Deep Neural Networks for Identification of Sentential Relations

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

Wenpeng Yin

München 2017

Erstgutachter: Prof. Dr. Hinrich Schütze

Zweitgutachter: Prof. Dr. Thomas Hofmann

Drittgutachter: Assistenzprofessor Kevin Gimpel, PhD

Tag der Einreichung: 07. Juni 2017
Tag der mündlichen Prüfung: 10. Oktober 2017

Eidesstattliche Versicherung
(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eides statt, dass die Dissertation von mir selbstständig ohne unerlaubte Beihilfe angefertigt ist.

München, den 16. Oktober 2017

Wenpeng Yin

Abstract

Natural language processing (NLP) is one of the most important technologies in the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate mostly in language: web search, advertisement, emails, customer service, language translation, etc. There are a large variety of underlying tasks and machine learning models powering NLP applications.

Recently, deep learning approaches have obtained exciting performance across a broad array of NLP tasks. These models can often be trained in an end-to-end paradigm without traditional, task-specific feature engineering.

This dissertation focuses on a specific NLP task — sentential relation identification. Successfully identifying the relations of two sentences can contribute greatly to some downstream NLP problems. For example, in open-domain question answering, if the system can recognize that a new question is a paraphrase of a previously observed question, the known answers can be returned directly, avoiding redundant reasoning. For another, it is also helpful to discover some latent knowledge, such as inferring "the weather is good today" from another description "it is sunny today". This dissertation presents some deep neural networks (DNNs) which are developed to handle this sentential relation identification problem. More specifically, this problem is addressed by this dissertation in the following three aspects.

(i) Sentential relation representation is built on the matching between phrases of arbitrary lengths. Stacked Convolutional Neural Networks (CNNs) are employed to model the sentences, so that each filter can cover a local phrase, and filters in lower level span shorter phrases and filters in higher level span longer phrases. CNNs in stack enable to model sentence phrases in different granularity and different abstraction.

(ii) Phrase matches contribute differently to the tasks. This motivates us to propose an attention mechanism in CNNs for these tasks, differing from the popular research of attention mechanisms in Recurrent Neural Networks (RNNs). Attention mechanisms are implemented in both convolution layer as well as pooling layer in *deep CNNs*, in order to figure out automatically *which phrase of one sen-*

tence matches a specific phrase of the other sentence. These matches are supposed to be indicative to the final decision. Another contribution in terms of attention mechanism is inspired by the observation that *some sentential relation identification task, like answer selection for multi-choice question answering, is mainly determined by phrase alignments of stronger degree; in contrast, some tasks such as textual entailment benefit more from the phrase alignments of weaker degree.* This motivates us to propose a dynamic "attentive pooling" to select phrase alignments of different intensities for different task categories.

(iii) In certain scenarios, sentential relation can only be successfully identified within specific background knowledge, such as the multi-choice question answering based on passage comprehension. In this case, the relation between two sentences (question and answer candidate) depends on not only the semantics in the two sentences, but also the information encoded in the given passage.

Overall, the work in this dissertation models sentential relations in hierarchical DNNs, different attentions and different background knowledge. All systems got state-of-the-art performances in representative tasks.

Zusammenfassung

Die Verarbeitung natürlicher Sprachen (engl.: natural language processing - NLP) ist eine der wichtigsten Technologien des Informationszeitalters. Weiterhin ist das Verstehen komplexer sprachlicher Ausdrücke ein essentieller Teil künstlicher Intelligenz. Anwendungen von NLP sind überall zu finden, da Menschen hauptsächlich über Sprache kommunizieren: Internetsuchen, Werbung, E-Mails, Kundenservice, Übersetzungen, etc. Es gibt eine große Anzahl Tasks und Modelle des maschinellen Lernens für NLP-Anwendungen.

In den letzten Jahren haben Deep-Learning-Ansätze vielversprechende Ergebnisse für eine große Anzahl verschiedener NLP-Tasks erzielt. Diese Modelle können oft end-to-end trainiert werden, kommen also ohne auf den Task zugeschnittene Feature aus.

Diese Dissertation hat einen speziellen NLP-Task als Fokus: Sententielle Relationsidentifizierung. Die Beziehung zwischen zwei Sätzen erfolgreich zu erkennen, kann die Performanz für nachfolgende NLP-Probleme stark verbessern. Für *open-domain question answering*, zum Beispiel, kann ein System, das erkennt, dass eine neue Frage eine Paraphrase einer bereits gesehenen Frage ist, die bekannte Antwort direkt zurückgeben und damit mehrfaches Schlussfolgern vermeiden. Zudem ist es auch hilfreich, zu Grunde liegendes Wissen zu entdecken, so wie das Schließen der Tatsache "das Wetter ist gut" aus der Beschreibung "es ist heute sonnig". Diese Dissertation stellt einige tiefe neuronale Netze (eng.: deep neural networks - DNNs) vor, die speziell für das Problem der sententiellen Relationsidentifizierung entwickelt wurden. Im Speziellen wird dieses Problem in dieser Dissertation unter den folgenden drei Aspekten behandelt: (i) Sententielle Relationsrepräsentationen basieren auf einem Matching zwischen Phrasen beliebiger Länge. Tiefe *convolutional neural networks* (CNNs) werden verwendet, um diese Sätze zu modellieren, sodass jeder Filter eine lokale Phrase abdecken kann, wobei Filter in niedrigeren Schichten kürzere und Filter in höheren Schichten längere Phrasen umfassen. Tiefe CNNs machen es möglich, Sätze in unterschiedlichen Granularitäten und Abstraktionsleveln zu modellieren. (ii) Matches zwischen Phrasen tragen unterschiedlich zu unterschiedlichen Tasks bei. Das motiviert uns, einen Attention-Mechanismus für CNNs für diese Tasks einzuführen,

der sich von dem bekannten Attention-Mechanismus für *recurrent neural networks* (RNNs) unterscheidet. Wir implementieren Attention-Mechanismen sowohl im *convolution layer* als auch im *pooling layer* tiefer CNNs, um herauszufinden, welche Phrasen eines Satzes bestimmten Phrasen eines anderen Satzes entsprechen. Wir erwarten, dass solche Matches die finale Entscheidung stark beeinflussen. Ein anderer Beitrag zu Attention-Mechanismen wurde von der Beobachtung inspiriert, dass einige sententielle Relationsidentifizierungstasks, zum Beispiel die Auswahl einer Antwort für *multi-choice question answering* hauptsächlich von Phrasenalignierungen stärkeren Grades bestimmt werden. Im Gegensatz dazu profitieren andere Tasks wie textuelles Schließen mehr von Phrasenalignierungen schwächeren Grades. Das motiviert uns, ein dynamisches "attentive pooling" zu entwickeln, um Phrasenalignierungen verschiedener Stärken für verschiedene Taskkategorien auszuwählen. (iii) In bestimmten Szenarien können sententielle Relationen nur mit entsprechendem Hintergrundwissen erfolgreich identifiziert werden, so wie *multi-choice question answering* auf der Grundlage des Verständnisses eines Absatzes. In diesem Fall hängt die Relation zwischen zwei Sätzen (der Frage und der möglichen Antwort) nicht nur von der Semantik der beiden Sätze, sondern auch von der in dem gegebenen Absatz enthaltenen Information ab.

Insgesamt modellieren die in dieser Dissertation enthaltenen Arbeiten sententielle Relationen in hierarchischen DNNs, mit verschiedenen Attention-Mechanismen und wenn unterschiedliches Hintergrundwissen zur Verfügung steht. Alle Systeme erzielen state-of-the-art Ergebnisse für die entsprechenden Tasks.

Contents

Publications and Declaration of Co-Authorship

Chapter 2

Chapter 2 corresponds to the following publication:

> Wenpeng Yin, Hinrich Schütze; **Discriminative Phrase Embedding for Paraphrase Identification**; Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Denver, Colorado, USA, May 31 - June 5, 2015), pages 1368–1373

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 3

Chapter 3 corresponds to the following publication:

> Wenpeng Yin, Hinrich Schütze; **Convolutional Neural Network for Paraphrase Identification**; Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Denver, Colorado, USA, May 31 - June 5, 2015), pages 901–911

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 4

Chapter 4 corresponds to the following publication:

> Wenpeng Yin, Hinrich Schütze; **MultiGranCNN: An Architecture for General Matching of Text Chunks on Multiple Levels of Granularity**; Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (Beijing, China, July 26-31, 2015) Volume 1: Long Papers, pages 63–73

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 5

Chapter 5 corresponds to the following publication:

> Wenpeng Yin, Hinrich Schütze, Bing Xiang and Bowen Zhou; **ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs**; Transactions of the Association for Computational Linguistics, Volume 4, pages 259–272, 2016

Bing Xiang and Bowen Zhou contributed a reference dataset and collaborated with me on designing the experimental evaluation. I also regularly discussed this work with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My coauthors assisted me in improving the draft.

Chapter 6

Chapter 6 corresponds to the following publication:

> Wenpeng Yin, Hinrich Schütze; **Task-Specific Attentive Pooling of Phrase Alignments Contributes to Sentence Matching**; Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Valencia, Spain, April 3-7, 2017): Volume 1, Long Papers, pages 699–709

I regularly discussed this work with my advisor, but I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My advisor assisted me in improving the draft.

Chapter 7

Chapter 7 corresponds to the following publication:

> Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou and Hinrich Schütze; **Simple Question Answering by Attentive Convolutional Neural Network**; Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (Osaka, Japan, December 11-17 2016), pages 1746–1756.

Mo Yu contributed the design and implementation of the active entity linker. I collaborated with Bing Xiang and Bowen Zhou on system design and experimental evaluation. I also regularly discussed this work with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My coauthors assisted me in improving the draft.

