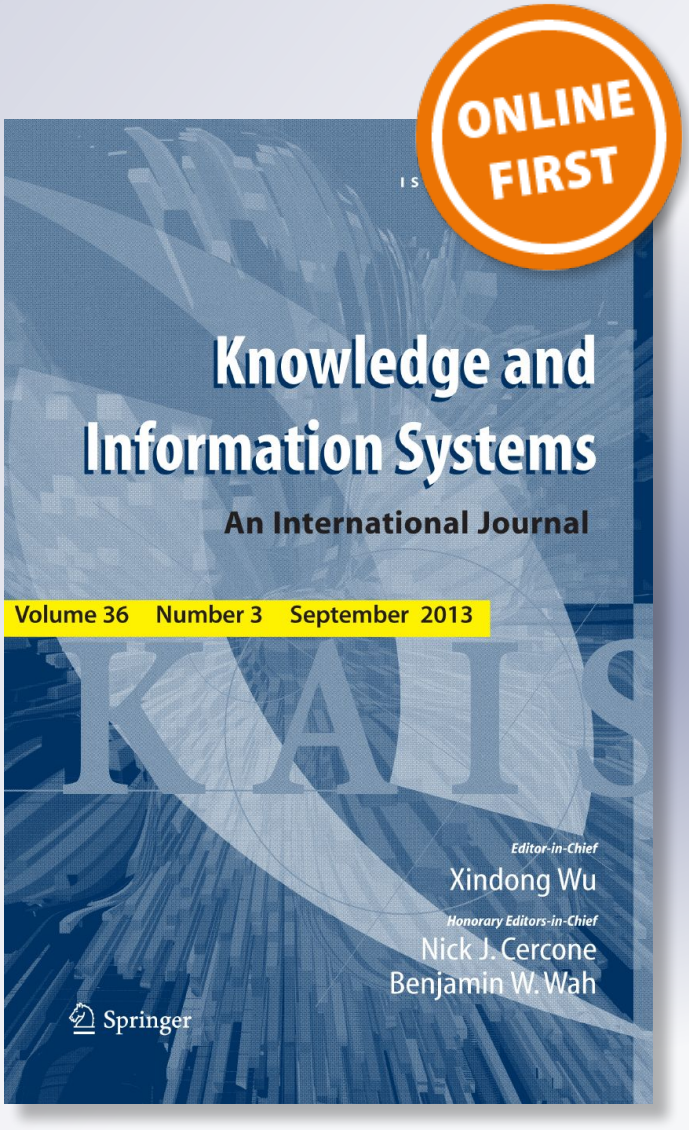
Improving short-text representation in convolutional networks by dependency parsing

Siheng Zhang, Wensheng Zhang & Jinghao Niu

Knowledge and Information Systems
An International Journal

ISSN 0219-1377

Knowl Inf Syst
DOI 10.1007/s10115-018-1312-9



Springer

Knowledge and Information Systems
https://doi.org/10.1007/s10115-018-1312-9

REGULAR PAPER

Improving short-text representation in convolutional networks by dependency parsing

Siheng Zhang[1] · **Wensheng Zhang**[1] · **Jinghao Niu**[1]

Received: 24 June 2017 / Revised: 23 April 2018 / Accepted: 28 November 2018

Abstract
Automatic question answering (QA) system is the inevitable trend of future search engines. As the essential steps of QA, question classification and text retrieval both require algorithms to capture the semantic information and syntactic structure of natural language. This paper proposes dependency-based convolutional networks to learn a representation of sentences. First, we use dependency layer to map discrete word depth on the dependency tree of a sentence into continuous real space. Then, the mapping result serves as weight of word vectors and convolutional kernels are employed as feature extractors for further specific tasks. The method proposed allows convolutional networks to take the advantage of higher representational ability of dependency structure. Experiments involving three tasks including text classification, duplicate classification and text pairs ranking confirm the advantages of our model.

Keywords Convolutional neural network · Dependency parsing · Question answering system · Question classification · Semantic equivalence

1 Introduction

Automatic question answering (QA) system has been studied for nearly half a century and achieves great success in both general and professional domains. For example, the Watson system, which is developed by IBM and won the first prize of a quiz show named 'Jeopardy!' [1], is playing an important role in diagnosis and treatment.[1] Generally speaking, a QA system contains three basic modules: preprocessing, text retrieval and answer generation

[1] http://www.watsonclinic.com/.

✉ Wensheng Zhang
zhangwenshengia@hotmail.com

Siheng Zhang
zhangsiheng2015@ia.ac.cn

Jinghao Niu
niujinghao2015@ia.ac.cn

[1] Institute of Automation, Chinese Academy of Sciences, School of Computer and Control Engineering, University of Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing 100190, China

Published online: 11 December 2018

[1–4]. More precisely, the preprocessing procedure includes subtasks like standardization, co-reference resolution, relation extraction and question classification [2,4].

Question classification is to predict the lexical answer type (LAT) of a question, which can not only narrow the search space but also guide the design of search template [4]. In order to develop QA system for general purpose, there exists some hierarchical taxonomies of LAT, which are summarized based on questions in general domain [5,6]. However, taxonomy needs further design in clinic domain. Based on questions collected from Dutch public health Web site, Boot et al. [7] found that International Classification of Primary Care (ICPC), which is proposed by World Health Organization, and Taxonomy of Generic Clinical Questions (TGCQ) [8] are far away from practical application.

Text retrieval is used in two parts of QA systems: One is extracting candidate answers from the knowledge database and the other is finding additional evidence from the literature to evaluate them [3]. Nowadays, with the rise of QA community, both candidates and evidence expand their boundaries by considering the data in a paired form as questions and answers from the Internet.

During developing a Chinese-speaking clinic QA system, we find that few questions can be solved by just using the knowledge database. So we adapt the framework of Watson from a self-contained system [1] to an open one. The system has access to the historical data from our QA community,[2] through which professional doctors are hired to answer questions from patients. Part of workflow is shown in Fig. 1. First, we analyze attributes including LAT of the query. Second, we recall the QA pairs in historical database if the questions have the same LAT. Third, do duplicate classification, i.e., to check whether the recalled questions are semantic equivalent to the query. As the duplicate questions have been solved, it is reasonable to suppose that the corresponding answers are good candidates.

In this paper, rather than the whole system, we focus on question classification and duplicate classification, both of which require algorithms to learn a good representation of natural language. Since semantic vector for single words have been widely studied [9–11], representation of natural language focuses on the compositionality in semantic vector spaces [12–16].

One of the most popular models is convolutional neural networks (CNN) [12]. Originally invented for computer vision (CV) tasks, CNN has been shown to be effective in natural language processing (NLP) tasks including text classification [18], semantic parsing [19] and et al. However, it has some shortcomings in NLP because location invariance and local compositionality of CNN [20] are inappropriate to natural language.

To overcome this problem, we propose dependency-based CNN, which allow CNN to leverage the syntactic structure of language. Syntactic parsing theory includes phrase structure grammar and dependency grammar [25]. This paper uses the latter. Our contributions fall into three aspects:

- Dependency layer is used to map discrete word depth of dependency grammar into continuous real space. And the mapping result can be learned during training so it is task specific.
- Dependency layer is integrated into convolutional networks to generate a better representation of short text, without any detailed (word-level or phrase-level) annotation.
- The model is adapted to different tasks including text classification, duplicate classification and text pairs ranking. Experiments validate its advantages and scalability.

The following sections are organized as: Sect. 2 introduces previous work related to language representation, deep learning methods as well as NLP tasks; in Sect. 3, we formulate

[2] http://www.120ask.com/.

