Text Coherence Analysis Based on Deep Neural Network

Baiyun Cui, Yingming Li[†], Yaqing Zhang and Zhongfei Zhang
Zhejiang University, China
baiyunc@yahoo.com,{yingming,yaqing,zhongfei}@zju.edu.cn

ABSTRACT

In this paper, we propose a novel deep coherence model (DCM) using a convolutional neural network architecture to capture the text coherence. The text coherence problem is investigated with a new perspective of learning sentence distributional representation and text coherence modeling simultaneously. In particular, the model captures the interactions between sentences by computing the similarities of their distributional representations. Further, it can be easily trained in an end-to-end fashion. The proposed model is evaluated on a standard Sentence Ordering task. The experimental results demonstrate its effectiveness and promise in coherence assessment showing a significant improvement over the state-of-the-art by a wide margin.

CCS CONCEPTS

• **Information systems** → *Content analysis and feature selection*;

KEYWORDS

deep coherence model; distributional representation; coherence analysis

1 INTRODUCTION

Coherence is a key property of any well-organized text. It evaluates the degree of logical consistency for text and can help document a set of sentences into a logically consistent order, which is at the core of many text-synthesis tasks such as text generation and multi-document summarization. An example is shown in Table 1 for the coherence problem.

Although coherence is significant in constructing a meaningful and logical multi-sentence text, it is difficult to capture and measure as the concept of coherence is too abstract. The problem of coherence assessment was first proposed in 1980s, and since then a variety of coherence analysis methods have been developed, such as the centering theory [8, 16] which establishes constraints on the distribution of discourse entities in coherent text, and the content approaches [2, 7] as the extensions of HMMs for global coherence which consider text as a sequence of topics and represent topic shifts within a specific domain.

Table 1: Examples of a coherent text and an incoherent one.

Text 1	Text 2
Tom loves reading books.	Tom loves reading books.
He prefers reading books at library.	He missed his lunch today.
So he always goes to library.	So he always goes to library.
label=1 (coherent)	label=0 (incoherent)

Another widely used type of approaches in the literature is to encode input text into a set of sophisticated lexical and syntactic features, and then apply machine learning methods (e.g., SVM) to measure coherence between these representations based on the features. Features being explored include entity-based features [1], syntactic patterns [12], conference clues to ordering [5], named-entity features [6], and others. But, identifying and defining those features are always an empirical process which requires considerable experience and domain expertise.

Recently, a promising coherence framework [11] has been proposed based on a deep learning framework, where it adopts recurrent and recursive neural networks [13, 14] in computing vectors for input sentences. However, it pays little attention to essential semantic interactions between sentences, which are also necessary in coherence assessment. Furthermore, in the recurrent framework, terms are simply piled up within a sentence such that long-distance dependencies are difficult to capture due to the vanishing gradient problem [3] and on the other hand, the recursive neural network still suffers from a severe dependence on external resources to construct its syntactic trees.

To overcome the above limitations, in this work, we present a novel deep coherence model (DCM) based on convolutional neural networks to learn coherence for the given text. We study the text coherence problem with a new perspective of learning sentence distributional representation and text coherence modeling simultaneously. In particular, word embeddings are first explored to generate sentence matrix for each sentence [10, 15, 17, 18], and then sentence models map sentences to distributional vectors in parallel, which are used for learning coherence between them. Further, interactions between sentences are captured by computing the similarities of their distributional representations. Finally, the sentence vectors and their corresponding similarity scores are concatenated together to estimate the text coherence.

Our work differs from the existing approaches in several important aspects: 1) we propose a distributional sentence model based on convolutional neural networks (CNNs) to map input sentences to advanced representations; 2) our architecture uses intermediate sentence vectors to compute their similarity scores and includes them in the final representation, which constitutes a much richer representation of text.

[†] Corresponding author

CIKM'17 , November 6–10, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4918-5/17/11...$15.00
https://doi.org/10.1145/3132847.3133047

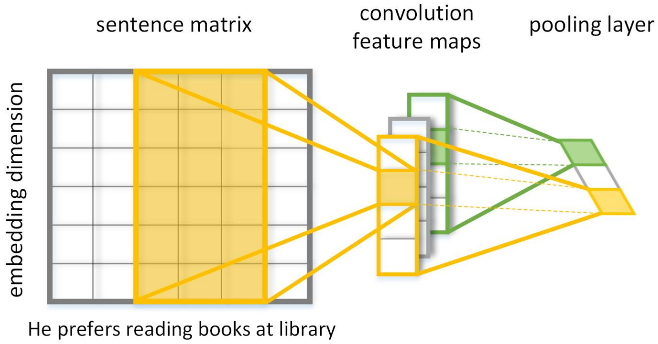

Figure 1: The sentence model based on a CNN for distributional representation.