Chapter 8

Chapter 8 corresponds to the following publication:

> Wenpeng Yin, Sebastian Ebert and Hinrich Schütze; **Attention-Based Convolutional Neural Network for Machine Comprehension**; Proceedings of 2016 North American Chapter of the Association for Computational Linguistics Human-Computer Question Answering Workshop (San Diego, California, USA, June 16, 2016), pages 15–21

I collaborated with Sebastian Ebert on Figures 1, 2 and 3. I also regularly discussed this work with my coauthors. Apart from these explicitly declared exceptions, I conceived of the original research contributions and performed implementation and evaluation. I wrote the initial draft of the article and did most of the subsequent corrections. My coauthors assisted me in improving the draft.

München, 16. Oktober 2017

Wenpeng Yin

Chapter 1

Introduction

Sentences play a dominant role in expressing human beings' opinion, composing the semantics of single words to form an expression of a complete meaning. Further, a sequence of sentences constructs a passage that conveys more comprehensive content. This dissertation mainly studies the identification of sentential relations. Two sentences can correlate variously, such as paraphrasing, question-answer, textual entailment etc. Identifying the sentential relations helps in comprehending the sentence meaning as well as downstream natural language processing (NLP) tasks.

Almost all NLP problems, including sentential relation identification, were dominated by shallow machine learning methods with intensive feature engineering. The resurgence of deep neural networks (DNNs) enables the possibility of solving NLP problems via deep systems with no or fewer artificial features. Most NLP tasks have acquired state-of-the-art by DNN systems. This dissertation presents our work employing DNNs for sentential relation identification problem.

This chapter first introduces some basics of DNNs, including fully-connected feedforward neural networks, backpropagation algorithm, and two typical DNN types – Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), then covers some work for sentence representation learning, and finally elaborates our work in sentential relation identification.

1.1 Deep Neural Networks

Neural networks are powerful statistical models within the machine learning area. As parametrical methods, they learn hundreds or thousands of parameters of hypothesis (i.e., a non-linear function) to map some input data to some output data, so as to solve complex problems like natural language understanding. Neural networks introduce multiple types of representations on different levels of abstraction throughout their networks. The principle behind this is to build complex representations out of simpler ones to form a "hierarchy of concepts".

Neural networks are one of the most powerful modeling paradigms ever invented. The conventional approach to modeling requires breaking big problems up into many small, precisely defined tasks that the computer can easily perform. By contrast, in a neural network we do not tell the computer how to solve our problem. Instead, it learns from observational data, figuring out its own solution to the problem at hand.

Automatically learning from data sounds promising. However, until about a decade ago, there was no effective methodology for training neural networks to surpass more traditional approaches, except for a few specialized problems. What changed ten years ago was the *discovery of techniques for learning in so-called deep neural networks* (DNNs) (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2009). These techniques are now known as deep learning. They have been developed further, and today DNNs achieve outstanding performance on many important problems in computer vision, speech recognition, and natural language processing.

Before the DNN's resurgence, the neural networks discussed or applied in community are *shallow* in the sense that the number of system layers is usually at most two – logistic regression merely consists of input layer and output layer, PCA has an observed layer (i.e., the inputs) and a single hidden layer. A DNN is *deep*, on one hand, because mostly it has more than three layers of units, more importantly, when it is characterized by following two conditions (Bengio and LeCun, 2007; Cho, 2014):

- The network can be deepened by inserting more hidden layers;
- The parameters of each layer can be trained.

Therefore, it is worth mentioning that there is no absolute number of layers that distinguishes deep neural networks from shallow ones. In real-world applications, the depth of a DNN grows by a generic procedure of adding and training one or more layers, until it can yield satisfactory results for a target problem. In other words, the task and the dataset jointly decide how many layers a deep neural network needs.

In summary, deep neural networks are part of the broader machine learning field of learning representations of data that

- use a cascade of nonlinear layers of hidden units for feature learning. Each successive layer takes the output from the previous layer as input;
- learn hierarchical levels of representations that correspond to different levels of abstraction.

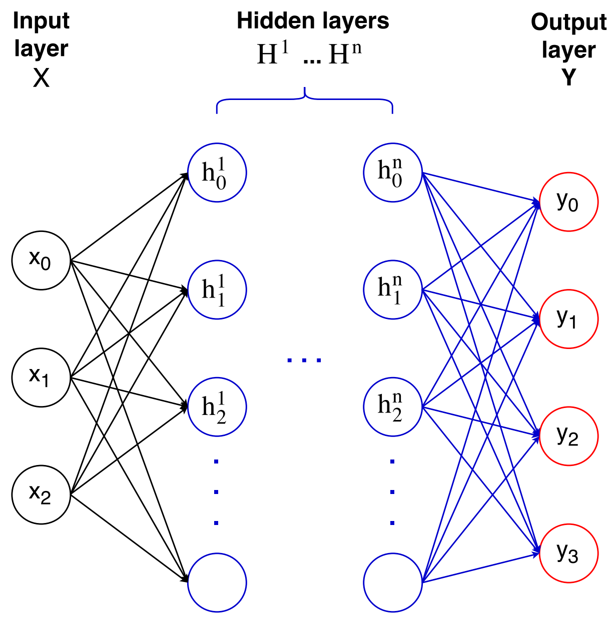


Figure 1.1 – *Fully-connected neural network. Three input features, four outputs and n hidden layers*

In a simple case, there might be two sets of neurons: one set that receives an input signal and one that sends an output signal. In conventional machine learning techniques, the models such as a classifier directly connect the input signal and output signal and make decisions based on the input signals. In a DNN, there are many nonlinear layers (a.k.a hidden layers) between the input and the output, allowing the algorithm to learn more sophisticated feature patterns and hence make more reliable decisions.

1.1.1 DNN Basics

As shown in Figure 1.1, typically a fully-connected DNN consists at least of three layer types: the input layer, the hidden layer and the output layer. By adding more hidden layers, the neural network can describe highly complex functions. The total number of layers specifies the model's depth, while the maximum size of the layers specifies the width of the model. The units of one layer are each connected to the units of the next layer.

The units of the input layer represent different features x_i of the input data, while the units of the output layer represent one of more classes y_i. A DNN describes a function $y^\star = f^\star(x)$ which maps the input features X over several hidden layers to the output classes Y. This function thereby approximates a real while unknown mapping function y=$f(x)$. A DNN approximates the function $f()$

by fitting the model's parameters θ so that predicted outputs $y^\star$ are as close as possible to the real outputs y.

$$\theta = \min_{\theta^\star} |f^\star(x, \theta^\star), y| \tag{1.1}$$

where $|\cdot|$ can be any distance function.

The learning process of a DNN consists of two iterative steps: i) *Forward propagation* – computing the prediction $y^\star$ by current parameters $\theta^\star$; ii) *Backpropagation* – updating the parameters $\theta^\star$ by the current loss between the prediction $y^\star$ and the ground truth y. We briefly describe the two steps in following parts.

Forward Propagation

In the feedforward step, a neuron in successive layer is computed by a weighted summation over all neurons of previous layer, for example, the j^{th} hidden neuron in the hidden layer H^1 in Figure 1.1 is calculated as follow:

$$h^1_j = \sigma(\sum_i x_i \cdot w^1_{ij} + b_j) \tag{1.2}$$

where weight w^1_{ij} connects input neuron x_i to the hidden neuron h^1_j, b_j acts as bias, and σ is a nonlinear function. In a similar way, all neuron values in subsequent layers can be derived layer-by-layer, until reaching the output layer. At the output layer, the prediction is compared with the ground truth, then we get a loss l:

$$l(\theta) = |f^\star(x, \theta), y| \tag{1.3}$$

In NLP, the breakthroughs of DNN lie in not only the stacked hidden layers, but also the parameterization of input layer (and output layer if needed). Contrary to image processing in which the input features are mostly the raw pixels, in NLP, the input words can be even denoted by a parameterized embeddings. That means the whole DNN, from inputs to outputs, can be parameterized, the objective is then to find the best parameter space which can best represent the inputs (and outputs in some cases) and approximate the mapping function $f(x)$.

Backpropagation

The parameter θ of the DNN has to be learned during training by approximating the model's hypothesis to the training data. In DNNs, it is impossible to compute error signal for internal neurons directly, because output values of these neurons are unknown. For many years the effective method for training multi-layer networks has been unknown. Only in the middle eighties the backpropagation

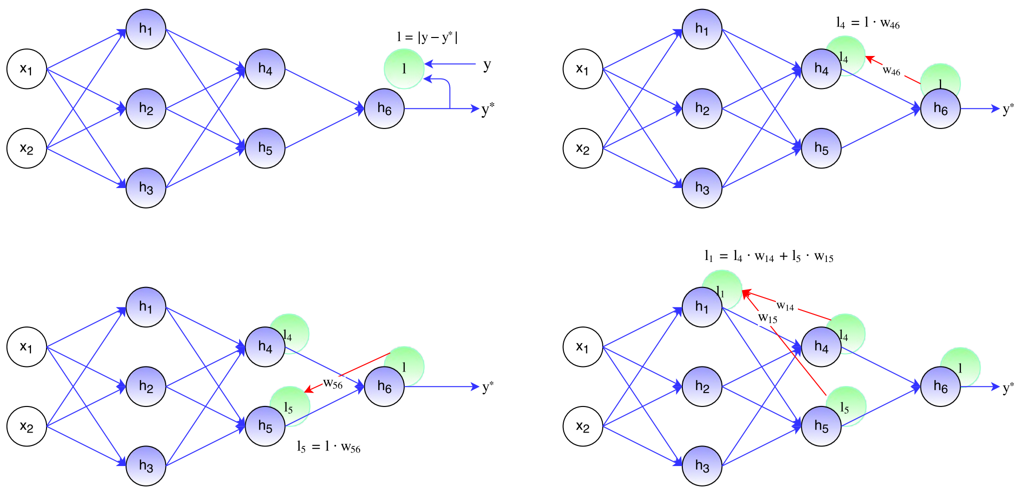


Figure 1.2 – *Error backpropagation examples*

algorithm was popularized by Rumelhart et al. (1986).[1] Before that, delta rule (Widrow and Hoff, 1960) was a gradient descent learning rule for updating the weights of the inputs to artificial neurons *in a single-layer neural network*. Backpropagation afterwards acts as a generalization of the delta rule *towards multi-layered feedforward networks*, making use of the chain rule to iteratively compute gradients for each layer.