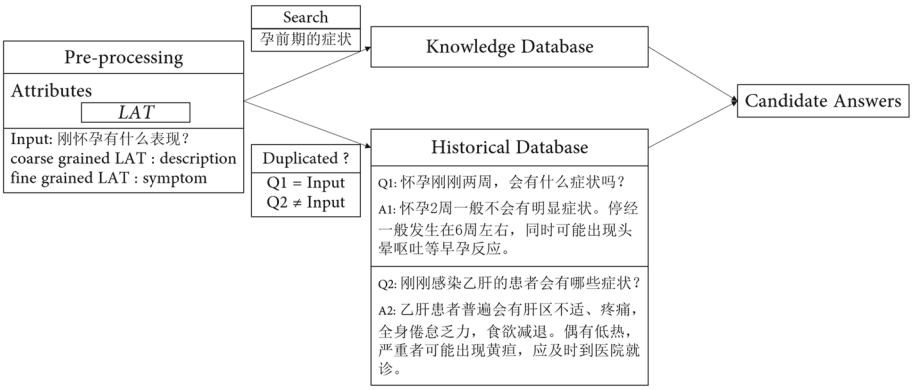


Fig. 1 Part of workflow of our clinic QA system. The system fetches questions with the same LAT in the database and examines whether they are semantic equivalent with input. As an example, Q1 is a paraphrase of input while Q2 is not, so A1 is considered as a candidate answer. The rest of the system is out of the scope of this paper so we neglect it

dependency-based CNN and show how it can be adapted for the three different tasks, i.e., text classification, duplicate classification and text pairs ranking; Sects. 4 and 5 report the experimental setup, results and discussions; Sect. 6 summarizes the work.

2 Related work

This section is organized in three aspects: 1. previous work on representation of language; 2. comparison of deep learning methods for NLP tasks and some background knowledge of syntactic parsing which inspires our work; 3. NLP tasks with regard to our work.

2.1 Representation of language

Tai et al. pointed out that the language representation models can be divided into three types: bag-of-words model (BoW), sequence model and tree-structured model [15]. Different representations require different learning algorithms when facing specific tasks. So before explaining why we choose to improve CNN, we discuss the differences among representation models.

BoW Sentence representation is generated by just summing up or taking average of word representation (e.g., one-hot feature) [4–6,21], hence it cannot distinguish different meanings caused by disorder of words. As an example, 'cat climbs tree' and 'tree climbs cat' have same representation under the perspective of BoW but are totally semantic different in fact. One of the most popular classifiers over features generated by BoW is Support Vector Machines (SVM) [5]. Additional features, like part-of-speech (POS) [4], further improved the performance.

Sequence Sentence representation is regarded as an order-sensitive function of the sequence of tokens [9–11]. Mikolov et al. proposed an efficient method to learn distributed word embeddings [11]. Taking embeddings of tokens (usually are words, but can be characters too) as input, CNN utilizes layers with filters in different sizes to aggregate local features into task-specific high-level features [18,22,23].

Tree Tree-structured model best captures the recursive and compositional property of language. Explicitly or implicitly using the properties promotes learning algorithms. Wen et al. used tricks like neglecting leaves on the syntactic tree [6]. Li et al. [17] and Zhang et al. [24] proposed SVM with tree kernel. Recursive neural tensor network (RNTN) computes vectors for a node from its children with a shared tensor-based compositional function [15]. Tai et al. developed tree-structured LSTM for both phrase structure grammar and dependency grammar [16].

2.2 Deep learning methods for NLP tasks

There are various deep neural networks designed for different NLP tasks, each with their own large amount of related work to which we cannot do full justice given space constraints. In this paper, we just give a brief comparison among the networks which inspired our work: recursive networks, recurrent networks and convolutional networks.

Recursive networks in which a neuron's state is computed by a compositional function over its children [13], or even the compositional function is composed of that of its children [14]. It fits the properties of language but is time- and memory-consuming because of a large amount of parameters. Work like RNTN [15] eased memory limitation but is still time-consuming.

Recurrent networks only allow strictly sequential information propagation. Work like tree-LSTM [16] solved this limitation by computing a neuron's state from an input vector and the hidden states of arbitrarily many child neurons.

Note that RNTN and tree-LSTM both require textual data to be in the form of phrase trees with manual label (i.e., word-level or phrase-level annotation) for each node. It will cost enormous human resource and so is not practical.

Convolutional networks extract high-level features from low-level features [26] and can be implemented in parallel. Although CNN has been widely used in NLP tasks [18,19,22,23], there exists a mismatch between CNN and NLP. This mismatch includes two aspects: *First*, location invariance, which makes sense in CV (pixels close to each other are likely to be semantically related), does not hold in NLP (it is important where in the sentence a word appears); *Second*, local compositionality is inadequate. As parts of phrase may be separated by several other words, high-level features cannot exactly capture their relationship.

In the next section, we will propose our model to ease this mismatch. Unlike tree-LSTM, which explicitly adapted recurrent networks into tree topologies, we turn to implicitly integrating tree structure information into convolutional networks.

To further explain where our inspiration is from, we introduce some basic theory of language. In the theory of language, syntactic parsing includes phrase structure grammar (a.k.a., constituency grammar) and dependency grammar [25]. The latter one, dependency grammar, views the sentence logic in terms of predicates and their arguments. A word becomes an appendage to another, and finally refers to the only predicate in the sentence. In this paper, we aim to develop CNN to be aware of dependency structure.

2.3 NLP tasks

In this paper, we focus on three tasks: text classification (including question classification), duplicate classification and text pairs ranking. The latter two tasks involve learning interactions between text pairs and can be solved using a same model. Hu et al. [22] and Severyn et al. [23] employed a fully connected network taking concatenated features as input. Features are

individually extracted from text by CNN. The latter work, text pairs matching convolutional neural network (TP-CNN), also concatenated similarity score as part of feature.

Similarity score is usually computed by noisy channel model, which suppose that there exists noise of literal words from a text to another. The idea of noisy channel scoring can trace back to the work of Echihabi et al. [27], in which a noisy channel model was used to explicitly map answer sentence's parse tree into question's. Bordes et al. [28] used vector multiplication to represent this noisy model, and TP-CNN further employed a matrix to allow more free and complex interactions between pair of text. In this paper, we follow what TP-CNN did.

3 Dependency-based convolutional networks

From the analysis above, we see that there is an urgent need to develop a convolutional model which can take advantage of syntactic structure. Consider two kinds of methodology, one is to explicitly use different convolution kernels for different syntactic compositional type, respectively, the other is to implicitly weight words based on their grammatical positions. The former will cause a large increase in time and space complexity because it violates parameter sharing, which is an important property of CNN. So we follow the second inspiration and propose *dependency-based CNN* (depCNN) to learn the representation of sentences.

3.1 CNN

Before moving on to the formulation of depCNN, it is necessary to review that of CNN [18]. Briefly speaking, a simple CNN contains one layer of convolution on top of word vectors. Suppose that word vectors are in d-dimensional space, i.e., $\mathbf{x}'_i \in \mathbf{R}^d$, then a sentence of length l (zero-padding if necessary) is concatenated by l word vectors:

$$\mathbf{x}' = [\mathbf{x}'_1; \mathbf{x}'_2; \ldots; \mathbf{x}'_l] \in \mathbf{R}^{l \times d} \tag{1}$$

For convenience, let $\mathbf{x}'_{i:j}$ refer to the concatenation of word vectors $\mathbf{x}'_i, \mathbf{x}'_{i+1}, \ldots, \mathbf{x}'_j$. A convolution operation applies a filter $\mathbf{w} \in \mathbf{R}^{h \times d}$ over a window of h words to extract a feature of local phrase region from ith to $(i + h - 1)$th word, i.e.,

$$f_i = g(\mathbf{w} * \mathbf{x}'_{i:i+h-1} + b) \tag{2}$$

in which $g(\cdot)$ is the activation function, $b \in \mathbf{R}$ is a bias term. In this paper, we set $g(\cdot)$ to be Rectified Linear Units(ReLU) [29], i.e., $g(x) = \max\{0, x\}$.