The proposed model is evaluated on a standard Sentence Ordering task. The experimental results demonstrate the effectiveness and promise in coherence assessment showing considerable improvements over the state-of-the-art literature [11] by a wide margin.

2 MODEL CONSTRUCTION

In this section, we first introduce how to compute distributional sentence vectors based on CNNs and word embeddings. Then, a framework for evaluating the coherence of a sequence of sentences is proposed with the sentence representations.

2.1 Distributional Sentence Representation

Given a sequence of sentences, as is shown in Figure 1, the proposed sentence model is able to map each sentence into a distributional vector, and then the dense sentence representation is transformed through a wide convolutional layer, a non-linearity and a max pooling layer into a low-dimensional and real-valued feature vector.

In the following, we describe the main building blocks of our sentence model in details: sentence matrix and CNN including convolutional layers, activations and pooling layers.

2.1.1 Sentence Matrix. Since the input sentence is comprised of several raw words which cannot be directly processed by subsequent layers of the network, we adopt distributional word embeddings to translate the words into real-valued feature vectors and then combine them to form a sentence matrix.

The input sentence **s** consists of a sequence of words: $[w_1, \ldots, w_{|s|}]$, where $|s|$ denotes the total number of words within the sentence. Word embeddings matrix $\mathbf{W} \in \mathbb{R}^{d \times |V|}$ is formed by concatenating embeddings of all words in a finitely sized vocabulary V, where d denotes the dimension of this embedding. Each word is mapped to integer indices $1, \ldots, |V|$ in vocabulary V and then represented by a distributional vector $\mathbf{w} \in \mathbb{R}^d$ looked up in this word embeddings matrix. Hence, a sentence matrix $\mathbf{S} \in \mathbb{R}^{d \times |s|}$ is established for each input sentence **s**, where each i-th column represents a word embedding $\mathbf{w}_i$ of the i-th word in the sentence:

$$\mathbf{S} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{|s|}] \tag{1}$$

So far we have obtained a sentence matrix **S**. In the following, a CNN is applied to the input sentence matrix to capture higher level semantic concepts, using a series of transformations including convolution, nonlinearity and pooling operations.

2.1.2 Convolutional Neural Network. The aim of the convolutional layer is to extract useful patterns using different filters which represent a variety of significant features of the input sentence.

Corresponding to the input $\mathbf{S} \in \mathbb{R}^{d \times |s|}$, a convolution filter is also a matrix of weights $\mathbf{F} \in \mathbb{R}^{d \times m}$ with width m and has the same dimensionality d as the given sentence matrix. As shown in Figure 1, the filter slides along the column dimension of **S** producing a vector $\mathbf{c} \in \mathbb{R}^{|s|-m+1}$ as an output, where each component is computed as follows:

$$c_i = (\mathbf{S} * \mathbf{F})_i = \sum_{k,j} (\mathbf{S}_{[:,\, i-m+1:i]} \otimes \mathbf{F})_{kj} \tag{2}$$

where $\mathbf{S}_{[:,\, i-m+1:i]}$ is a matrix slice with size m along the columns and $\otimes$ is the element-wise multiplication. Essentially, in order to capture more features and build a richer representation for the input sentence, the networks apply a set of filters sequentially convolved with the distributional sentence matrix **S**. Such filter banks $\mathbf{F} \in \mathbb{R}^{n \times d \times m}$ work in parallel generating multiple feature maps of dimension $n \times (|s| - m + 1)$.

After convolution operations, we apply a non-linear activation function $\alpha()$ to learn nonlinear decision boundaries and adopt a rectified linear (ReLU) function defined as $max(0, \mathbf{x})$ which can not only speed up the training process but also sometimes increase the accuracy. Furthermore, we add *max* pooling layer to the distributional sentence model aiming to reduce the representation and aggregate the information. This *max* pooling operates on columns of the feature map matrix **C** and enables to return the maximum value of the output from the convolutional layer as follows: pool($\mathbf{c}$): $\mathbb{R}^{|s|-m+1} \rightarrow \mathbb{R}$, which has just passed through the activation function.