The backpropagation algorithm is a learning procedure, which adjusts the parameters to minimize the loss l in Equation 1.3. The idea is to propagate error signal l (computed in single teaching step) back to all neurons, whose output signals were input for discussed neuron. Hence, backpropagation is also called "backward propagation of errors".

As Figure 1.2 shows, generally the error signal for i^{th} neuron in layer n backpropagated from the j^{th} neuron in layer $n+1$ is computed as:

$$l_i^n = w_{ij}^{n+1} \cdot l_j^{n+1} \tag{1.4}$$

The weight w_{ij}^{n+1} used to propagate errors back is equal to that used during computing output value in feedforward process. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). Because the loss is passed backwards through the network from the output layer to the hidden layers and further to the input layer, and those internal weights are updated based on those internal losses, this differentiation algorithm is called "backpropagation algorithm".

[1]A detailed account of the history of backpropagation is beyond the scope of this dissertation. See Schmidhuber (2015) for a comprehensive review of all work that led to the modern form of the backpropagation algorithm, in particular, (Werbos, 1974) and (Linnainmaa, 1970).

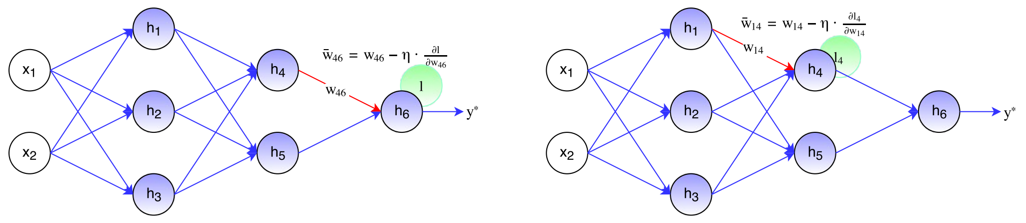

x1
x2
h1
h2
h3
h4
h5
h6
y*
$\bar{w}_{46} = w_{46} - \eta \cdot \frac{\partial l}{\partial w_{46}}$
w46
l
$\bar{w}_{14} = w_{14} - \eta \cdot \frac{\partial l_4}{\partial w_{14}}$
w14
l4

Figure 1.3 – *Weight Updating Examples*

The goal of backpropagation is to compute the partial derivatives $\partial l/\partial w$ and $\partial l/\partial b$ of the loss l with respect to any weight w or bias b in the network. Therefore, the partial derivatives of the loss l w.r.t each parameter have to be computed. When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. The formula below represents the derivative of a parameter w^n_{ij} with respect to the loss.

$$g^n_{ij} = \frac{\partial l^n_j}{\partial w^n_{ij}} \tag{1.5}$$

Then, the parameter w^n_{ij} will be updated as follows:

$$\bar{w}^n_{ij} = w^n_{ij} - \eta \cdot g^n_{ij} \tag{1.6}$$

where coefficient η is called learning rate, affecting network teaching speed. The weight updating is illustrated in Figure 1.3.

The choice of learning rate η is important, since a high value can cause too strong a change, causing the minimum to be missed, while a too low learning rate slows the training unnecessarily. In order to avoid oscillation inside the network such as alternating connection weights, and to improve the rate of convergence, most practical DNN systems use an adaptive learning rate, implemented by algorithms such as Adam (Kingma and Ba, 2015), AdaGrad (Duchi et al., 2011) and so on.

Strengths of Fully-connected DNNs

DNNs are widely recognized to have strengths in four folds:

- **Less need of engineered features**. One motivation to develop DNN systems to solve for example NLP problems is to relieve the burden of feature engineering. Feature defining and extraction occupy the dominant efforts in conventional machine learning system. DNNs instead try to learn task-specific features automatically from the dataset;

- **Strong modeling capability**. It was shown by (Hornik et al., 1989; Cybenko, 1989) that one-layer Multi Layer Perceptron is a universal approximator – it can approximate with any desired non-zero amount of error a family of functions that include all continuous functions on a closed and bounded subset of $\mathbb{R}^n$, and any function mapping from any finite dimensional discrete space to another (Goldberg, 2016);

- **Parameterize all system components**. In conventional shallow machine learning systems, the input X and the output Y are known and fixed, only their connection are parameterized and trained. In DNNs, all system components, including inputs, connections and outputs, are parameterized – they are treated as weights to train until convergence to a good parameter space which yields good performance in NLP task;

- **Strong generalization power**. DNNs often have far more trainable parameters than the number of training samples. Nonetheless, some of these models exhibit remarkably small generalization error, i.e., difference between "training error" and "test error" (Zhang et al., 2017). The strong generalization power of DNNs can be attributed to two reasons: (i) Distributed representations of inputs in both training data as well as testing data. These parameterized continuous representations build connections between samples which, in conventional representation schemes, stand far away from each other; (ii) Various and effective regularization approaches have been successfully applied to DNN systems, such as dropout (Srivastava et al., 2014), layer normalization (Ba et al., 2016) and so on.

Limitations of Fully-connected DNNs

However, there are some limitations for the basic feedforward deep neural networks, which motivate model modification in mainly two categories:

- **Too many parameters to train**. As shown in Figure 1.1, each two adjacent layers are fully connected. This results in a requirement of a huge number of parameters, for example, given input feature size n and the neuron size m in the first hidden layer, there will be $n \times m$ paramters to train merely in this single hidden layer.

  An alternative is to split the large input into multiple small groups, sharing parameters across groups. Each group provides some local features, then the system composes all of them to form the feature representation of the whole input. This is achieved by convolutional neural networks (LeCun et al., 1998).

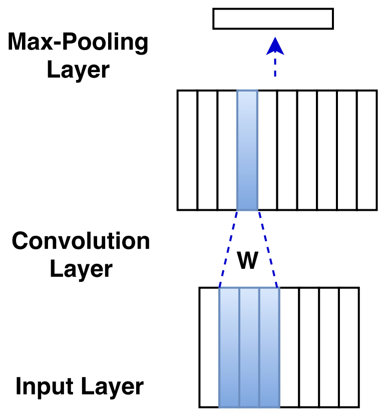


Figure 1.4 – *Convolutional Neural Network*

- **Hard to deal with sequence data with variable lengths**. In above discussion, we always assume that the input size is fixed. However, in reality this may be impossible to control. Especially, text input mostly has different lengths, and in online application, it is also impossible to know in advance how much input there will be in streaming data.

  Hence, an ideal way is to make the system read input elements one by one, sharing parameters at each step. This is the principle of another typical DNN system – recurrent neural networks (Elman, 1990).

Next, we introduce convolutional neural networks and recurrent neural networks.

1.1.2 Convolutional Neural Network

When all input units are connected with all hidden units, the network architecture is described as "fully connected". These networks with complete connection between all units have many parameters to learn, the parameter matrices become very large, and the matrix multiplications are computationally expensive. The idea of Convolutional Neural Networks (CNNs) (LeCun et al., 1998) is to reduce the connections between the input units and the hidden units, instead of fully connect them. Each hidden unit will obtain weighted inputs only from selected input units. With a sequence of words as input, this means only a local phrase (i.e., n-gram) will be processed by a hidden unit in CNN. In Figure 1.4, the idea of the restricted connection between the input and hidden units is illustrated. Next, we elaborate the architecture of a CNN for processing a sentence input.

Input Layer

Sequence x contains m entries. Each entry is represented by a d-dimensional dense vector; thus the input x is represented as a feature map of dimensionality $d \times m$. Figure 1.4 shows the input layer as the lower rectangle with multiple columns.

Convolution Layer

is used for representation learning from sliding n-grams. For an input sequence with m entries: $x_1, x_2, \ldots, x_m$, let vector $\mathbf{c}_i \in \mathbb{R}^{nd}$ be the concatenated embeddings of n entries $x_{i-n+1}, \ldots, x_i$ where n is the filter width and $0 < i < m + n$. Embeddings for x_i, $i < 1$ or $i > m$, are zero padded. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^{d'}$ for the n-gram $x_{i-n+1}, \ldots, x_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d' \times nd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b}) \tag{1.7}$$

where bias $\mathbf{b} \in \mathbb{R}^{d'}$.

Maxpooling Layer

All n-gram representations $\mathbf{p}_i$ $(i = 1 \cdots m + n - 1)$ are used to generate the representation of input sequence x by maxpooling: $\mathbf{x}_j = \max(\mathbf{p}_{1,j}, \mathbf{p}_{2,j}, \cdots)$ $(j = 1, \cdots, d)$. The objective of maxpooling is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions. This is done in part to help prevent over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.

1.1.3 Recurrent Neural Networks

The recurrent neural network is also called Elman network (Elman, 1990). Its architecture is shown in Figure 1.5. Each current input x_t is composed with the previous hidden state h_{t-1} to generate a new hidden state at time t as follow:

$$\mathbf{h}_t = \sigma(\mathbf{V}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \tag{1.8}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ represents the token in x at position t, $\mathbf{h}_t \in \mathbb{R}^h$ is the hidden state at t, supposed to encode the history x_1, $\cdots$, x_t. $\mathbf{V} \in \mathbb{R}^{h \times d}$ and $\mathbf{U} \in \mathbb{R}^{h \times h}$ are parameters.