Sliding the kernel along the length of text with one word each step extracts features from all possible window of words. These features are aggregated to produce a feature map:

$$\mathbf{f} = [f_1; f_2; \ldots; f_{n-h+1}] \in \mathbf{R}^{n-h+1} \tag{3}$$

Perform max-pooling [30] over the feature map. Max-pooling supposes highest value to be the most important feature among the feature map. That is to say, take $\hat{f} = \max\{\mathbf{f}\}$ as the feature corresponding to this convolutional kernel. What's more, pooling mechanism solves variable length problem of textual data.

Up to now, we have described how to extract one feature from one kernel. To cover the rich semantic information of a sentence, CNN applies kernels of different window sizes, each with multiple filters, to extract multiple features. Denote the number of kernels to be m, then the final representation of a sentence is:

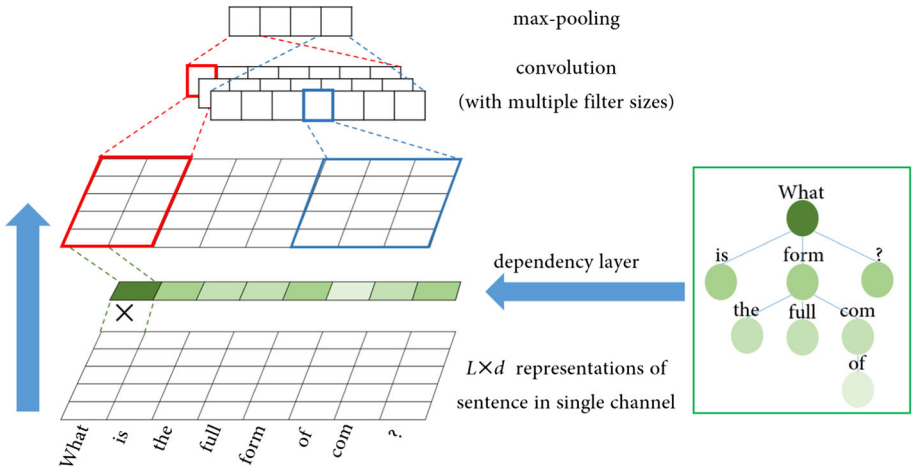


Fig. 2 Architecture of depCNN. On the dependency syntax tree, 'What' is the headword, so its depth is 1. Depth of other words can be calculated recursively. Here we neglect the fc-layer for output because the resulting feature can be used for different tasks but not just classification

$$\mathbf{u} = [\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_m] \tag{4}$$

3.2 depCNN

Unlike CNN directly taking concatenated word vectors as input, in depCNN, a fully connected layer (we call it *dependency layer*) is introduced to map the depth of words on the dependency syntax tree into continuous real space. Then, the mapping results are used to reweight the word vectors, which allows more compositional feature maps. Next, convolution operation over the weighted word vectors extracts semantic representation. Figure 2 shows the architecture of it.

Dependency layer maps word's depth into word's weight.

$$\begin{aligned} \boldsymbol{\lambda} &= sigmoid(\mathbf{W}_d\mathbf{d} + \mathbf{b}_d) \\ \mathbf{x}_i &= \lambda_i \mathbf{x}'_i \end{aligned} \tag{5}$$

in which $\mathbf{d} = [d_1, d_2, \ldots, d_l] \in \mathbf{R}^l$, with each entry representing the depth of a word on the dependency syntax tree. $\mathbf{b}_d \in \mathbf{R}^l$ is a bias term. $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_l] \in \mathbf{R}^l$, with each entry representing a word's weight. We use $sigmoid(\cdot)$ as activation function here to restrict the word's weights in the range of [0, 1].

Here we demonstrate that weighting word vectors is equivalent to use structure-specific kernels. Convolution over a window of h weighted words is:

$$\begin{aligned} \mathbf{w} * \mathbf{x}_{i:i+h-1} &= \sum_{j=i}^{i+h-1} \mathbf{w}_{[j,:]} \cdot \mathbf{x}_j^T \\ &= \sum_{j=i}^{i+h-1} (\lambda_j \mathbf{w}_{[j,:]}) \cdot \mathbf{x}_j'^T \end{aligned} \tag{6}$$

which means that a kernel now is structure-aware compared to that in simple CNN while preserving the computational advantage.

Also note that in our implementation, since sentence length is a hyper-parameter, zero-padding is also used here. Fortunately, this makes no effect on feature maps, because it is easy to cast the padding word's weight as zero by setting a specific value (such as -1) of padding depth. Since the reweighted padding word vectors are zeros, feature extracted by convolution operation over them will not be chosen to be an important feature by the max-pooling operation.

Consider the error flow propagating back to word vectors. Partial derivatives of the error signal occurring at a convolutional kernel with respect to a word vector is:

$$\frac{\partial f_i}{\partial \mathbf{x}'_j} = \lambda_j \frac{\partial f_i}{\partial \mathbf{x}_j} \tag{7}$$

As $\lambda_j \in [0, 1]$, depCNN suffers less disturbance on word vectors during training compared to CNN. Because word embeddings are pretrained on corpus of a large size [10,11] (always much larger than corpus for specific NLP tasks), the semantic relationship of words is more likely to make sense. So it is wise to be cautious when fine-tuning word vectors on corpus for specific natural language tasks.

After weighting word vectors, we extract features as what CNN does and get the representation of a sentence $\mathbf{u}$. Besides, we use regularization techniques. For text classification task, the representation is fed into a fully connected layer with dropout [31] to get final prediction:

$$\mathbf{z} = \text{softmax}(\mathbf{w}_{fc}(\mathbf{u} \odot \mathbf{r}) + \mathbf{b}_{fc}) \tag{8}$$

in which $\odot$ stands for elementwise multiplication. $\mathbf{b}_{fc}$ is a bias term. $\mathbf{r}$ is a vector with entries subject to Bernoulli distribution with an expectation $p \in [0.5, 1)$.

Loss function consists of cross-entropy and L_2-norm regularization, with a ratio l_2 controlling the trade-off between them. Denote the number of classes as C, the true label of ith sample as a one-hot vector $\mathbf{y}_i \in \mathbf{R}^C$, while the probability of each class predicted by depCNN as $\mathbf{z}_i \in \mathbf{R}^C$. Aggregating cross-entropy loss of all samples in dataset D gives the loss function:

$$\mathcal{L}(\mathbf{y}, \mathbf{z}) = -\frac{1}{|D|} \sum_{D} \sum_{i=1}^{C} \mathbf{y}_i \ln \mathbf{z}_i + l_2 \|\theta\|_2 \tag{9}$$

in which $\theta = \{\mathbf{W}_d, \mathbf{b}_d, \mathbf{W}_{fc}, \mathbf{b}_{fc}\}$. We constrain their norms to avoid over-fitting.

3.3 Extend to text pairs

For tasks involving text pairs, the model can be extended to capture the interactions between them. Following the framework proposed by Severyn et al. [23], we replace CNN by our proposed model depCNN, see Fig. 3.

Given the feature that extracted by depCNN from each text individually, a similarity score can be calculated as:

$$x_{sim} = \mathbf{u}_1 \mathbf{M} \mathbf{u}_2 \tag{10}$$

The score measures the similarity of text pair by transforming one side of text into the semantic space of another [23], which has been widely used as a scoring model in information retrieval and question answering.