2.2 Coherence Computation

Here we explain how to map several input sentences to the final coherence probability and provide a full description of the remaining components in the networks, e.g., similarity matrix, join layer, hidden and softmax layer.

We first define a window of sentences as a clique q and associate each clique with a label y_q that indicates its coherence, where y_q takes the value 1 if coherent, and 0 otherwise. Consequently, for a document D consisting of N sentences $D = \{s_1, s_2, \ldots, s_N\}$, it is comprised of N_d cliques. Taking window size $L = 3$ for example, $N_d = N - 2$, and the cliques we need to consider are as follows:

$$< s_1, s_2, s_3 >, < s_2, s_3, s_4 >, \ldots, < s_{N-2}, s_{N-1}, s_N > \tag{3}$$

To articulate clearly the coherence computation methodology, in the following we use the case of a clique of three neighboring sentences to present the methodology and the architecture of our model is shown in Figure 2. The method, however, can be implemented using a clique of any number of neighboring sentences and in fact in the experiments we have implemented and evaluated the method in different clique sizes. It appears that the performance differences for different clique sizes are not significant.

Similarity computation. Since sentences in coherent text always talk about a main topic and share some events and emotions in common, we compute sentence-to-sentence semantic similarity

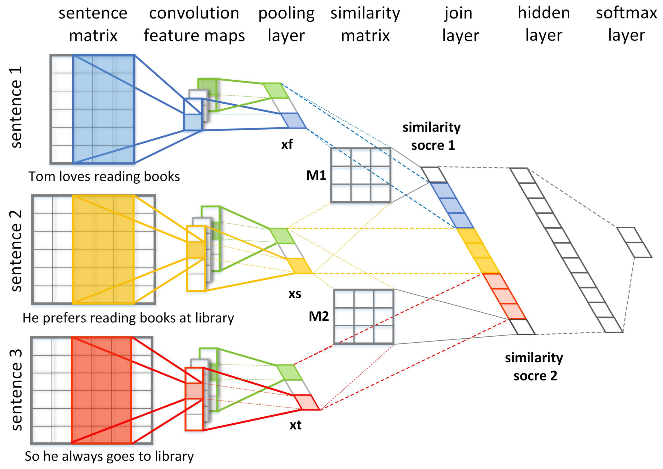

Figure 2: The architecture of our deep coherence model (DCM) for text coherence analysis with two matrices to encode similarity between adjacent sentences.

to encode this essential information which can certainly produce positive effects on coherence assessment. Assume that the output of our sentence model is three distributional representations: the first one $\mathbf{x}_f$, the second one $\mathbf{x}_s$ and the third one $\mathbf{x}_t$. Following the approach used in [4], we define the similarity between any neighboring sentence vectors as follows:

$$Sim(fs) = \mathbf{x}_f^T \mathbf{M}_1 \mathbf{x}_s \tag{4}$$

the similarity matrix $\mathbf{M}_1$ is a parameter of the network that can be optimized during the training. In this model, more common elements between these two vectors, closer $\mathbf{x}_s' = \mathbf{M}_1 \mathbf{x}_s$ is to $\mathbf{x}_f$, and thus the higher similarity score $Sim(fs)$.

Join layer. For the coherence assessment of the three input sentences, similarity computation produces two single scores: $Sim(fs)$ and $Sim(st)$ capturing syntactic and semantic aspects of the similarity from the input three-sentence text. Additionally, along with two similarity scores, our architecture also includes intermediate representations of the three sentences into the final vector

$$\mathbf{x}_{join} = [\mathbf{x}_f^T, Sim(fs), \mathbf{x}_s^T, Sim(st), \mathbf{x}_t^T] \tag{5}$$

which together constitute a much richer final representation for computing the final coherence probability.

Hidden layer. The hidden layer performs the following transformation: $\mathbf{h} = f(\mathbf{W}_h \mathbf{x}_{join} + \mathbf{b}_h)$, where $\mathbf{W}_h$ is the weight matrix of the hidden layer, $\mathbf{b}_h$ is a bias vector and $f()$ is the non-linearity function. The output of the hidden layer $\mathbf{h}$ can be viewed as a final abstract representation obtained by a series of transformations from the input layer through a series of convolutional and pooling operations.

