

A Convolutional Neural Network for Modelling Sentences

Nal Kalchbrenner

Edward Grefenstette

Phil Blunsom

{nal.kalchbrenner, edward.grefenstette, phil.blunsom}@cs.ox.ac.uk

Department of Computer Science
University of Oxford

The ability to accurately represent sentences is central to language understanding. We describe a convolutional architecture dubbed the Dynamic Convolutional Neural Network (DCNN) that we adopt for the semantic modelling of sentences. The network uses Dynamic k -Max Pooling, a global pooling operation over linear sequences. The network handles input sentences of varying length and induces a feature graph over the sentence that is capable of explicitly capturing short and long-range relations. The network does not rely on a parse tree and is easily applicable to any language. We test the DCNN in four experiments: small scale binary and multi-class sentiment prediction, six-way question classification and Twitter sentiment prediction by distant supervision. The network achieves excellent performance in the first three tasks and a greater than 25% error reduction in the last task with respect to the strongest baseline.

1 Introduction

The aim of a sentence model is to analyse and represent the semantic content of a sentence for purposes of classification or generation. The sentence modelling problem is at the core of many tasks involving a degree of natural language comprehension. These tasks include sentiment analysis, paraphrase detection, entailment recognition, summarisation, discourse analysis, machine translation, grounded language learning and image retrieval. Since individual sentences are rarely observed or not observed at all, one must represent a sentence in terms of features that depend on the words and short n -grams in the sentence that are frequently observed. The core of a sentence model involves a feature function that defines the process

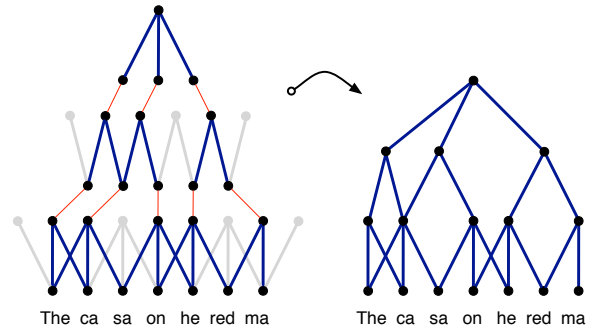


Figure 1: Subgraph of a feature graph induced over an input sentence in a Dynamic Convolutional Neural Network. The full induced graph has multiple subgraphs of this kind with a distinct set of edges; subgraphs may merge at different layers. The left diagram emphasises the pooled nodes. The width of the convolutional filters is 3 and 2 respectively. With dynamic pooling, a filter with small width at the higher layers can relate phrases far apart in the input sentence.

by which the features of the sentence are extracted from the features of the words or n -grams.

Various types of models of meaning have been proposed. Composition based methods have been applied to vector representations of word meaning obtained from co-occurrence statistics to obtain vectors for longer phrases. In some cases, composition is defined by algebraic operations over word meaning vectors to produce sentence meaning vectors (Erk and Padó, 2008; Mitchell and Lapata, 2008; Mitchell and Lapata, 2010; Turney, 2012; Erk, 2012; Clarke, 2012). In other cases, a composition function is learned and either tied to particular syntactic relations (Guevara, 2010; Zanzotto et al., 2010) or to particular word types (Baroni and Zamparelli, 2010; Coecke et al., 2010; Grefenstette and Sadrzadeh, 2011; Kartsaklis and Sadrzadeh, 2013; Grefenstette, 2013). Another approach represents the meaning of sentences by way of automatically extracted logical forms (Zettlemoyer and Collins, 2005).

A central class of models are those based on neural networks. These range from basic neural bag-of-words or bag-of- n -grams models to the more structured recursive neural networks and to time-delay neural networks based on convolutional operations (Collobert and Weston, 2008; Socher et al., 2011; Kalchbrenner and Blunsom, 2013b). Neural sentence models have a number of advantages. They can be trained to obtain generic vectors for words and phrases by predicting, for instance, the contexts in which the words and phrases occur. Through supervised training, neural sentence models can fine-tune these vectors to information that is specific to a certain task. Besides comprising powerful classifiers as part of their architecture, neural sentence models can be used to condition a neural language model to generate sentences word by word (Schwenk, 2012; Mikolov and Zweig, 2012; Kalchbrenner and Blunsom, 2013a).