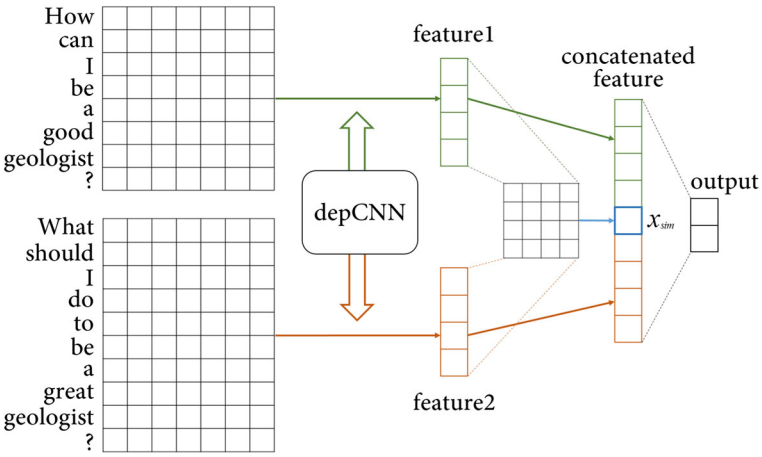


Fig. 3 Text pairs modeling using depCNN. If the output is in binary code, i.e., 0 or 1, it can be used for duplicate classification. If the output is in the form of probability, it can be used for text pairs ranking

Finally, features concatenated by score and text features are fed into a fully connected network for final prediction:

$$\mathbf{y} = \text{softmax}(\mathbf{w}_{fc}[\mathbf{u}_1, x_{sim}, \mathbf{u}_2] + \mathbf{b}_{fc}) \tag{11}$$

And the loss function is the same with that of text classification.

Note that neural network can output final prediction in form of binary code or probability. For duplicate classification, binary code is adopted, while for text pairs ranking, probability is adopted to determine the rank.

Following what Kim [18] did, our model varies with the different initialization ways of word embeddings: (1) *rand*: word vectors are randomly initialized and can be fine-tuned during training; (2) *static*: word vectors are initialized with pretrained word embeddings and keep static during training; (3) *non-static*: same to (2) except that word vectors can be fine-tuned; (4) *multi*: combine (2) and (3), which can be viewed as a single channel, to form a multi-channel model.

4 Datasets and experimental setup

Experiments of this paper involve three tasks: text classification (*TREC*, *MR*, *CR*, *MPQA*, *SST2/SST5* and *Clinic-1*), duplicate classification (*Quora* and *Clinic-2*) and text pairs ranking (*TREC-QA*). All experiments are carried out on computer with single CPU (Intel Xeon E5-2683 v3) and single GPU (GTX TITAN X).

Here is a summary of the datasets. Note that because LAT taxonomy in clinic domain is not practical [7], we test our model on *Clinic-1* for triage. Besides, question pairs in *Clinic-2* may not have the same LAT.

1. *TREC* Task involves classifying questions into 6 LATs (including person, location and so on). 5452 samples are for training, while 500 for test.
2. *MR* 10,662 movie reviews which are positive or negative.

Table 1 Summary of *Clinic-1* datasets

Mental health	Chirurgery	Internal	Paediatrics
7943	19,300	13,160	7406
Chinese medicine	Obstetrics	E.N.T	Nutrition
19,466	56,554	27,256	8915

Table 2 summary of *TREC-QA* datasets

	Questions	QA pairs	Correct (%)
TRAIN-ALL	1229	53,417	12.0
TRAIN	94	4718	7.4
DEV	82	1148	19.3
TEST	100	1517	18.7

3. *CR* 4225 customer reviews of 17 products (cameras, MP3s, etc), which are positive or negative. This series of 3 datasets consists of 5, 9 and 3 products individually [32–34].[3]
4. *MPQA* Opinion polarity detection subtask of the MPQA dataset with 15400 samples.[4] Task is to predict the objectiveness/subjectiveness of an opinion [35].
5. *SST2/SST5* Stanford Sentiment Treebank (SST5) is an extension of MR and with train/dev/test splits and 5 fine-grained labels (very positive, positive, neutral, negative, very negative), relabeled by Socher et al [15]. SST2 is the same as SST5 but with neutral reviews removed and binary labels.[5]
6. *Clinic-1* We collect a total of 160,000 queries from the Web site[6] to study medical triage, i.e., advice for patients to corresponding medical departments. Class distribution is shown in Table 1).
7. *Quora* 404,349 question pairs from Quara, one of the biggest online question answering community. 149,306 are duplicated and 255,043 are not.
8. *TREC-QA* A set of factoid questions with candidate answers is collected from TREC-QA tracks 8–13. Manual judgement of candidate answer sequences is provided for the entire TREC 13 set and for the first 100 questions from TREC 8-12. An additional training set TRAIN-ALL contains 1299 questions from the entire TREC 8-12 collection and comes with automatic judgements, so it is more informative but noisy (Table 2).
9. *Clinic-2* Task involves judging whether the text retrieved by search engine are semantic equivalent to the original query. The data are collected from the Web site and labeled by five medical bachelors. When opinions conflict, the majority is adopted. There are 115 queries, each with several candidates, forming 986 pairs (357 positive and 629 negative). Among them, 96 of 115 queries have at least one semantic equivalent candidate, while the rest does not have any.

[3] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets.

[4] http://mpqa.cs.pitt.edu/corpora/mpqa_corpus/.

[5] http://nlp.stanford.edu/sentiment/ Data are actually provided with phrase-level annotation; however, to enable direct comparison in learning representation of sentences, we do not use phrase-level annotation here.

[6] http://www.120ask.com/.

4.1 Train/dev/test split

Because the datasets have different splits, we perform different operations on them: 1. for datasets with train/dev/test split (*TREC-QA, SST2/SST5*), we do nothing; 2. for datasets with train/test split but no development set (*TREC*), 10% of the training data are randomly selected as the development set; 3. for datasets without any split (the other 6 datasets), tenfold cross-validation is carried out.

For the former two situations, models with highest performance on development set during training will then be applied on the test set.

4.2 Hyper-parameters

For all datasets except *TREC-QA*, we use: filter windows (h) of size 3, 4, 5 with 100 feature maps each size; dropout rate $p = 0.5$; l_2 constraint of 0.01; word embedding dimension of 300; sentence length of 50 (neglect the excess part and perform zero-padding when necessary). According to the filter sizes and numbers, the similarity matrix is of size 300×300. For the training process, we use 30 epochs with mini-batch size of 50. Adam algorithm, which can dynamically adjust the learning rate of each parameter based on the first- and second-order moment of gradient [36], over shuffled mini-batches is adopted to optimize the models.

As for *TREC-QA*, to enable direct comparison with TP-CNN [13], we use almost the same parameters: questions' length of 33 and answers' length of 60, word embedding dimension of 50, training epochs of 50, mini-batch size of 50. Early-stopping strategy is activated once there is no more improvement for 3 epochs. Training is done through mini-batch stochastic gradient descent (SGD). The only difference is that, we use filter windows of size 4 and 5, with a number of 50 each size on TRAIN and 100 on TRAIN−ALL, so the similarity matrix is of size 100×100 and 200×200, respectively. We will show that TP-CNN with these parameters achieves higher performance.

4.3 Pretrained word vectors

Initializing word vectors with those obtained from an unsupervised neural language model is popular especially in the absence of a large supervised training set. Before pretraining word embeddings, we use CoreNLP tool [32] in English and HanLP tool[7] in Chinese for parsing.

