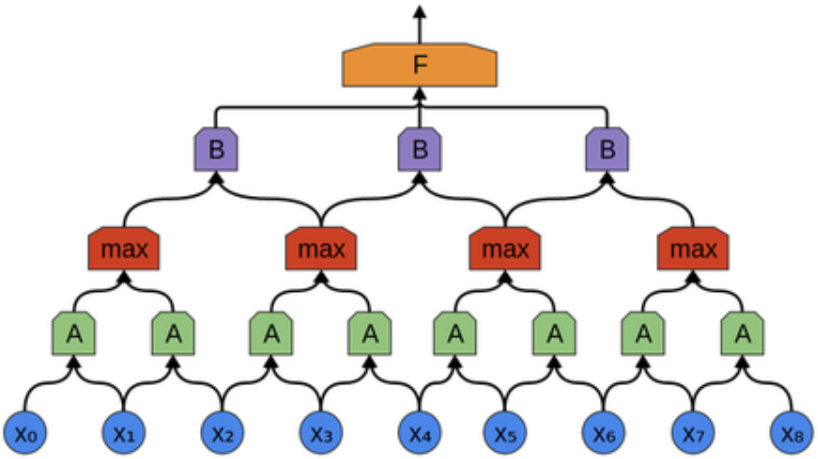

Figure 2.3: Pooling [10]

$$\mathbf{c}_j = \mathbf{m}^\top \mathbf{s}_{j-m+1:j} \tag{2.2}$$

Convolution can be of two types depending on the values of s and m in the above equation. The narrow type of convolution requires that $s \geq m$ and yields a sequence $\mathbf{c} \in \mathbb{R}^{s-m+1}$ whereas wide convolution does not have such a restriction and yields a sequence $\mathbf{c} \in \mathbb{R}^{s+m-1}$. [7]

2.2 Pooling

To make the network resilient to small transformations in the data and better generalization, we take the maximum of features over small blocks in the previous layer. This approach is termed as **max-pooling**.

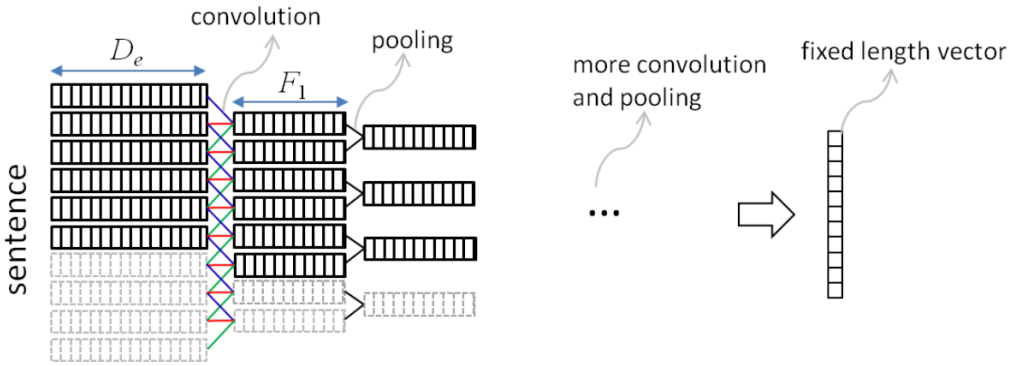

Figure 2.4: A typical CNN [11]

2.3 Convolutional Neural Networks

Convolutional Neural Networks or CNN can be seen as a kind of neural network that uses many identical copies of the same neuron.It can express computationally large models with lesser number of parameters. The network has multiple interleaved convolutional and pooling layers.

Chapter 3

Approach

3.1 Wide Convolution

We start with randomly initialized word embeddings $w_i \in \mathbb{R}^d$ for every word in a sentence to create a sentence matrix $s \in \mathbb{R}^{d \times s}$. We define a filter m as the filter of convolution which is multiplied with every m-gram in the sentence s to obtain the sequence c (Equation 2.2).

Wide convolution ensures that all filter weights reach the entire sentence especially the marginal words. At the same time, we also get a guarantee that application of filter m to the input sentence s always produces a valid non-empty feature vector c [7].

3.2 k-max Dynamic Pooling

Instead of selecting a single feature from the previous layer, k most active features are selected. This allows us to pool together features that may be many positions apart while preserving word order. Although the top most pooling layer's value is fixed to k_{top} to guarantee equal length inputs to the fully connected layers, the value of k is dynamically dependent on the number of current convolution layer l, total number of convolution layers in the network L and the sentence length s:

$$k_l = \max\left(k_{top}, \left\lceil \frac{L-l}{L} s \right\rceil\right) \tag{3.1}$$