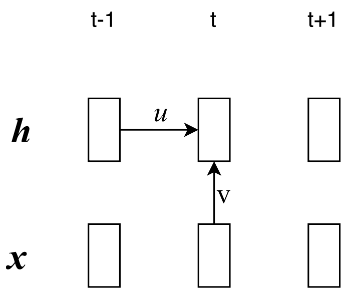


Figure 1.5 – *Simple Recurrent Neural Network*

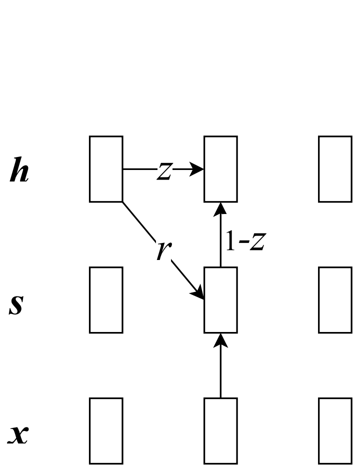


Figure 1.6 – *GRU*

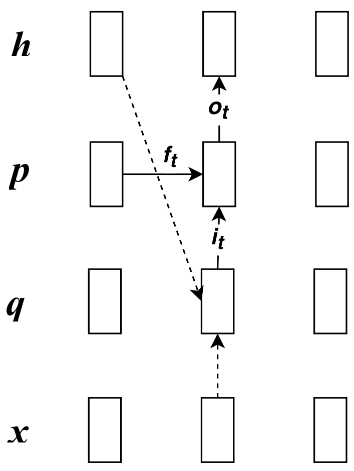


Figure 1.7 – *LSTM*

By this recurrent procedure, all inputs in X can be encoded sequentially into a global representation. Unfortunately, vanishing gradient problem prevents standard RNNs from learning long-term dependencies. LSTMs (Long Short Term Memory (Hochreiter and Schmidhuber, 1997)) were designed to combat vanishing gradients through a gating mechanism. GRUs (Gated Recurrent Units), first used in (Cho et al., 2014a), are a simpler variant of LSTMs that share many of the same properties.

Gated Recurrent Unit (GRU)

GRU, as shown in Figure 1.6, models text x as follows:

$$\mathbf{z} = \sigma(\mathbf{U}^z\mathbf{x}_t + \mathbf{W}^z\mathbf{h}_{t-1} + \mathbf{b}_z) \tag{1.9}$$
$$\mathbf{r} = \sigma(\mathbf{U}^r\mathbf{x}_t + \mathbf{W}^r\mathbf{h}_{t-1} + \mathbf{b}_r) \tag{1.10}$$
$$\mathbf{s}_t = \tanh(\mathbf{U}^s\mathbf{x}_t + \mathbf{W}^s(\mathbf{h}_{t-1} \circ \mathbf{r}) + \mathbf{b}_s) \tag{1.11}$$
$$\mathbf{h}_t = (1 - \mathbf{z}) \circ \mathbf{s}_t + \mathbf{z} \circ \mathbf{h}_{t-1} \tag{1.12}$$

$\mathbf{z} \in \mathbb{R}^{h\times h}$ and $\mathbf{r} \in \mathbb{R}^{h\times h}$ are two gates. All $\mathbf{U} \in \mathbb{R}^{h\times d}$,$\mathbf{W} \in \mathbb{R}^{h\times h}$ are parameters. "$\circ$" denotes element-wise product. In order to generate the hidden state h_t, GRU first generates a temporary result s_t by a *tanh* non-linearity over the ensemble of input x_t and the preceding hidden state h_{t-1}, then h_t equals to the trade-off between this temporary result s_t and history h_{t-1} by gate z.

Long Short-Term Memory (LSTM)

LSTM, denoted in Figure 1.7, models the word sequence x as follows:

$$\mathbf{i}_t = \sigma(\mathbf{x}_t\mathbf{U}^i\mathbf{x}_t + \mathbf{W}^i\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{1.13}$$
$$\mathbf{f}_t = \sigma(\mathbf{U}^f\mathbf{x}_t + \mathbf{W}^f\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{1.14}$$
$$\mathbf{o}_t = \sigma(\mathbf{U}^o\mathbf{x}_t + \mathbf{W}^o\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{1.15}$$
$$\mathbf{q}_t = \tanh(\mathbf{U}^q\mathbf{x}_t + \mathbf{W}^q\mathbf{h}_{t-1} + \mathbf{b}_q) \tag{1.16}$$
$$\mathbf{p}_t = \mathbf{f}_t \circ \mathbf{p}_{t-1} + \mathbf{i}_t \circ \mathbf{q}_t \tag{1.17}$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{p}_t) \tag{1.18}$$

LSTM has three gates: input gate i_t, forget gate f_t and output gate o_t. All gates are generated by a *sigmoid* function over the ensemble of input x_t and the preceding hidden state h_{t-1}. In order to generate the hidden state at current step t, it first generates a temporary result q_t by a *tanh* non-linearity over the ensemble of input x_t and the preceding hidden state h_{t-1}, then combines this temporary result q_t with history p_{t-1} by input gate i_t and forget gate f_t respectively to get an updated history p_t, finally uses output gate o_t over this updated history p_t to get the final hidden state h_t.

1.1.4 CNNs vs. RNNs

In above sections, we have introduced the rationales and architectures of two typical deep neural networks – CNNs and RNNs. Now, we have a brief comparison between them:

- First CNNs and RNNs have different styles in dealing with input. CNNs divide the input into small parts, then compose the features of those parts. RNNs read the input entries one by one, accumulating the features until reaching the last entry, so the final feature vector is expected to represent the whole input sequence. As a result, CNNs are hierarchical architecture, RNNs instead are sequential architecture.

- Second, as CNNs mostly focus on local regions, CNNs can encode the local structure while maybe losing the global structure. RNNs can encode both

	CNN	RNN
handle input	divide then compose	one-by-one
system	hierarchical	sequential
input structure	local structure	local & global structure
long-distance dependency	hard	easier
good at	pattern detection	global semantics

Table 1.1 – *CNNs vs. RNNs*

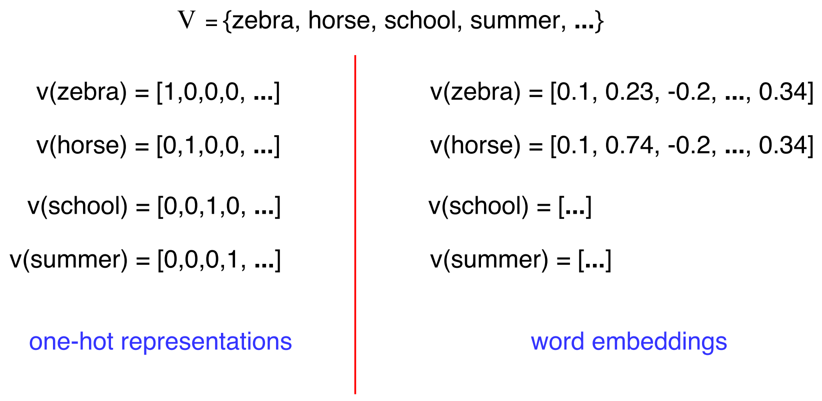

V = {zebra, horse, school, summer, ...}
v(zebra) = [1,0,0,0, ...]
v(horse) = [0,1,0,0, ...]
v(school) = [0,0,1,0, ...]
v(summer) = [0,0,0,1, ...]
one-hot representations
v(zebra) = [0.1, 0.23, -0.2, ..., 0.34]
v(horse) = [0.1, 0.74, -0.2, ..., 0.34]
v(school) = [...]
v(summer) = [...]
word embeddings

Figure 1.8 – *One-hot representations vs. word embeddings*

local as well as global structure, such as some long-distance dependency. As a result, CNNs are widely developed to detect position-independent patterns while RNNs are widely utilized to learn global semantics.

We summarize the comparison in Table 1.1.

1.2 Word Distributed Representations

The success of DNNs in NLP partially relies on the outstanding distributed representations of words. We can also treat a DNN as a system with two modules, one is representing input units, the other is composing the unit representations. In this perspective, effective word representations act as the first backbone of a successful DNN system.

1.2.1 Word Embedding Basics

Word distributed representations – also called "word embeddings" – are low-dimensional, dense vectors with continuous values. Figure 1.8 depicts the comparison between conventional one-hot word representations and word embeddings.

Given a vocabulary with size V, one-hot representation denotes each single word as a binary vector of length $|V|$ with one value 1 at the word-specific index and remaining values 0. It enjoys simplicity, however, it is memory inefficient and the word similarity is unable to be detected. For example, based on the left part in Figure 1.8, each pair of words share zero similarity, even including the pair "(zebra, horse)" which denote two very similar animals.

Unlike the conventional one-hot representations, a single word in embedding space is a d-dimensional vector (mostly $d << |V|$); Similar words will have similar vectors – information is shared between similar words. Consider the right part of Figure 1.8, "zebra" and "horse" have same feature values in all indices except for the second location. It indicates the high similarity of these two words. One benefit of using dense and low-dimensional vectors is computational, the other is generalization power – if we believe some features may provide similar clues, it is worthwhile to provide a representation that is able to capture these similarities. For example, assume we have observed the word "dog" many times during training, but only observed the word "cat" a handful of times, or not at all. If each of the words is associated with its own dimension, occurrences of "dog" will not tell us anything about the occurrences of "cat". However, in the dense vector representations the learned vector for "dog" may be similar to the learned vector from "cat", allowing the model to share statistical strength between the two events.