For English, we use *GloVe*[8] published by Stanford [31]. We choose the one with smallest vocabulary size of 400 thousand that were trained on Wikipedia 2014[9] and Gigaword 5.[10] The vectors have dimensionality of 300. To enable direct comparison, we use word embeddings published by Severyn [13] for *TREC-QA*.

For Chinese, we run the tool *word2vec* [22] on *Clinic-1*, which contains 81870 words (including many medical terminologies). Hundred-dimensional word embeddings are obtained by training the skip-gram model with window size 5 and filtering out words with frequency less than 5.

Lastly, words not present in pretrained embeddings are initialized from a uniform distribution $\mathcal{U}[-0.25, 0.25]$.

[7] http://hanlp.linrunsoft.com/.

[8] https://nlp.stanford.edu/projects/glove/.

[9] https://dumps.wikimedia.org/enwiki/20140102/.

[10] https://catalog.ldc.upenn.edu/LDC2011T07.

Table 3 Ablation study on *TREC*

acc (%)	With	Without
rand	$92.2_{32.5s}$	$91.2(91.2)_{32.3s}$
static	$\mathbf{95.0}_{34.3s}$	$93.6(92.8)_{33.1s}$
non-static	$94.2_{33.4s}$	$\mathbf{93.6}(93.6)_{33.3s}$
multi	$94.6_{61.1s}$	$93.4(92.2)_{59.6s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.005$. With accuracy from Kim [18] in brackets, and runtime as subscripts

5 Results and analysis

This section is organized by three tasks. For each task, both ablation study and comparison study are carried out on all datasets. And for the datasets which need tenfold cross-validation, we report the average with the standard deviation.

5.1 Text classification

5.1.1 Baselines and metrics

We compare the accuracy of depCNN to: (1) SVM-1: linear kernel with BoW model on unigram [4]; (2) SVM-2: linear kernel with BoW and POS [5]; (3) SVM_s: SVM with uni-bi-trigrams, wh word, headword, POS, parser, hypernyms and 60 hand-coded rules as features [21]; (4) PV: linear kernel SVM on top of paragraph vectors [37]. We trained 50 epochs to get 300-dimensional paragraph vectors; (5) RNN: recurrent neural networks on word vectors; (6) LSTM: long short-term memory [38] on word vectors; (7) GRU: gated recurrent networks [39] on word vectors.

5.1.2 Results

Ablation study on the four variants of depCNN confirms the effectiveness of dependency layer, no matter what initialization method of word vectors is (Tables 3, 4, 5, 6, 7, 8, 9). Note that without dependency layer, depCNN has the same structure with Kim's work [18], but achieves comparable or higher accuracy,[11] which demonstrates the fairness of the comparison.

To examine the results by statistical significance, assuming a null hypothesis that depCNN is not better than CNN, we calculate *p-value* using *t test*, and check whether it is below the thresholds (reported in the table's captions). If so, the null hypothesis is rejected, and the alternative hypothesis is favored, i.e., depCNN outperforms CNN. In addition, the average training time is reported as subscripts, from which we can see that adding dependency layer does not lead to a significant increase in computational cost.

Compared with other algorithms in Table 10, we can see that depCNN achieves the best performance among all algorithms, with always highest accuracy and smallest deviation. Note that SVM_s used manual feature designed for *TREC* [21], so cannot be extended to other

[11] More details: 1. For *CR*, dataset used by Kim [18] is a subset of ours; 2. For *SST2/SST5*, Kim [18] and other state-of-the-art obtained a higher performance by using phrase-level annotation, which is out of our scope.

Table 4 Ablation study on *MR*

acc (%)	With	Without
rand	$76.767 \pm 0.246_{52.4s}$	$75.579 \pm 1.776(76.1)_{56.4s}$
static	$81.145 \pm 0.334_{51.2s}$	$80.201 \pm 0.767(81.0)_{51.3s}$
non-static	$81.332 \pm 0.923_{53.5s}$	$\mathbf{80.957 \pm 0.938}(81.5)_{54.1s}$
multi	$\mathbf{81.520 \pm 0.778}_{99.1s}$	$80.769 \pm 0.886_{101.3s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.01$. With accuracy from Kim [18] in brackets, and runtime as subscripts

Table 5 Ablation study on *CR*

acc (%)	With	Without
rand	$80.040 \pm 1.776_{26.4s}$	$78.787 \pm 3.313(79.8)_{26.2s}$
static	$\mathbf{83.129 \pm 1.075}_{28.5s}$	$\mathbf{81.443 \pm 3.129}(84.7)_{26.3s}$
non-static	$82.254 \pm 2.580_{26.9s}$	$80.551 \pm 3.136(84.3)_{25.4s}$
multi	$82.593 \pm 0.778_{50.9s}$	$81.013 \pm 3.304(85.0)_{46.3s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.001$. With accuracy from Kim [18] in brackets, and runtime as subscripts

Table 6 Ablation study on *MPQA*

acc (%)	With	Without
rand	$73.611 \pm 4.37_{182.1s}$	$73.236 \pm 4.679_{177.9s}$
static	$80.659 \pm 4.156_{183.1s}$	$\mathbf{79.861 \pm 4.372}_{185.3s}$
non-static	$\mathbf{80.732 \pm 3.867}_{183.9s}$	$79.536 \pm 4.819_{178.6s}$
multi	$79.895 \pm 5.298_{361.3s}$	$79.611 \pm 5.873_{353.3s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.05$. With runtime as subscripts

Table 7 Ablation study on *SST2*

acc (%)	With	Without
rand	$77.924_{55.2s}$	$77.320_{57.3s}$
static	$83.361_{59.4s}$	$82.922_{58.6s}$
non-static	$\mathbf{83.471}_{62.1s}$	$\mathbf{83.031}_{59.0s}$
multi	$82.812_{117.4s}$	$82.482_{123.2s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.01$. With runtime as subscripts

datasets. And POS tagger works poorly for Chinese oral text so SVM-2 is not carried out on *Clinic-1*.

From Table 10, we also find out some results in depth: *First*, on *TREC*, SVM_s achieves the best performance as depCNN does. To understand the performance of SVM_s, let us trace back to previous work on question classification, which showed that some types of

Table 8 Ablation study on *SST5*

acc (%)	With	Without
rand	$43.344_{48.5s}$	$42.172_{54.8s}$
static	$44.253_{53.4s}$	$43.213_{51.0s}$
non-static	$44.706_{50.5s}$	$43.710_{51.5s}$
multi	$\mathbf{44.977}_{126.4s}$	$\mathbf{44.434}_{124.0s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.01$. With runtime as subscripts

Table 9 Ablation study on *Clinic-1*

acc (%)	With	Without
rand	$79.793 \pm 0.624_{910.2s}$	$79.377 \pm 0.330_{880.2s}$
static	$79.980 \pm 0.170_{909.1s}$	$\mathbf{79.894 \pm 0.146}_{974.5s}$
non-static	$\mathbf{80.118 \pm 0.117}_{898.0s}$	$79.757 \pm 0.378_{909.7s}$
multi	$79.982 \pm 0.400_{1678.2s}$	$78.852 \pm 0.898_{1665.4s}$

Bold values indicate the best initialization way for each model
t test: $p < 0.05$. With runtime as subscripts