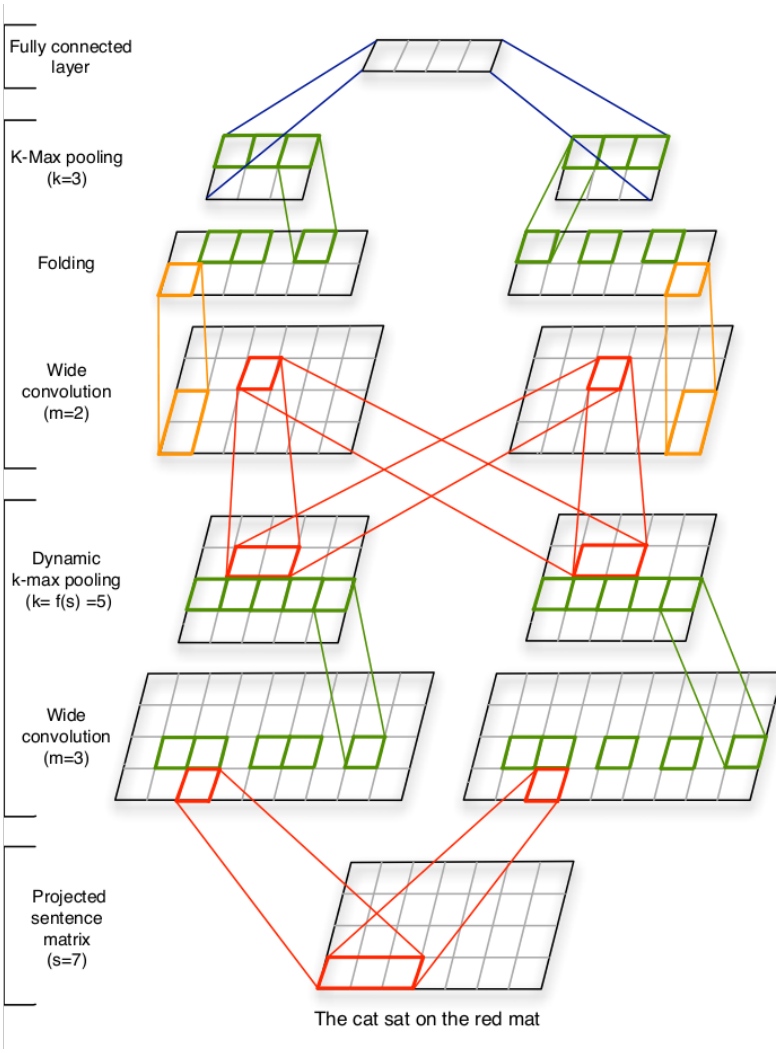

Figure 3.1: A DCNN for the seven word input sentence. Word embeddings have size d = 4. The network has two convolutional layers with two feature maps each. The widths of the filters at the two layers are respectively 3 and 2. [7]

3.3 Non-linear Feature Function

We apply a non-linear activation function $g = \tanh$ and a bias $b \in \mathbb{R}^d$ component wise to the pooled matrix.

$$a = g\left(\mathbf{M}\begin{bmatrix}\mathbf{w}_j \\ \vdots \\ \mathbf{w}_{j+m-1}\end{bmatrix} + \mathbf{b}\right) \tag{3.2}$$

Together with pooling this allows us to achieve position invariance while allowing us to obtain a range of higher order features [7].

3.4 Multiple Feature Maps

All the above three operations can be applied in parallel and repeatedly $(F_1^i, \ldots, F_n^i)$ to obtain to get feature maps of even higher order. Each feature map F_j^i is computed using distinct set of convolution filters arranged in a matrix $m_{j,k}^i$ with each feature map F_k^{i-1} of lower order as:

$$F_j^i = \sum_{k=1}^{n} m_{j,k}^i * F_k^{i-1} \tag{3.3}$$

where $*$ denotes wide convolution [7].

3.5 Folding

Until now in the description of the network, different rows have remained independent of each other until the top fully connected layer. We use a very simple method called *folding* that allows us to introduce dependence between these rows, without using any additional parameters [7]. Between a colvolution layer and a k-max pooling layer, we sum up every two rows in a feature map component wise [7], thus halving the size of the representation.

3.6 Training

The topmost fully connected layer is followed by a softmax non-linearity that allows the network to predict the probability distribution over classes given the input sentence. We train the network to minimize the cross-entropy of the predicted and true distributions [7]. The parameters learned include

word embeddings, convolution filter weights and the fully connected layer weights. We use mini-batch backpropagation and Adagrad [12] update rule based gradient optimization to train the network.

Chapter 4

Sentence Model Properties

Here we describe the properties of sentence model induced by DCNN and the notion of the *feature graph*.

4.1 Word and n-Gram Order

Any good sentence model should be able to capture two particular important features from a given sentence – what relevant words (n-grams) are used in the sentence and where these words (n-grams) occur relative to each other in an input sentence. DCNN is considerate of both these aspects. Wide convolution helps recognize specific n-grams that have size less than or equal to the filter width m [7]. The subsequent pooling operation maintains their order and relative positions while allowing invariance over absolute positions.