1.2.2 Word Embedding Learning

Due to the importance of word embeddings in DNN systems, there are large numbers of work specifically study the learning of high-quality word embeddings. Different with sentential relation identification, word embedding learning can mostly be carried out over unlabeled big data, such as Wikipedia, news and so on. We can organize the word embedding work into three categories: rich context based, word shape based and meta embeddings.

Rich Context Word embeddings are typically induced using neural language models, which use neural networks as the underlying predictive model (Bengio, 2008). Collobert and Weston (2008) present a neural language model that could be trained over billions of words, because the gradient of the loss was computed stochastically over a small sample of possible outputs. For each training update, the model reads an n-gram $x = (w_1, \cdots, w_n)$ from the corpus. The model concatenates the learned embeddings of the n words, and also creates a corrupted or noise n-gram $\bar{x} = (w_1, \cdots, w_{n-q}, \bar{w}_n)$, where $\bar{w}_n \neq w_n$ is chosen uniformly from the vocabulary. Then, the system predicts a score $s(x)$ for x by passing

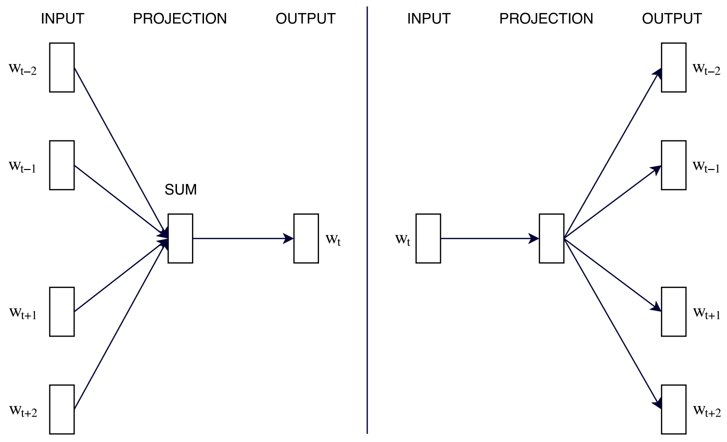


Figure 1.9 – *CBOW vs. Skip-gram*

the concatenated representation vector through a single hidden layer neural network. The training criterion is that n-grams that are present in the training corpus like x must have a score at least some margin higher than the corrupted n-grams like $\bar{x}$. The log-bilinear model (Mnih and Hinton, 2007, 2008) is a probabilistic and linear neural model. Given an n-gram, the model concatenates the embeddings of the $n-1$ first words, and learns a linear model to predict the embedding of the last word. CBOW and Skip-gram models in (Mikolov et al., 2013b), as shown in Figure 1.9 further simplifies the log-bilinear model – no hidden layer, no structure information. GloVe (Pennington et al., 2014) is a global log-bilinear regression model that combines the advantages of two major model families in the word representation literature: global matrix factorization and local context window methods.

Word Shape Very soon, NLP community found that context-based embedding learning approaches can not encode the inner-structure of words. For example, GloVe word embeddings can not reflect the shape correlation between words "supervised" and "unsupervised" as both share very similar context. The continuous skip-gram model in (Mikolov et al., 2013b) was modified by Bojanowski et al. (2017) to consider subword units, representing words by a sum of its character n-grams.

Meta Embeddings Apart from trying to improve the word embedding quality in a single system, some work (Yin and Schütze, 2016; Bollegala et al., 2017; Muromägi et al., 2017) tries to do ensemble over multiple different versions of pretrained word embeddings to get "meta-embeddings". Those individual embedding sets are complementary since their training data and training algorithms are mostly different. Ensemble approaches can on one hand enhance the ultimate embedding quality, on the other hand, extend the vocabulary.

1.3 Sentence Representation Learning

As sentential relation identification addressed in this dissertation relies on sentence representation learning, this section therefore briefly describes the progresses of sentence representation learning via DNNs. Similar with image processing, sentences are also initialized into a matrix representation, such as a matrix with columns denoting word embeddings in sequence. Then, CNNs model a sentence locally and hierarchically while RNNs model sequentially. Depending on if human annotation is required, sentence representation learning can be carried out in unsupervised or supervised manner.

1.3.1 Unsupervised Sentence Representation Learning

Without access of annotated labels, sentences learn representations by composing embeddings of constituent words to either *predict their own content words* or *predict adjacent sentences*.

The most straightforward way is to compose pre-trained word embeddings by element-wise addition (Mitchell and Lapata, 2010), it acts as a strong baseline in many NLP systems despite its simplicity.

Some work gets inspiration from n-gram language modeling. For example, ParagraphVector (Le and Mikolov, 2014) first initializes a sentence representation randomly, then combines it with some context words to predict the next word. CNNLM (Yin and Pei, 2015) uses a CNN to encode a piece of context, then uses the generated context representation to predict the next word.

Inspired by some word embedding learning approaches in which a word embedding is trained by predicting the context words, a sentence representation can also be derived by predicting adjacent sentences, assuming that all sentences in the corpus appear in a natural order. For example, given consecutive sentences S_{i-1}, S_i, S_{i+1} in a document, the SkipThought model (Kiros et al., 2015) uses GRU to encode all sentences, then predicts target sentences S_{i-1} and S_{i+1} given source sentence S_i. FastSent (Hill et al., 2016) is a simplified version of SkipThought through replacing sentence encoding model GRU by simpler addition – each sentence starts as the sum of word embeddings.

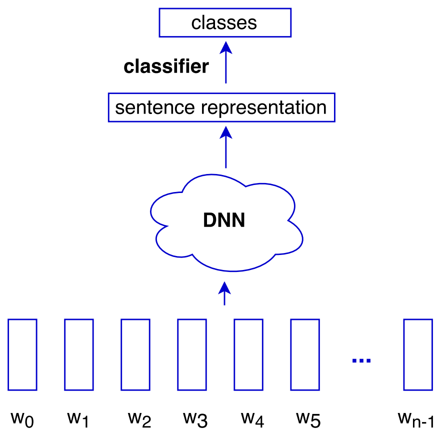


Figure 1.10 – *Supervised sentence representation learning by DNN*

1.3.2 Supervised Sentence Representation Learning

Figure 1.10 shows a common framework for supervised sentence representation learning by DNN (assuming a classification task). As input, each word w_i ($i \in \{0, \cdots, n-1\}$) is denoted by an embedding (randomly initialized or pretrained) of dimension d, then the whole sentence is represented as a matrix S $\in \mathbb{R}^{d \times n}$; A DNN system works on this sentence input format to generate a global sentence representation, which is finally forwarded into a classifier to figure out the label of this input sentence. Literature mainly made progresses in terms of *enriching input representations*, *enriching DNN expressivity* and *improving objective function*.

Enrich Input Representation

The pioneering work (Collobert and Weston, 2008; Collobert et al., 2011) obtained great success by presenting words into *word embeddings*. This kind of initialization afterwards acts as a mainstream in downstream NLP tasks (Kalchbrenner et al., 2014). Some work (Kim, 2014; Yin and Schütze, 2015; Zhang et al., 2016) explored initializing word by multiple pretrained word embeddings, as different pretrained embedding versions are supposed to provide complementary information.

However, recognizing and representing language units at word level brings challenges in processing rare words and morphological variants. Recently, increasing numbers of work pay attention to studying the *subword structures*. A typical stream of work lies in character-level sentence encoding. It does not require word segmentation any more, instead, treating a sentence, or more generally a piece of text, as a sequence of characters. Examples include encoding character sequences (dos Santos and Guimarães, 2015; Ling et al., 2015; Ballesteros et al., 2015; Zhang et al., 2015b; Kim et al., 2016), encoding character n-grams (Wieting et al., 2016) and encoding morphology (Cotterell et al., 2016; Kann et al., 2016) etc.

In addition to the word embeddings at input layer, linguistic features are often incorporated into DNNs for better performance. For example, Yu et al. (2016) add part-of-speech tags to the words in machine comprehension task. Vu et al. (2016) consider position features between generic words and entity mentions for relation classification. Chapter 8 also introduces our work which considered question types, i.e., "wh-" words, to promote the representation learning of question sentences in question answering. Generally, linguistic features can provide strong support to the DNN systems, especially in the case of limited training set.

Enrich DNN

Collobert and Weston (2008) and Collobert et al. (2011) used basic convolution layer and max-pooling layer to model sentences. Kalchbrenner et al. (2014) proposed *k-max pooling* for CNN. Mou et al. (2015) built CNN upon constituency trees and dependency trees of a sentence. Palangi et al. (2016) modeled sentences by LSTMs. Vu et al. (2016) combined CNN and RNN for sentence-level relation classification. Yin and Schütze (2017) equipped CNNs with the commonly employed attention mechanism in RNNs to get outstanding performance in text classification task.

Improve Objective Functions

For sentence classification tasks, the most commonly-used loss function is negative likelihood (a.k.a "softmax loss"). dos Santos et al. (2015) presented a ranking loss to make the true label score above a positive threshold and the false label score below a negative threshold. Liu et al. (2016) proposed a generalized large-margin softmax loss which explicitly encourages intra-class compactness and inter-class separability between learned features. The purpose is to generalize the softmax loss to a more general large-margin softmax loss in terms of angular similarity, leading to potentially larger angular separability between learned features.

1.4 Tasks for Sentential Relation Identification

The main content of this dissertation summarizes our work in addressing sentential relation identification problem. Sentences can have various relation types. This section first gives an overview of some typical sentential relation identification tasks, then presents their comparison and potential challenges.