Table 10 Comparison study on text classification

acc (%)	TREC	MR	CR	MPQA	SST2	SST5	Clinic-1
depCNN	**95.0**	$\mathbf{81.520}_{\pm 0.778}$	$\mathbf{83.129}_{\pm 1.075}$	$\mathbf{80.732}_{\pm 3.867}$	**83.471**	**44.977**	$\mathbf{80.118}_{\pm 0.117}$
SVM-1 [4]	86.4	$74.210_{\pm 1.726}$	$75.356_{\pm 4.137}$	$68.903_{\pm 7.949}$	59.967	30.860	$69.540_{\pm 1.530}$
SVM-2 [5]	89.6	$74.451_{\pm 2.014}$	$75.567_{\pm 4.215}$	$69.154_{\pm 7.627}$	61.395	29.683	–
SVM_S [21]	**95.0**	–	–	–	–	–	–
CNN [18]	93.6	81.5	$81.443_{\pm 3.129}$	$79.861_{\pm 4.372}$	83.031	44.434	$79.894_{\pm 0.146}$
PV [37]	42.3	$56.640_{\pm 1.377}$	$71.308_{\pm 3.622}$	$65.885_{\pm 5.096}$	75.409	41.023	$62.442_{\pm 0.756}$
RNN	73.8	$70.457_{\pm 1.782}$	$71.875_{\pm 3.500}$	$69.489_{\pm 6.648}$	77.056	33.136	$76.646_{\pm 0.598}$
LSTM [38]	90.0	$72.819_{\pm 1.377}$	$76.813_{\pm 3.117}$	$72.259_{\pm 6.488}$	78.778	42.864	$78.440_{\pm 0.262}$
GRU [39]	92.8	$72.943_{\pm 1.301}$	$75.188_{\pm 4.126}$	$71.563_{\pm 6.400}$	78.556	42.000	$77.751_{\pm 0.314}$

Bold values indicate the best model for each data set
For datasets without any split, we report avg. ± std. of tenfold CV; otherwise, we report accuracy on test set
SVM_S uses manual feature designed just for TREC;
POS tagger works poorly for Chinese oral text so we do not run SVM-2 on Clinic-1

words are highly correlated with its LAT, such as 'headword' (see table 7 in Loni's work [4]). Utilizing these syntactic knowledge, SVM_S [21] over features including 60 hand-coded rules achieved the best performance. DepCNN also achieves this performance, showing the great representation learning ability of it. *Second*, with corpus size growing, the difference of depCNN and CNN becomes smaller. DepCNN outperforms CNN with 1.4% on *TREC*. However, the difference becomes smaller on the larger datasets (*MR, SST2/SST5*), and for *Clinic-1*, the largest one, the difference reduces to 0.22%. We suggest that it is because the larger corpus can supervise more refined convolutional kernels, while with smaller corpus, introducing dependency layer allows convolutional kernels to capture the intrinsic structure of textual data, so can show a significant improvement.

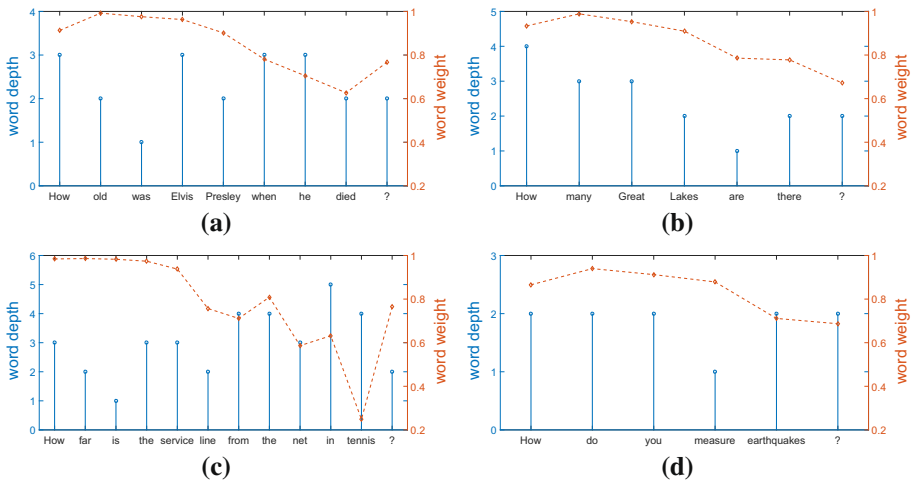


Fig. 4 Examples of the word weights on *TREC*

5.1.3 Case study

'Feeding' a text into depCNN, we can fetch the words' weights and final predictions during forward passing procedure. Cases are chosen to validate that depCNN can learn a task-specific weighting mechanism.

In *TREC* (Fig. 4), we choose the questions with the same leading word 'How' but with different LATs. We find out that: 1, words next to 'How' [i.e., 'old' (Fig. 4a), 'many' (Fig. 4b), 'far' (Fig. 4c) and 'do' (Fig. 4d)] are most important because each of them forms an independent semantic together with the interrogative pronoun 'How'; 2, objects that are consulted [i.e., 'Elvis Presley' (Fig. 4a), 'Great Lakes' (Fig. 4b), 'service line' (Fig. 4c) and 'measure' (Fig. 4d)] are also important; 3, subordinate clause or modifier (i.e., 'when he died' (Fig. 4a) and 'from the net in tennis' (Fig. 4c)) are of least importance because they have few impact on LAT.

In *MR* (Fig. 5), we choose two positive and two negative examples. We can conclude that: words with a strong emotional inclination [i.e., 'idiots' (Fig. 5a), 'involving' (Fig. 5b), 'warm' and 'realistic' (Fig. 5c), 'beautifully', 'subtly' and 'fine' (Fig. 5d)] are of higher importance. Moreover, in Fig. 5d, the model captures the progressive relation between two phrases ('is beautifully mounted' and 'managing to walk a fine line') so assign a higher weight for the latter.

What's more, dependency parsing is not always correct, especially for oral text. For example, the headword of 'video games are more involving than this mess.' is 'are' but not 'involving', and proper names (i.e., 'St. Louis' and 'Elvis Presley') should be in the same layer on syntax tree. In spite of parsing error, experiment results confirm that depCNN shows robustness for parsing noise to some extent.

5.2 Duplicate classification

5.2.1 Baselines and metrics

We compare depCNN to: (1) Word-Cnt: Logistic regression on the number of repeated words between a pair of text; (2) PV: Logistic regression on the cosine distance of paragraph vectors

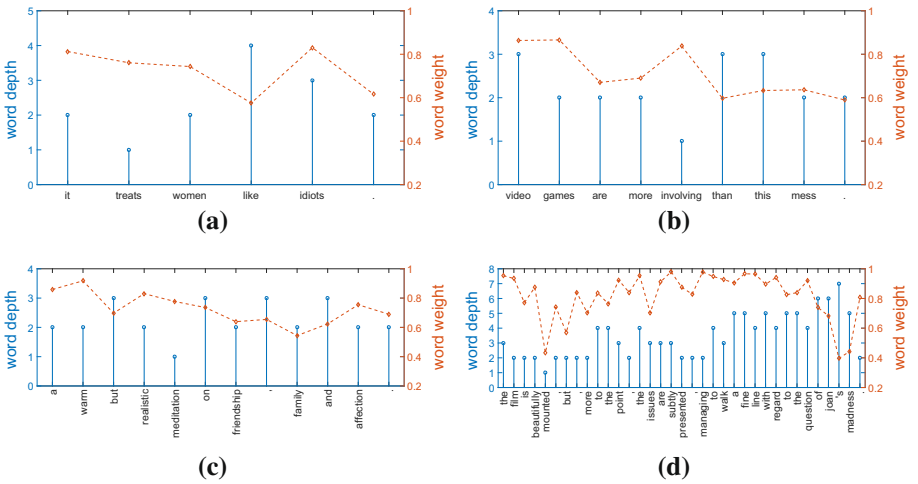


Fig. 5 Examples of the word weights on *MR*

Table 11 Ablation study on *Quora*

	With		Without	
	acc (%)	F1 (%)	acc (%)	F1 (%)
rand	79.81 ± 0.128	73.22 ± 0.246	79.47 ± 0.250	$\mathbf{73.09 \pm 0.221}$
static	79.72 ± 0.241	$\mathbf{73.57 \pm 0.121}$	79.32 ± 0.111	72.57 ± 0.474
non-static	79.41 ± 0.241	73.17 ± 0.621	79.15 ± 0.206	73.03 ± 0.190
multi	$\mathbf{80.42 \pm 0.220}$	73.02 ± 0.619	$\mathbf{80.07 \pm 0.259}$	72.98 ± 0.306

Bold values indicate the best initialization way for each model
Runtime for multi/others: 2.0 h/50 min

[28]. We trained 50 epochs to get 300-dimensional paragraph vectors; (3) TP-CNN: text pairs matching convolutional neural network which is the state-of-the-art method [23]. Note that without dependency layer, our model has the same structure as TP-CNN, but here we do not use additional features in the output layer.