4.2 Induced Feature Graph

The neural network with its weighted connections between different layers, forms a directed acyclic graph. Edges of its subgraphs reflect the varying ranges of higher order features – short and focused or global and long as the input sentence. This structure is internal to the network, defined by the feed-forward propagation of the input [7]. Moreover this induced graph structure is more general than a parse tree since it is not limited to syntactically dictated phrases.

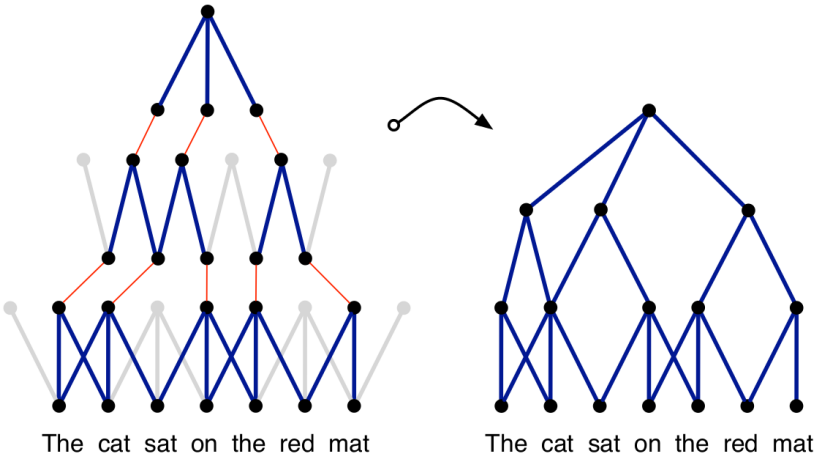

Figure 4.1: Induced feature graph [7]

Chapter 5

Results

Using the datasets as explained in the appendix, the accuracies for different methodologies were as follows:

Experiment	**Features**	**Accuracy**
DCNN	CNN with dynamic k-max pooling	**71.5**
Word Vector with SVM [13]	tf-idf; word vector	89.97
MT Based using SVM [14]	tf-idf	65.96
In language using SVM [14]	tf-idf	78.14

Examples

- *Success*
  - इस फिल्म में काजोल का जरूरी योगदान है। – Positive
  - रण कोई नई बात नहीं कहती. – Negative
- *Failure*
  - यह सब किसके दिमाग की उपज होती है? – Classified as positive
  - यह पुस्तक ईसाई धर्म के अलावा अन्य धर्मों के बच्चों के लिए उपयोगी नहीं होगा – Classified as positive

Certain sentences are pretty confusing. Especially when they are of rhetorical nature. Any neural network model requires a large corpus of datasets for better modeling and higher accuracies. Since the available labeled dataset is quite small in size, we did not get as high accuracies as other methods like word2vec which can find word embeddings in an unsupervised fashion and hence can work on a larger dataset.

Appendix A

Dataset

We trained and tested our code on datasets taken from [14], [15]:

- Product Review dataset (LTG, IIIT Hyderabad) containing 350 Positive reviews and 350 Negative reviews.
- Movie Review dataset (CFILT, IIT Bombay) containing 127 Positive reviews and 125 Negative reviews.

Examples

- *Positive* : मैं इस उत्पाद से बहुत खुश हूँ यह आराम दायक और सुन्दर है यह खरीदने लायक है.
- *Negative* : यह बहुत खराब है और अन्य कार्यक्रमों के साथ काम वास्तव में बाधक हैं.

Preprocessing

Preprocessing involved cleaning the reviews, extracting vocabulary, and representing these reviews as vectors of word indices. The extracted vocabulary consists of 4620 words. We initialize our word embeddings with random values. To handle varying sentence length, we pad the shorter sentences with null character so that it gets easier to input matrices into the network by converting them to equal length sentences. The network still has information about the actual length of the sentences.

Bibliography

[1] Katrin Erk and Sebastian Padó. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics, 2008.

[2] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.

[3] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*, 2012.

[4] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[5] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[6] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, pages 1700–1709, 2013.

[7] Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014.

[8] Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *COLING (Posters)*, pages 1071–1080, 2012.

[9] Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *SLT*, pages 234–239, 2012.

[10] Christopher Olah. Conv nets: A modular perspective. `https://colah.github.io/posts/2014-07-Conv-Nets-Modular/`. Accessed: 10-04-2015.

[11] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050, 2014.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[13] Amitabha Mukerjee Pranjal Singh. Word vector averaging: Parserless approach to sentiment analysis. regICON-2015: Regional Symposium on Natural Language Processing, March 2015.

[14] Aditya Joshi., Balamurali A. R., and Pushpak Bhattacharyya. A fall-back strategy for sentiment analysis in a new language: a case study for hindi. In *International Conference on Natural Language Processing*, pages 1081–1091, 2010.

[15] A.R. Balamurali, Aditya Joshi, and Pushpak Bhattacharyya. Cross-lingual sentiment analysis for Indian languages using linked wordnets. In *Proc. of COLING 2012: Posters*, pages 73–82, Mumbai, India, December 2012.