1.4.1 Task Introduction

Answer Selection (AS)

When a question meets multiple answer candidates, a system needs to pick out the correct one. Considering the following example:

Q: what bird family is the owl

C_1: Most are solitary and nocturnal, with some exceptions (e.g., the Northern Hawk Owl).

C_2: Owls hunt mostly small mammals, insects, and other birds, although a few species specialize in hunting fish.

C_3: Owls are a group of birds that belong to the order Strigiformes, constituting 200 extant bird of prey species.

C_4: They are found in all regions of the Earth except Antarctica, most of Greenland and some remote islands.

The question Q needs to pick up the correct answer (C_3 here) from some candidates C_i (i=1,2,3,4).

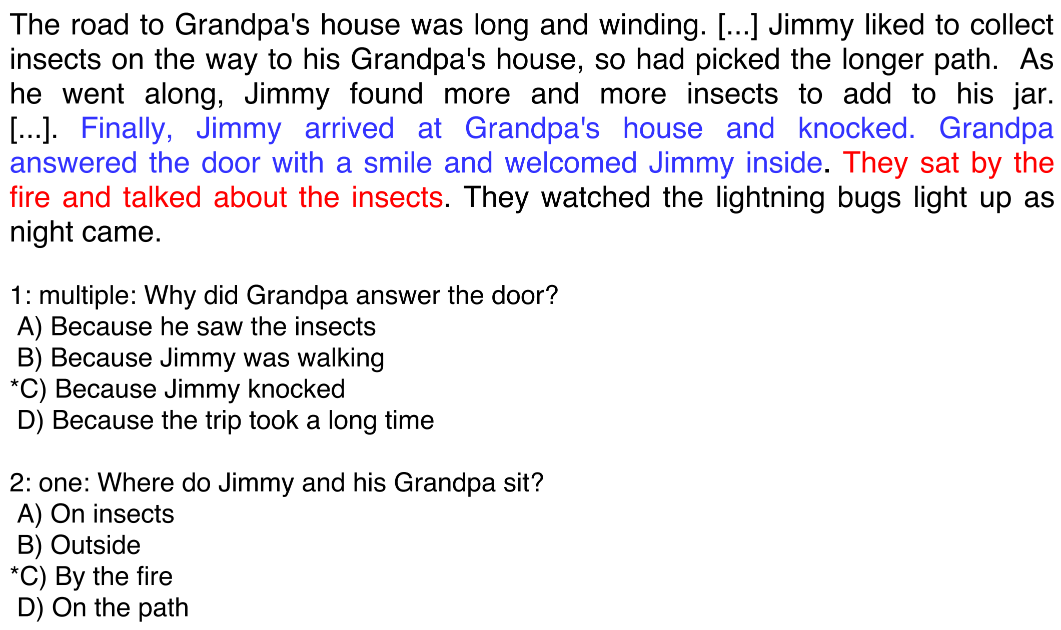
In above example, the relation (Q, C_i) can be recognized based on the content within the sentences. In certain scenarios, their relation can only be inferred through extra knowledge. For example, in the machine comprehension task depicted in Figure 1.11, a question can only find the right answer based on correct comprehension of the passage. We call this situation as "sentential relation identification in background" which will be introduced in Section 1.5.4

Paraphrase Identification (PI)

A paraphrase is a restatement of the meaning of a text or passage using other words. Paraphrase identification therefore is the task of examining two text entities (e.g., sentence) and determining whether they have the same meaning. In order to obtain high accuracy on this task, thorough syntactic and semantic analysis of the two text entities is required.

According to granularity, paraphrases are of four types: i) Lexical level, such as "solve" and "resolve"; ii) Phrase level, such as "look after" and "take care of"; iii) Sentence level, such as "the table was set up in the carriage shed" and "the table was laid under the cart-shed"; iv) Discourse level. In this dissertation, we focus on the sentential paraphrase identification – given any pair of sentences, automatically identify whether these two sentences are paraphrases.

Paraphrase identification is potentially important for simplifying input sentences and alleviating data sparseness in machine translation (Callison-Burch et al.,


The road to Grandpa's house was long and winding. [...] Jimmy liked to collect insects on the way to his Grandpa's house, so had picked the longer path. As he went along, Jimmy found more and more insects to add to his jar. [...]. Finally, Jimmy arrived at Grandpa's house and knocked. Grandpa answered the door with a smile and welcomed Jimmy inside. They sat by the fire and talked about the insects. They watched the lightning bugs light up as night came.

1: multiple: Why did Grandpa answer the door?
 A) Because he saw the insects
 B) Because Jimmy was walking
*C) Because Jimmy knocked
 D) Because the trip took a long time

2: one: Where do Jimmy and his Grandpa sit?
 A) On insects
 B) Outside
*C) By the fire
 D) On the path

Figure 1.11 – *Machine Comprehension task*

2006; Marton et al., 2009), question reformulation in question answering (Tomuro, 2003) and information retrieval (Zhang et al., 2015a), sentence clustering in summarization (Radev et al., 2004), sentence rewriting in natural language generation (Barzilay and Lee, 2003; Mitchell et al., 2014) and so on.

Textual Entailment (TE)

TE in natural language processing is a directional relation between text fragments. The relation holds whenever the truth of one text fragment follows from another text. In the TE framework, the entailing and entailed texts are termed premise (t) and hypothesis (h), respectively. And there are, as a result, three classes: entailment, neutral and contradiction. Contrary to paraphrase, TE is similar but weakens the relationship to be unidirectional.

Many NLP applications, like question answering, information extraction, (multi-document) summarization and machine translation evaluation, need to recognize that a particular target meaning can be inferred from different text variants. Typically entailment is used as part of a larger system, for example in a prediction system to filter out trivial or obvious predictions (Mombourquette et al., 2017; Chang et al., 2017).

1.4.2 Task Analysis

This dissertation mainly discusses textual entailment, paraphrase identification and answer sentence selection tasks.

Both textual entailment and answer sentence selection tasks are directional relations, i.e., the relation can not hold if we exchange the two text pieces. Two paraphrasing sentences have also entailment relationship towards each other. In the machine comprehension task depicted in Figure 1.11, the document should be able to entail the information in the combination of question and the correct answer sentence.

Both textual entailment and paraphrase identification tasks are mostly treated as classification problem, while the answer sentence selection and machine comprehension tasks are treated as ranking problems – picking the top-1 ranked sentence as the predicted answer. In addition, badly-matched n-grams are observed to be more decisive in textual entailment task while well-matched n-grams are found more decisive in paraphrase identification and answer selection tasks. Considering textual entailment example "a couple is eating *inside* at a table || a couple is eating *outside* at a table", the word pair "(inside, outside)" is badly-matched as other words are overlapping words, so the relation of these two sentences are essentially determined by the pair "(inside, outside)". Considering an answer sentence selection example "*how big* is *Munich* || according to the Wikipedia 2005, the *city* had *a population of* 1,700,381", the two pairs "(how big, a population of)" and "(Munich, city)" are better-matched than other n-grams, and the system can very likely determine the sentence relation even though there are some badly-matched n-grams such as "according to the Wikipedia 2005".

Consistently, all these tasks face the challenges – various relation types, open domain sentences. Textual entailment mostly have three relation types – *entailment, contradictory, neutral*, paraphrase identification has binary relations – paraphrasing or not, question answering (answer sentence selection and machine comprehension) include more unexpected relation types and descriptions. DNN systems are expected to have good representations and strong generalization.

1.5 Systems for Sentential Relation Identification

1.5.1 Independent Sentence Modeling

The most basic framework to identify sentential relations is to examine two sentence representations which are derived separately like we introduced in Section 1.3. Figure 1.12 depicts the Siamese architecture (Bromley et al., 1993) in which pairs of inputs are processed by two copies of a neural network. The pair of

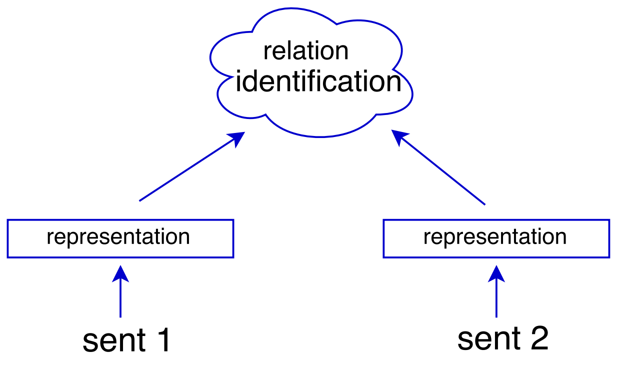


Figure 1.12 – *Siamese architecture for modeling sentential relations*

networks are trained to make their output vectors similar for input pairs that are labeled as similar, and dissimilar for input pairs that are labeled as dissimilar. Examples include (Yu et al., 2014; Yang et al., 2015; Hosseini-Asl and Guha, 2015) etc. However, an apparent shortcoming of this Siamese system is that sentences are encoded independently, without considering the mutual impact. This is not beneficial for identifying sentential relations.

Next we use three subsections to elaborate our work in this sentential relation identification task.

1.5.2 Sentence Interaction

In our first three publications in Chapters 2-4, we will introduce our first effort to better model a sentence by considering its counterpart. Considering the following sentences S_1 and S_2:

S_1: please **turn** the light **off**

S_2: please **unplug** the light

They can only be correctly recognized as paraphrase if the discontinuous phrase "turn $\cdots$ off" in S_1 can be figured out semantically equivalent to "unplug" in S_2. Based on this observation, we realize that successful sentential relation detection first *relies on representation of phrases of arbitrary granularity*. A primary attempt, to the end, is to detect those phrases, learning embeddings for them in a large corpus. Based on representations of words and phrasal units, a sentence can be better represented as its components are more semantic-complete units now.