Note that the class distribution is imbalanced. More precisely speaking, equivalent text pairs are significantly less than inequivalent pairs. So in this part, not only accuracy but also $F1$-score is used to measure the models.

When developing clinic QA system, we also concern about two metrics on *Clinic-2*: (1) *QTP*(query true positive rate): ratio of queries with at least one paraphrase that are correctly solved; (2) *QTN*(query true negative rate): ratio of queries with no paraphrases that are correctly detected. A system can reply more queries with a higher *QTP*, and make fewer mistakes with a higher *QTN*.

5.2.2 Results

Ablation study on *Quora* (Table 11) shows that dependency layer helps improve the performance for all initialization ways. Note that there is no significant difference in training time between models with/without dependency layer.

Table 12 Ablation study on *Clinic-2*

	With				Without			
	acc (%)	F1 (%)	QTP (%)	QTN (%)	acc (%)	F1 (%)	QTP (%)	QTN (%)
rand	72.820$_{\pm 4.290}$	63.953$_{\pm 5.640}$	70.730$_{\pm 6.794}$	**77.089**$_{\pm 4.219}$	72.410$_{\pm 1.780}$	59.347$_{\pm 7.308}$	63.696$_{\pm 11.595}$	77.212$_{\pm 5.644}$
static	**75.255**$_{\pm 1.301}$	**68.015**$_{\pm 2.370}$	74.566$_{\pm 8.885}$	76.181$_{\pm 6.020}$	**74.936**$_{\pm 4.201}$	**63.848**$_{\pm 5.671}$	66.135$_{\pm 6.122}$	**79.657**$_{\pm 7.168}$
non-static	73.119$_{\pm 4.271}$	65.567$_{\pm 4.995}$	**75.713**$_{\pm 2.765}$	71.515$_{\pm 5.837}$	73.027$_{\pm 3.476}$	63.388$_{\pm 4.525}$	**68.431**$_{\pm 3.720}$	73.841$_{\pm 6.912}$
multi	69.977$_{\pm 3.968}$	61.564$_{\pm 3.502}$	71.510$_{\pm 5.725}$	69.096$_{\pm 7.126}$	69.355$_{\pm 5.421}$	53.129$_{\pm 15.601}$	58.241$_{\pm 17.778}$	80.601$_{\pm 8.908}$

Bold values indicate the best initialization way for each model
Runtime for multi/others: 200 s/120 s

Table 13 Comparison study on duplicate classification

	Quora		Clinic-2			
	acc (%)	F1 (%)	acc (%)	F1 (%)	QTP (%)	QTN (%)
depCNN	**80.42**$_{\pm 0.220}$	**73.02**$_{\pm 0.619}$	**75.255**$_{\pm 1.301}$	**68.015**$_{\pm 2.370}$	**74.566**$_{\pm 8.885}$	76.181$_{\pm 6.020}$
Word-Cnt	61.158$_{\pm 0.153}$	46.441$_{\pm 0.223}$	51.023$_{\pm 12.178}$	31.955$_{\pm 12.435}$	51.414$_{\pm 12.823}$	49.490$_{\pm 11.187}$
PV	67.468$_{\pm 0.359}$	51.874$_{\pm 5.384}$	52.931$_{\pm 4.014}$	47.185$_{\pm 3.976}$	69.389$_{\pm 4.859}$	44.127$_{\pm 4.722}$
TP-CNN	80.07$_{\pm 0.259}$	72.98$_{\pm 0.306}$	74.936$_{\pm 4.201}$	63.848$_{\pm 5.671}$	66.135$_{\pm 6.122}$	**79.657**$_{\pm 7.168}$

Bold values indicate the best model for each data set

Table 14 Ablation study on *TREC-QA* TRAIN

	With		Without	
	MAP	MRR	MAP	MRR
rand	0.7043	0.7652	0.6907	0.7516
static	**0.7502**	**0.8094**	0.7325(0.7329)	**0.8018(0.7962)**
non-static	0.7218	0.8063	0.7180	0.7545
multi	0.7345	0.7972	**0.7349**	0.7749

Bold values indicate the best initialization way for each model
Runtime for multi/others: 310 s/170 s

Table 15 Ablation study on *TREC-QA* TRAIN-ALL

	With		Without	
	MAP	MRR	MAP	MRR
rand	0.7447	0.7936	0.7389	0.7948
static	**0.7669**	**0.8215**	**0.7654**(0.7459)	**0.8186**(0.8078)
non-static	0.7624	0.8156	0.7588	0.8096
multi	0.7507	0.8108	0.7518	0.7993

Bold values indicate the best initialization way for each model
Runtime for multi/others: 50 min/25 min

Ablation study on *Clinic-2* (Table 12) has a bit difference. QTN falls down when using dependency layer, however, improvements are more significant on other metrics. In detail, the largest decline proportion of QTN is -13.95%, occurring in multi-channel (69.096% v.s. 80.601%). But in that channel, accuracy, $F1$-score and QTP improves with a ratio of 0.90% (69.977% v.s. 69.355%), 15.88% (61.564% v.s. 53.129%) and 22.78% (71.510% v.s. 58.241%), respectively. Moreover, smaller variance demonstrates that dependency layer helps the model to be more robust. And also the training time of them is nearly the same.

It can be found that among all algorithms (Table 13), depCNN achieves the best performance.

5.3 Text pairs ranking

5.3.1 Baselines and metrics

Again, we compare depCNN to: (1) Word-Cnt, (2) PV and (3) TP-CNN [23], note that we use the same additional features. And because we use more types of filter sizes, our reimplementation also exceeds TP-CNN (Tables 14, 15).

Mean Average Precision (MAP) and Mean Reciprocal Rank(MRR), which are common in information retrieval and question answering task, are used to evaluate the quality of depCNN in this part.

MRR is computed as: $\text{MRR} = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{rank(q)}$, where $rank(q)$ is the position of the first correct answer in the candidate list. Hence, MRR is suitable for cases where each question has only a single correct answer. Differently, MAP pays more attention to the ranks of all the correct answers. MAP is computed as the mean over the average precision scores for each

Table 16 Comparison study on text pairs ranking

	TRAIN		*TRAIN-ALL*	
	MAP	MRR	MAP	MRR
depCNN	**0.7502**	**0.8094**	**0.7669**	**0.8215**
Word-Cnt	0.5820	0.6421	0.5820	0.6421
PV	0.5272	0.5986	0.5629	0.6441
TP-CNN	0.7329	0.7962	0.7459	0.8078

Bold values indicate the best model for each data set

query, i.e., $\text{MAP} = \frac{1}{|Q|}\sum_{q=1}^{|Q|} AvgP(q)$. To ensure direct comparison, the official scorer *trec_eval* is used here.