We define a convolutional neural network architecture and apply it to the semantic modelling of sentences. The network handles input sequences of varying length. The layers in the network interleave one-dimensional convolutional layers and dynamic k -max pooling layers. Dynamic k -max pooling is a generalisation of the max pooling operator. The max pooling operator is a non-linear subsampling function that returns the maximum of a set of values (LeCun et al., 1998). The operator is generalised in two respects. First, k -max pooling over a linear sequence of values returns the subsequence of k maximum values in the sequence, instead of the single maximum value. Secondly, the pooling parameter k can be dynamically chosen by making k a function of other aspects of the network or the input.

The convolutional layers apply one-dimensional filters across each row of features in the sentence matrix. Convolving the same filter with the n -gram at every position in the sentence allows the features to be extracted independently of their position in the sentence. A convolutional layer followed by a dynamic pooling layer and a non-linearity form a feature map. Like in the convolutional networks for object recognition (LeCun et al., 1998), we enrich the representation in the first layer by computing multiple feature maps with different filters applied to the input sentence. Subsequent layers also have multiple feature maps computed by convolving filters with all the maps from the layer below. The weights at

these layers form an order-4 tensor. The resulting architecture is dubbed a Dynamic Convolutional Neural Network.

Multiple layers of convolutional and dynamic pooling operations induce a structured feature graph over the input sentence. Figure 1 illustrates such a graph. Small filters at higher layers can capture syntactic or semantic relations between non-continuous phrases that are far apart in the input sentence. The feature graph induces a hierarchical structure somewhat akin to that in a syntactic parse tree. The structure is not tied to purely syntactic relations and is internal to the neural network.

We experiment with the network in four settings. The first two experiments involve predicting the sentiment of movie reviews (Socher et al., 2013b). The network outperforms other approaches in both the binary and the multi-class experiments. The third experiment involves the categorisation of questions in six question types in the TREC dataset (Li and Roth, 2002). The network matches the accuracy of other state-of-the-art methods that are based on large sets of engineered features and hand-coded knowledge resources. The fourth experiment involves predicting the sentiment of Twitter posts using distant supervision (Go et al., 2009). The network is trained on 1.6 million tweets labelled automatically according to the emoticon that occurs in them. On the hand-labelled test set, the network achieves a greater than 25% reduction in the prediction error with respect to the strongest unigram and bigram baseline reported in Go et al. (2009).

The outline of the paper is as follows. Section 2 describes the background to the DCNN including central concepts and related neural sentence models. Section 3 defines the relevant operators and the layers of the network. Section 4 treats of the induced feature graph and other properties of the network. Section 5 discusses the experiments and inspects the learnt feature detectors.¹

2 Background

The layers of the DCNN are formed by a convolution operation followed by a pooling operation. We begin with a review of related neural sentence models. Then we describe the operation of *one-dimensional convolution* and the classical Time-Delay Neural Network (TDNN) (Hinton, 1989; Waibel et al., 1990). By adding a max pooling

¹Code available at www.nal.co

layer to the network, the TDNN can be adopted as a sentence model (Collobert and Weston, 2008).

2.1 Related Neural Sentence Models

Various neural sentence models have been described. A general class of basic sentence models is that of Neural Bag-of-Words (NBoW) models. These generally consist of a projection layer that maps words, sub-word units or n -grams to high dimensional embeddings; the latter are then combined component-wise with an operation such as summation. The resulting combined vector is classified through one or more fully connected layers.

A model that adopts a more general structure provided by an external parse tree is the Recursive Neural Network (RecNN) (Pollack, 1990; Küchler and Goller, 1996; Socher et al., 2011; Hermann and Blunsom, 2013). At every node in the tree the contexts at the left and right children of the node are combined by a classical layer. The weights of the layer are shared across all nodes in the tree. The layer computed at the top node gives a representation for the sentence. The Recurrent Neural Network (RNN) is a special case of the recursive network where the structure that is followed is a simple linear chain (Gers and Schmidhuber, 2001; Mikolov et al., 2011). The RNN is primarily used as a language model, but may also be viewed as a sentence model with a linear structure. The layer computed at the last word represents the sentence.

Finally, a further class of neural sentence models is based on the convolution operation and the TDNN architecture (Collobert and Weston, 2008; Kalchbrenner and Blunsom, 2013b). Certain concepts used in these models are central to the DCNN and we describe them next.