Detecting phrases in large corpus is not trivial in practical, and pre-detected phrases may not cover the phrases appearing in the sentences. A more desirable

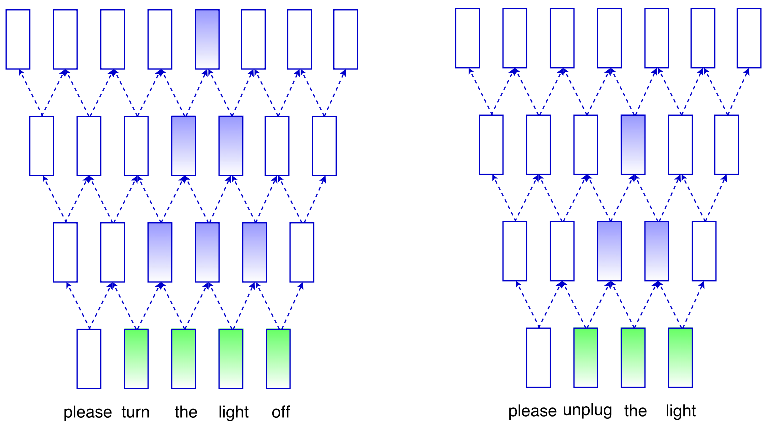


Figure 1.13 – *Stacked CNN for phrase representation*

way is to explore DNN to detect phrases automatically. Socher et al. (2011) used parser to split a sentence, then phrases lie in leaves of the parsing trees, finally recursive neural network was employed to encode the parsing trees. However, parsing is not totally reliable in open-domain applications, and only detects limited phrases based on the sentence structure. The 3rd and 4th chapters of this dissertation introduce how we develop stacked CNN systems to detect multigranular and hierarchical phrases from sentences. The benefit of CNN over recursive NN is that CNN does not depend on any toolkits, totally training from scratch.

Figure 1.13 depicts stacked CNN layers, with filter width 2, to detect and represent phrases in sentences “please turn the light off” and “please unplug the light”. As a result, the top blue block on the left figure represents the phrase “turn the light off” and the top blue block on the right figure represent the phrase “unplug the light”. Our work in Chapters 3-4 will elaborate how to make use of their interaction to enhance the sentential relation identification problems. Figure 1.14 depicts using RNN to encode the sentence from left to right, then choosing the hidden states of head word and tail word of a phrase to form the phrase representation.

Based on phrase representations detected by CNN or RNN, we can model the sentence interaction by comparing any phrases of one sentence with any phrases of the other sentence. This fine-grained comparison is supposed to provide more detailed information than the Siamese framework introduced in Section 1.5.1.

1.5.3 Attention Mechanism

Conventional sentential relation identification does not distinguish which parts are more indicative for the task. Human beings, instead, can figure out which part of a sentence matches well to a specific part of the other sentence. For exam-

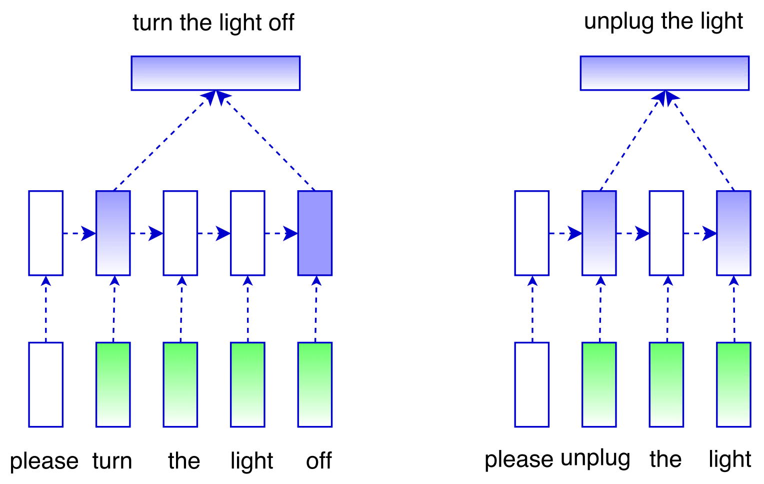


Figure 1.14 – *RNN for phrase representation*

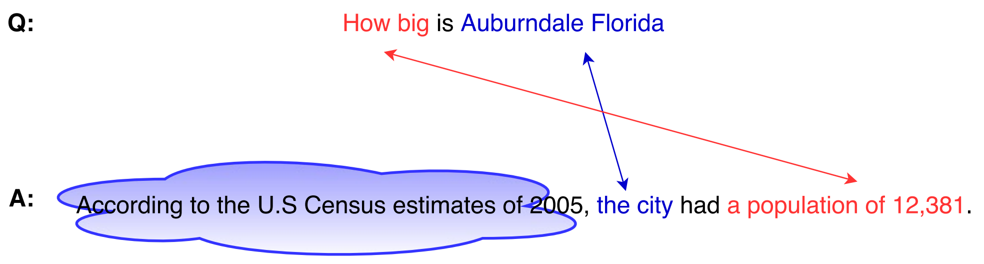

Q:
How big is Auburndale Florida
A:
According to the U.S Census estimates of 2005, the city had a population of 12,381.

Figure 1.15 – *Attention example for answer selection task*

ple, in answer selection task for multi-choice question answering shown in Figure 1.15, to successfully identify the relationship between the question “How big is Auburndale Florida” and the answer candidate “According to the U.S. Census estimates of 2005, the city had a population of 12,381”, we should mainly compare the phrase pairs (How big, a population of 12,381) and (Auburndale Florida, the city), the long phrase “According to the U.S. Census estimates of 2005” has no semantically matching counterpart, hence it can be *neglected* without influencing the decision.

Instead, in the textual entailment example in Figure 1.16, a correct decision between **S1** “*the kids are playing outdoors and the man is smiling nearby*” and **S2** “*the kids are playing in a yard and an old man is standing in the background*” depends on attention over the match between phrase pairs (outdoors, in a yard) and (the man is smiling nearby, an old man is standing in the background), the repeating part “the kids are playing” in both sentences is less indicative – discarding its two occurrences in the two sentences will not influence the result.

So, attention mechanism is intensively explored recently. A representative

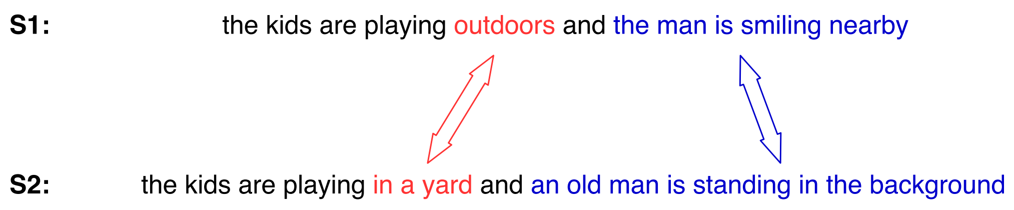

S1:
the kids are playing outdoors and the man is smiling nearby
S2:
the kids are playing in a yard and an old man is standing in the background

Figure 1.16 – *Attention example for textual entailment task*

attention mechanism in sentential relation identification is by Rocktäschel et al. (2016), depicted in Figure 1.17, in which premise and hypothesis are concatenated to be forwarded into RNN, then word-by-word attention decides which words in Premise match well to the words in Hypothesis. Such kind of inspiration was also explored in machine translation community (Bahdanau et al., 2015) – attention is employed to determine which phrases in source language are more important for the current phrase generation in target language side.

In our publications in Chapters 5-7, we will have three aspects of contributions about attention mechanism:

1. Based on the study of answer selection task in Figure 1.15 and textual entailment task in Figure 1.16, we realize that it is feasible and reasonable to match cross-sentence phrases locally. For example, we can match "outdoors" and "in a yard" without considering their context. This match is already a strong indicator. Most attention mechanism are implemented based on RNN system which models long-range context representation rather than a local representation. CNN, instead, can model the representation of local regions by sliding filters. Hence, we proposed the first attention mechanism in CNN architectures for natural language processing, in Chapter 5.

2. A further study of the attention distribution in AS task in Figure 1.15 and TE task in Figure 1.16 demonstrates that AS benefits more if we focus on strongly aligned phrases, TE instead prefers to compare those weakly-aligned phrases. Motivated, in our publication described in Chapter 6, we have different attentive pooling in CNN systems. More specifically, we proposed a k-min-max-pooling for TE task so that phrases that are badly aligned can contribute more to the sentence representations; we proposed a k-max-max-pooling for AS so that phrases that are well aligned can act a dominant role in the final sentence representations. These kind of refined sentence representations are supposed to provide more informative information for the identification task.