Softmax layer. The output of the hidden layer $\mathbf{h}$ is further fed to the fully connected softmax classification layer, and the coherence probability of this three-sentence text can be summarized as:

$$p(y_q|\mathbf{x}_{join}) = \text{sigmod}(\mathbf{W}_s \mathbf{h} + \mathbf{b}_s) \tag{6}$$

where $\mathbf{W}_s$ is a weight matrix and $\mathbf{b}_s$ denotes the bias.

2.3 Training

Parameters in our deep neural network include: the word embeddings matrix $\mathbf{W}$, filter weights and biases of the convolutional layers in sentence model, two similarity matrices $\mathbf{M}_1$ and $\mathbf{M}_2$, and parameters of the hidden and softmax layers. We use θ to represent them:

$$\theta = \{\mathbf{W}; \mathbf{F}_{x_f}; \mathbf{b}_{x_f}; \mathbf{F}_{x_s}; \mathbf{b}_{x_s}; \mathbf{F}_{x_t}; \mathbf{b}_{x_t}; \mathbf{M}_1; \mathbf{M}_2; \mathbf{W}_h; \mathbf{b}_h; \mathbf{W}_s; \mathbf{b}_s\} \tag{7}$$

and the negative conditional log-likelihood of the training set is:

$$C = -\log \prod_{i=1}^{N_t} p(y_q|\mathbf{g}^{(i)}; \theta) \tag{8}$$

where $\mathbf{g}^{(i)} = (\mathbf{x}_f^i, \mathbf{x}_s^i, \mathbf{x}_t^i)$ denotes the i-th training clique of three neighboring sentences and N_t indicates the number of training cliques. We train the overall model to minimize this function and optimize parameters in the network by computing their gradients within shuffled mini-batches based on back propagation algorithm.

3 DOCUMENT COHERENCE ASSESSMENT

In this section, we apply the proposed framework to evaluate coherence for any documents with varying lengths. With the definition of a clique in Section 2.2, the function to compute the coherence score for a whole document is given by [11]:

$$S_D = \prod_{q \in D} p(y_q = 1) \tag{9}$$

It is reasonable to choose product operations rather than plus operations as the coherence of the whole document is related to the coherence of each clique, and any incoherent clique would have an extremely adverse impact to the entire document. For document pair $< D_1, D_2 >$, if $S_{D_1} > S_{D_2}$, we would say document D_1 is more coherent than D_2.

4 COHERENCE EXPERIMENTS

We evaluate the proposed coherence modeling framework on a common evaluation task usually adopted in the existing literature: Sentence Ordering.

Data. We employ two corpora which are widely used in this task [1, 2, 5, 6, 12]. The first is a collection of aviation accident reports written by officials from the National Transportation Safety Board and the second contains Associated Press articles from the North American News Corpus on the topic of earthquakes. The size of the word vocabulary V for the experiments using accident corpus is 4501 and with approximately 11.5 sentences per document on average. For the earthquake corpus, $|V| = 3022$ with about 10.4 sentences per document on average. Following the setup of [11], 100 source articles are used for training, and 100 (accidents) and 99 (earthquakes) are used for testing. A maximum of 20 random permutations were generated for each test article to create the pairwise data. Positive cliques are directly obtained from the original training document and negative examples are created by random permutations of its sentences within the document. Moreover, like the method in [17], we employ the word2vec tool to compute the word embeddings for sentence matrix construction.

Baselines. To demonstrate that the CNN truly improves the coherence assessment performance in comparison with the state-of-the-art methods, we compare DCM with the following representative

Table 2: Survey of the results with average accuracy in two corpora on the Sentence Ordering task.

Model	Accident	Earthquake	Average
DCM	**0.950**	**0.995**	**0.973**
DCM_Nosim	0.925	0.986	0.956
Recursive	0.864	0.976	0.920
Recurrent	0.840	0.951	0.895
Entity Grid	0.904	0.872	0.888
HMM	0.822	0.938	0.880
HMM+Content	0.742	0.953	0.848
Conference+Syntax	0.765	0.888	0.827
Graph	0.846	0.635	0.741

methods: Recursive [11], Recurrent [11], Entity Grid [1], HMM [12], HMM+Content [12], Conference+Syntax [1], and Graph [9].