2.2 Convolution

The *one-dimensional convolution* is an operation between a vector of weights $\mathbf{m} \in \mathbb{R}^m$ and a vector of inputs viewed as a sequence $\mathbf{s} \in \mathbb{R}^s$. The vector $\mathbf{m}$ is the *filter* of the convolution. Concretely, we think of $\mathbf{s}$ as the input sentence and $s_i \in \mathbb{R}$ is a single feature value associated with the i -th word in the sentence. The idea behind the one-dimensional convolution is to take the dot product of the vector $\mathbf{m}$ with each m -gram in the sentence $\mathbf{s}$ to obtain another sequence $\mathbf{c}$:

$$\mathbf{c}_j = \mathbf{m}^T \mathbf{s}_{j-m+1:j} \quad (1)$$

Equation 1 gives rise to two types of convolution depending on the range of the index j . The *narrow* type of convolution requires that $s \geq m$ and yields

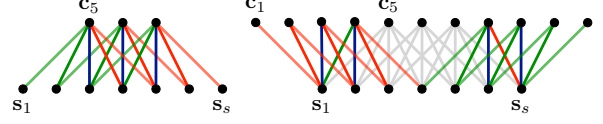


Figure 2: Narrow and wide types of convolution. The filter $\mathbf{m}$ has size $m = 5$.

a sequence $\mathbf{c} \in \mathbb{R}^{s-m+1}$ with j ranging from m to s . The *wide* type of convolution does not have requirements on s or m and yields a sequence $\mathbf{c} \in \mathbb{R}^{s+m-1}$ where the index j ranges from 1 to $s + m - 1$. Out-of-range input values s_i where $i < 1$ or $i > s$ are taken to be zero. The result of the narrow convolution is a subsequence of the result of the wide convolution. The two types of one-dimensional convolution are illustrated in Fig. 2.

The trained weights in the filter $\mathbf{m}$ correspond to a linguistic feature detector that learns to recognise a specific class of n -grams. These n -grams have size $n \leq m$, where m is the width of the filter. Applying the weights $\mathbf{m}$ in a wide convolution has some advantages over applying them in a narrow one. A wide convolution ensures that all weights in the filter reach the entire sentence, including the words at the margins. This is particularly significant when m is set to a relatively large value such as 8 or 10. In addition, a wide convolution guarantees that the application of the filter $\mathbf{m}$ to the input sentence $\mathbf{s}$ always produces a valid non-empty result $\mathbf{c}$, independently of the width m and the sentence length s . We next describe the classical convolutional layer of a TDNN.

2.3 Time-Delay Neural Networks

A TDNN convolves a sequence of inputs $\mathbf{s}$ with a set of weights $\mathbf{m}$. As in the TDNN for phoneme recognition (Waibel et al., 1990), the sequence $\mathbf{s}$ is viewed as having a time dimension and the convolution is applied over the time dimension. Each s_j is often not just a single value, but a vector of d values so that $\mathbf{s} \in \mathbb{R}^{d \times s}$. Likewise, $\mathbf{m}$ is a matrix of weights of size $d \times m$. Each row of $\mathbf{m}$ is convolved with the corresponding row of $\mathbf{s}$ and the convolution is usually of the narrow type. Multiple convolutional layers may be stacked by taking the resulting sequence $\mathbf{c}$ as input to the next layer.

The Max-TDNN sentence model is based on the architecture of a TDNN (Collobert and Weston, 2008). In the model, a convolutional layer of the narrow type is applied to the sentence matrix $\mathbf{s}$, where each column corresponds to the feature vec-

for $\mathbf{w}_i \in \mathbb{R}^d$ of a word in the sentence:

$$\mathbf{s} = \begin{bmatrix} | & | & | & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_s \\ | & | & | & | \end{bmatrix} \quad (2)$$

To address the problem of varying sentence lengths, the Max-TDNN takes the **maximum** of each row in the resulting matrix $\mathbf{c}$ yielding a vector of d values:

$$\mathbf{c}_{max} = \begin{bmatrix} \max(\mathbf{c}_{1,:}) \\ \vdots \\ \max(\mathbf{c}_{d,:}) \end{bmatrix} \quad (3)$$

The aim is to capture the most relevant feature, i.e. the one with the highest value, for each of the d rows of the resulting matrix $\mathbf{c}$. The fixed-sized vector $\mathbf{c}_{max}$ is then used as input to a fully connected layer for classification.