5.3.2 Results

Beyond the static way used by Severyn [13], we also perform ablation study using the other initializations of word embeddings (Tables 14, 15).

In almost all variants except multi-channel, dependency layer improves both MAP and MRR. Although MAP in multi-channel is a bit lower with dependency layer (0.7345 v.s. 0.7349 on TRAIN, and 0.7507 v.s. 0.7518 on TRAIN-ALL), MRR in that channel is significantly higher (0.7972 v.s. 0.7749 on TRAIN, and 0.8108 v.s. 0.7993 on TRAIN-ALL). Among the comparison algorithms (Table 16), our model also achieves the highest performance.

At the end of the experimental part, we would like to state our discovery of how to determine the initialization way of word embeddings. *First*, static way is better than non-static and multi way if the dataset is of small size, such as *TREC*, *CR*, *Clinic-2* and *TREC-QA* datasets (Tables 3, 5, 12, 14, 15). On the contrary, non-static or multi way will outperform static way (Tables 4, 6, 7, 8, 9, 11). We suggest that small datasets are not sufficient to learn a good word embeddings because disturbance on vectors of words that appear in the dataset will break the semantic relationship among words. *Second*, *multi* way is more complicated but with no striking improvements, we suggest not to use this way.

6 Summary

We propose dependency-based CNN (depCNN) which hires dependency layer to improve CNN for NLP tasks. Dependency layer is used to map words' depth on the dependency tree into word weights. Then CNN extracts feature from weighted word vectors for further purposes. DepCNN overcomes the shortcoming that CNN cannot fully capture the syntactic information inside a sentence, so can learn a better representation. Besides, we extend depCNN to learn the interactions between text pairs. Experiments on three tasks including text classification, duplicate classification and text pairs ranking demonstrate the effectiveness and scalability of proposed method. Though relying on dependency parsing, depCNN shows some extent of robustness when dependency parsing introduces some noise.

Acknowledgements The authors are thankful for the financial support from the National Natural Science Foundation of China (U1636220, 61432008, 61472423), the Huawei Innovation Research Program (HO2017050001BI), and the Beijing Natural Science Foundation (4172063).

References

1. Ferrucci DA (2012) Introduction to "This is Watson". IBM J Res Dev 56(3/4):1:1–1:15
2. Lally A, Prager JM, McCord MC et al (2012) Question analysis: how Watson reads a clue. IBM J Res Dev 56(3/4):2:1–2:14
3. Chu-Carroll J, Fan J, Schlaefer N, Zadrozny W (2012) Textual resource acquisition and engineering. IBM J Res Dev 56(3/4):3:1–3:11
4. Loni B (2011) A survey of state-of-the-art methods on question classification. Delft University of Technology, Tech. Rep: 1–40
5. Li X, Roth D (2002) Learning question classifiers. In: Proceedings of ACL, pp 1–7
6. Wen X, Zhang Y, Liu T et al (2006) Syntactic structure parsing based Chinese question classification. J Chin Inf Process 20(2):33–39
7. Boot C, Meijman FJ (2010) Classifying health questions asked by the public using the ICPC-2 classification and a taxonomy of generic clinical questions: an empirical exploration of the feasibility. Health Commun 25(2):175–181
8. Ely JW, Osheroff JA, Gorman PN et al (2000) A taxonomy of generic clinical questions: classification study. Br Med J 321(7258):429–432
9. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. Mach Learn Res 3:137–1155
10. Mikolov T, Karafiat M, Burget L et al (2010) Recurrent neural network based language model. In: Proceedings of Interspeech, pp 1045–1048
11. Mikolov T, Chen K, Corrado GS et al (2013) Efficient estimation of word representations in vector space. In: Proceedings of ICLR
12. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. In: Proceedings of ACL, pp 655–665
13. Socher R, Lin C, Manning CD et al (2011) Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of ICML, pp 129–136
14. Socher R, Huval B, Manning CD et al (2012) Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of EMNLP, pp: 1201–1211
15. Socher R, Perelygin A., Wu JY et al (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of EMNLP, pp 1631–1642
16. Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of ACL, pp 1556–1566
17. Li X, Roth D (2004) Learning question classifiers: The role of semantic information. In: Proceedings of COLING, pp 556–562
18. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of EMNLP, pp 1746–1751
19. Yih W, He X, Meek C (2014) Semantic parsing for single-relation question answering. In: Proceedings of ACL, pp 643–648
20. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, pp 2278–2324
21. Silva J, Coheur L, Mendes A, Wichert A (2011) From symbolic to sub-symbolic information in question classification. Artif Intell Rev 35(2):137–154
22. Hu B, Lu Z, Li H et al (2014) Convolutional neural network architectures for matching natural language sentences. In: International conference on NIPS, pp 2042–2050
23. Severyn A, Moschitti A (2015) Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of SIGIR, pp 373–382
24. Zhang D, Lee WS (2003) Question classification using support vector machines. In: Proceedings of SIGIR, pp 26–32
25. Lees RB, Chomsky N (1957) Syntactic structures. Language 33(3 Part 1):375–408
26. Farabet C, Couprie C, Najman L et al (2013) Learning hierarchical features for scene labeling. IEEE Trans Pattern Anal Mach Intell 35(8):1915–1929
27. Echihabi A, Marcu D (2003) A noisy-channel approach to question answering. In: Proceedings of ACL, pp 16–23
28. Bordes A, Weston J, Usunier N (2014) Open question answering with weakly supervised embedding models. In: Joint European conference on machine learning and knowledge discovery in databases, pp 165–180
29. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: International conference on NIPS, pp 1097–1105

30. Collobert R, Weston J, Bottou L et al (2011) Natural language processing from scratch. J Mach Learn Res 12(1):2493–2537
31. Srivastava N, Hinton GE, Krizhevsky A et al (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
32. Hu M, Liu B (2004) Mining opinion features in customer reviews. In: Proceedings on AAAI, pp 755–760
33. Ding X, Liu B, Yu PS (2008) A holistic lexicon-based approach to opinion mining. In: Proceedings of international conference on web search and data mining. ACM, pp 231–240
34. Liu Q, Gao Z, Liu B et al (2015) Automated rule selection for aspect extraction in opinion mining. In: Proceedings of IJCAI, pp 1291–1297
35. Wiebe J, Wilson T, Cardie C (2005) Annotating expressions of opinions and emotions in language. Lang Resour Eval 39(2–3):165–210
36. Kingma DP, Adam JB (2015) A method for stochastic optimization. In: Proceedings of ICLR, pp 1–10
37. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of ICML, pp 1188–1196
38. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
39. Cho K, van Merrienboer B, Gulcehre C et al (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of EMNLP
40. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation. In: Proceedings on EMNLP, pp 1532–1543
41. Manning CD, Surdeanu M, Bauer J et al (2014) The Stanford CoreNLP natural language processing toolkit. In: Meeting of the Association for Computational Linguistics: System Demonstrations

Siheng Zhang is a Ph.D. student in the Institute of Automation, Chinese Academy of Sciences (CAS), supervised by professor Wensheng Zhang. Currently, his research includes artificial intelligence and machine learning, especially text mining and knowledge graph.

Wensheng Zhang received the Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences (CAS), in 2000. He is a professor of Machine Learning and Data Mining and the Director of the Research and Development Department with Institute of Automation, CAS. His research interests include computer vision, pattern recognition, artificial intelligence and computer–human interactions.

Jinghao Niu is a Ph.D. student in the Institute of Automation, Chinese Academy of Sciences (CAS), supervised by professor Wensheng Zhang. Currently, his research includes pattern recognition and machine learning, especially knowledge graph and deep learning.