In addition, to verify the effectiveness of the similarity building blocks in the deep learning architecture, we also study a configuration of the proposed model without the similarity component: DCM_Nosim.

4.1 Training and Hyper-parameters

We train our deep learning architecture on a training set using stochastic gradient descent (SGD) and tune parameters of the network on a development set. The word embeddings matrix **W** has dimension 50 and the width of convolution filters is 4. There are 100 convolutional feature maps, such that each intermediate vector obtained in the sentence model has also dimension 100. Batch size is set to 500 examples and the network is trained for 20 epochs with early stopping.

4.2 Results and Discussion

Table 2 reports the results of DCM and all the competing methods in the evaluation task. The experimental results are averaged with 10 random initializations. As we see, DCM achieves a much stronger performance than all the existing methods by a large margin, showing a significant improvement of about 5.3% gain on average for the accident and earthquake corpora.

Compared with HMM and Entity Grid, DCM requires no manual feature engineering anymore and can automatically learn better sentence representations using distributional word embeddings. Further, the abstract sentence representations computed by DCM are more meaningful in exactly capturing the relevant semantic, logical and syntactic features in coherent context than all the competing models.

Different from recursive neural network [11], which asks for expensive preprocessing using syntactic parsers to construct syntactic trees and then builds the convolution on them, CNN does not require any NLP parsers for preprocessing or external semantic resources.

The superior performance of DCM over DCM_Nosim demonstrates the necessity of the similarity computation in coherence assessment, while both recursive and recurrent neural networks [11] ignore this point and cannot achieve perfect results.

5 CONCLUSION

In this paper, we develop a deep coherence model, DCM, based on convolutional neural networks for text coherence assessment. The text coherence problem is investigated with a new perspective of learning sentence distributional representation and text coherence modeling simultaneously. In particular, DCM captures the interactions between sentences by computing the similarities of their distributional representations. Further, it can be easily trained in an end-to-end fashion. DCM is evaluated on a standard Sentence Ordering task. The experimental results demonstrate its effectiveness and promise in coherence assessment showing significant improvements over the state-of-the-art models by a wide margin.

ACKNOWLEDGMENTS

We thank all reviewers for their valuable comments. This work was supported by National Natural Science Foundation of China (NSFC No. 61672456), the Fundamental Research Funds for the Central Universities (No. 2017QNA5008, 2017FZA5007), and Zhejiang Provincial Engineering Center on Media Data Cloud Processing and Analysis.

REFERENCES

[1] Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-Based Approach. *Computational Linguistics* 34, 1 (2008), 1–34.

[2] Regina Barzilay and Lillian Lee. 2004. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *HLT-NAACL*. 113–120.

[3] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks* 5, 2 (1994), 157–166.

[4] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open Question Answering with Weakly Supervised Embedding Models. In *ECML PKDD*. 165–180.

[5] Micha Elsner and Eugene Charniak. 2008. Coreference-inspired Coherence Modeling. In *ACL*. 41–44.

[6] Micha Elsner and Eugene Charniak. 2011. Extending the Entity Grid with Entity-Specific Features. In *ACL-HLT*. 125–129.

[7] Pascale Fung and Grace Ngai. 2006. One story, one flow: Hidden Markov Story Models for multilingual multidocument summarization. *TSLP* 3, 2 (2006), 1–16.

[8] Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics* 21, 2 (1995), 203–225.

[9] Camille Guinaudeau and Michael Strube. 2013. Graph-based Local Coherence Modeling. In *ACL*. 93–103.

[10] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. 1746–1751.

[11] Jiwei Li and Eduard H. Hovy. 2014. A Model of Coherence Based on Distributed Sentence Representation. In *EMNLP*. 2039–2048.

[12] Annie Louis and Ani Nenkova. 2012. A Coherence Model Based on Syntactic Patterns. In *EMNLP-CoNLL*. 1157–1168.

[13] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*. 3771–3775.

[14] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*. 1045–1048.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.

[16] Massimo Poesio, Rosemary Stevenson, Barbara Di Eugenio, and Janet Hitzeman. 2004. Centering: A Parametric Theory and Its Instantiations. *Computational Linguistics* 30, 3 (2004), 309–363.

[17] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *ACM SIGIR*. 373–382.

[18] Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling Relational Information in Question-Answer Pairs with Convolutional Neural Networks. *CoRR* abs/1604.01178 (2016).