The Max-TDNN model has many desirable properties. It is sensitive to the order of the words in the sentence and it does not depend on external language-specific features such as dependency or constituency parse trees. It also gives largely uniform importance to the signal coming from each of the words in the sentence, with the exception of words at the margins that are considered fewer times in the computation of the narrow convolution. But the model also has some limiting aspects. The range of the feature detectors is limited to the span m of the weights. Increasing m or stacking multiple convolutional layers of the narrow type makes the range of the feature detectors larger; at the same time it also exacerbates the neglect of the margins of the sentence and increases the minimum size s of the input sentence required by the convolution. For this reason higher-order and long-range feature detectors cannot be easily incorporated into the model. The max pooling operation has some disadvantages too. It cannot distinguish whether a relevant feature in one of the rows occurs just one or multiple times and it forgets the order in which the features occur. More generally, the pooling factor by which the signal of the matrix is reduced at once corresponds to $s - m + 1$; even for moderate values of s the pooling factor can be excessive. The aim of the next section is to address these limitations while preserving the advantages.

3 Convolutional Neural Networks with Dynamic k -Max Pooling

We model sentences using a convolutional architecture that alternates wide convolutional layers

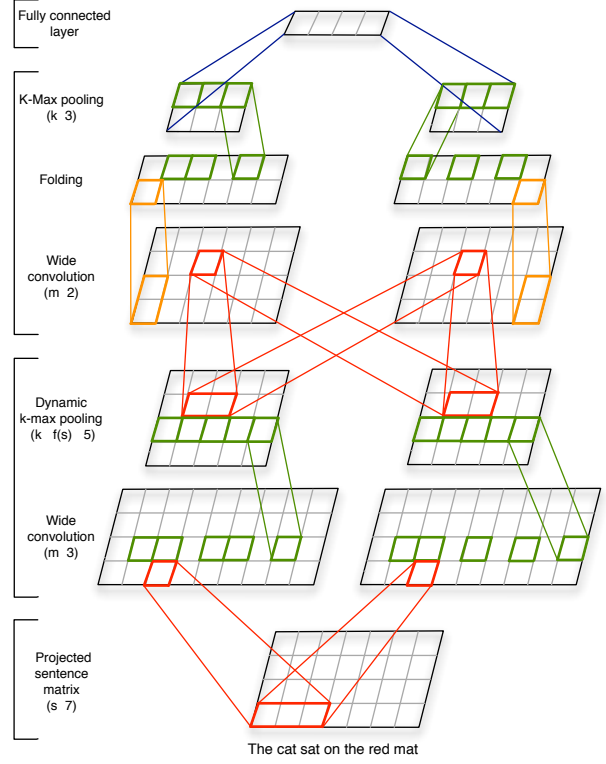


Figure 3: A DCNN for the seven word input sentence. Word embeddings have size $d = 4$. The network has two convolutional layers with two feature maps each. The widths of the filters at the two layers are respectively 3 and 2. The (dynamic) k -max pooling layers have values k of 5 and 3.

with dynamic pooling layers given by *dynamic k -max pooling*. In the network the width of a feature map at an intermediate layer varies depending on the length of the input sentence; the resulting architecture is the Dynamic Convolutional Neural Network. Figure 3 represents a DCNN. We proceed to describe the network in detail.

3.1 Wide Convolution

Given an input sentence, to obtain the first layer of the DCNN we take the embedding $\mathbf{w}_i \in \mathbb{R}^d$ for each word in the sentence and construct the sentence matrix $\mathbf{s} \in \mathbb{R}^{d \times s}$ as in Eq. 2. The values in the embeddings $\mathbf{w}_i$ are parameters that are optimised during training. A convolutional layer in the network is obtained by convolving a matrix of weights $\mathbf{m} \in \mathbb{R}^{d \times m}$ with the matrix of activations at the layer below. For example, the second layer is obtained by applying a convolution to the sentence matrix $\mathbf{s}$ itself. Dimension d and filter width m are hyper-parameters of the network. We let the operations be *wide* one-dimensional convolutions as described in Sect. 2.2. The resulting matrix $\mathbf{c}$ has dimensions $d \times (s + m - 1)$.

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/167031061033006115>