3. In some applications, a sentence in the pair may be a structured one. For example, in factoid question answering, a single-relation question of raw text

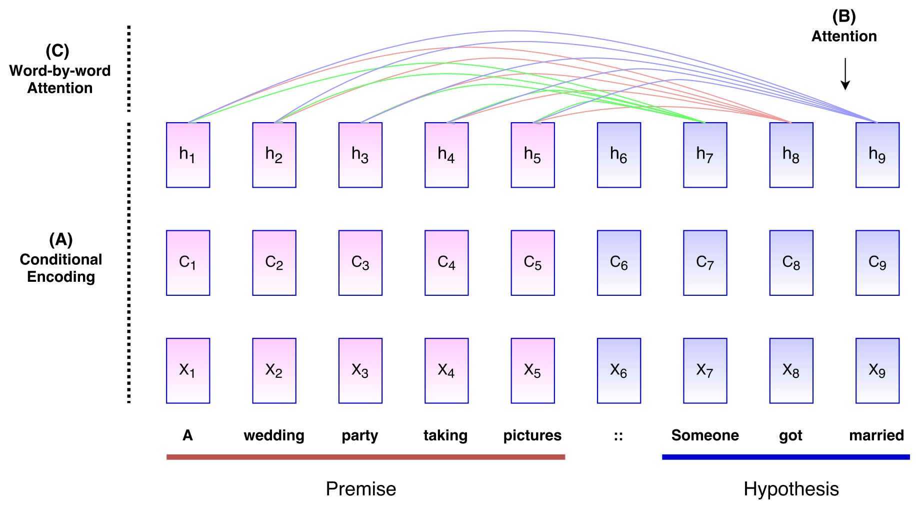

(B) Attention
(C) Word-by-word Attention
(A) Conditional Encoding
h1 h2 h3 h4 h5 h6 h7 h8 h9
c1 c2 c3 c4 c5 c6 c7 c8 c9
x1 x2 x3 x4 x5 x6 x7 x8 x9
A wedding party taking pictures :: Someone got married
Premise
Hypothesis

Figure 1.17 – *Attention example (Rocktäschel et al., 2016)*

needs to match with lots of one-hop facts in knowledge graph (KG). For instance, question "who is the president of U.S?" can match fact (United_States, President_of_Country,?). So, in this kind of sentential relation identification, one sentence can be highly-structured. The publication in Chapter 7 tries to solve the single-relation question answering problem by a two-part CNN. One character-level CNN matches the KG entity "United_States" with topical entity "U.S.", and another word-level CNN matches the KG relation "President_of_Country" with the question pattern "who is the president of <e>". Observing that KG relations are mostly the paraphrase or keyphrases of raw questions, we come up with a novel attentive max-pooling so that the n-gram in question which matches KG relations well can have higher probability to be selected by the max-pooling operation.

1.5.4 Sentential Relation Identification in Background

Above subsections introduced some sentential relation identification tasks in which only two target sentences are involved in the decision. In some scenarios, we can only determine the relationship of two sentences by the background knowledge. Considering the machine comprehension task in Figure 1.11, given a piece of passage and some questions for this passage, multiple answer candidates are provided to each question, a system is required to figure out which answer candidate is the correct one. In this application, passage knowledge is necessary to discover correctly the relationship between the question and each answer candidate. The same case can also happen in visual question answering in which the question and answer candidates are describing something about images, videos etc.

The publication in Chapter 8 introduces two paradigms to handle this task:

- Treating the passage background as a “translation” medium, which *translates* the question into the answer. So, the basic idea is to learn spaces for question, passage and answer candidates separately, finally use passage space to project question space into answer space;

- Treating the whole task as textual entailment – combining the passage and question as a big Premise and those answer candidates as Hypotheses, then our textual entailment approaches mentioned in above subsection can be employed to solve this task.

In this scenario, we have to handle another new challenge that does not exist in the conventional TE task – one sentence is too long (passage plus question), so the approaches for two simple sentences entailment can not work well in this case. Considering again the example in Figure 1.11, only the blue and red sentences are directly related to the question answering problem. Other sentences can be neglected. To this end, we proposed an attentive pooling at both word level and sentence level so that the most informative part of the passage can be refined to guide the task learning.

1.6 Summary

In this chapter, we first introduced some basic knowledge of deep neural networks, including feedforward phase, error backpropagation and gradient descent optimization etc; then we elaborated the sentence representation learning in both unsupervised and supervised training schemes; finally, sentence representation learning was extended to identify sentential relations in three novelties: employing sentence interaction, developing attention mechanisms and sentential relation identification in background.

In the following Chapters 2-8, each one describes one publication in this research topic. Specifically, Chapter 2 explores representation learning of phrases for sentential relation identification, Chapter 3 shows the first CNN for paraphrase identification task, Chapter 4 extends the model in Chapter 3 and tests in more tasks, Chapter 5 demonstrates the first attention mechanism in CNN for sentential relation identification, Chapter 6 develops the attention mechanism in Chapter 5 more fine-grained, so that different sentential relation identification tasks, such as answer selection and textual entailment, have more appropriate attention mechanisms, Chapter 7 develops attentive CNN for a sentential relation identification problem in which one sentence in the pair can be structured, such as a fact in knowledge graphs, Chapter 8 illustrates how the sentential relations are detected in background knowledge, such as a passage.

Chapter 2

Discriminative Phrase Embedding for Paraphrase Identification

Discriminative Phrase Embedding for Paraphrase Identification

Wenpeng Yin and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
wenpeng@cis.lmu.de

Abstract

This work, concerning paraphrase identification task, on one hand contributes to expanding deep learning embeddings to include continuous and discontinuous linguistic phrases. On the other hand, it comes up with a new scheme TF-KLD-KNN to learn the discriminative weights of words and phrases specific to paraphrase task, so that a weighted sum of embeddings can represent sentences more effectively. Based on these two innovations we get competitive state-of-the-art performance on paraphrase identification.

1 Introduction

This work investigates representation learning via deep learning in paraphrase identification task, which aims to determine whether two sentences have the same meaning. One main innovation of deep learning is that it learns distributed word representations (also called "word embeddings") to deal with various Natural Language Processing (NLP) tasks. Our goal is to use and refine embeddings to get competitive performance.

We adopt a supervised classification approach to paraphrase identification like most top performing systems. Our focus is representation learning of sentences. Following prior work (e.g., Blacoe and Lapata (2012)), we compute the vector of a sentence as the sum of the vectors of its components. But unlike prior work we use *single words, continuous phrases and discontinuous phrases* as the components, not just single words. Our rationale is that many semantic units are formed by multiple words – e.g., the continuous phrase "side effects" and the discontinuous phrase "pick ... off". The better we can discover and represent such components, the better the compositional sentence vector should be. We use the term *unit* to refer to single words, continuous phrases and discontinuous phrases.

Ji and Eisenstein (2013) show that not all words are equally important for paraphrase identification. They propose TF-KLD, a discriminative weighting scheme to address this problem. While they do not represent sentences as vectors composed of other vectors, TF-KLD is promising for a vector-based approach as well since the insight that units are of different importance still applies. A shortcoming of TF-KLD is its failure to define weights for words that do not occur in the training set. We propose TF-KLD-KNN, an extension of TF-KLD that computes the weight of an unknown unit as the average of the weights of its k nearest neighbors. We determine nearest neighbors by cosine measure over embedding space. We then represent a sentence as the sum of the vectors of its units, weighted by TF-KLD-KNN.

We use (Madnani et al., 2012) as our baseline system. They used simple features – eight different machine translation metrics – yet got good performance. Based on above new sentence representations, we compute three kinds of features to describe a pair of sentences – cosine similarity, element-wise sum and absolute element-wise difference – and show that combining them with the features from Madnani et al. (2012) gets state-of-the-art performance on the Microsoft Research Paraphrase (MSRP) corpus (Dolan et al., 2004).

In summary, our first contribution lies in embedding learning of continuous and discontinuous phrases. Our second contribution is the weighting scheme TF-KLD-KNN.

This paper is structured as follows. Section 2 reviews related work. Section 3 describes our method for learning embeddings of units. Section 4 introduces a measure of unit discriminativity that can be used for differential weighting of units. Section 5 presents experimental setup and results. Section 6 concludes.

2 Related work

The key for good performance in paraphrase identification is the design of good features. We now discuss relevant prior work based on the linguistic granularity of feature learning.

The first line is compositional semantics, which learns representations for words and then composes them to representations of sentences. Blacoe and Lapata (2012) carried out a comparative study of three word representation methods (the simple distributional semantic space (Mitchell and Lapata, 2010), distributional memory tensor (Baroni and Lenci, 2010) and word embedding (Collobert and Weston, 2008)), along with three composition methods (addition, point-wise multiplication, and recursive autoencoder (Socher et al., 2011)). They showed that addition over word embeddings is competitive, despite its simplicity.

The second category directly seeks sentence-level features. Ji and Eisenstein (2013) explored unigrams, bigrams and dependency pairs as sentence features. They proposed TF-KLD to weight features and used non-negative factorization to learn latent sentence representations. Our method TF-KLD-KNN is an extension of their work.

The third line directly computes features for sentence pairs. Wan et al. (2006) used N-gram overlap, dependency relation overlap, dependency tree-edit distance and difference of sentence lengths. Finch et al. (2005) and Madnani et al. (2012) combined several machine translation metrics. Das and Smith (2009) presented a generative model over two sentences' dependency trees, incorporating syntax, lexical semantics, and hidden loose alignments between the trees to model generating a paraphrase of a given sentence. Socher et al. (2011) used recursive autoencoders to learn representations for words and word sequences on each layer of the sentence parsing tree, and then proposed dynamic pooling layer to form a fixed-size matrix as the representation of the two sentences. Other work representative of this line is by Kozareva and Montoyo (2006), Qiu et al. (2006), Ul-Qayyum and Altaf (2012).

Our work, first learning unit embeddings, then adding them to form sentence representations, finally calculating pair features (cosine similarity, absolute difference and MT metrics) actually is a combination of above three lines.

3 Embedding learning for units

As explained in Section 1, "units" in this work include single words, continuous phrases and discontinuous phrases. Phrases have a larger linguistic granularity than words and thus will in general contain more meaning aspects for a sentence. For example, successful detection of continuous phrase "side effects" and discontinuous phrase "pick $\cdots$ off" is helpful to understand the sentence meaning correctly. This section focuses on how to detect phrases and how to represent them.

3.1 Phrase collection

Phrases defined by a lexicon have not been investigated extensively before in deep learning. To collect canonical phrase set, we extract two-word phrases defined in Wiktionary[1] and Wordnet (Miller and Fellbaum, 1998) to form a collection of size 95,218. This collection contains *continuous phrases* – phrases whose parts always occur next to each other (e.g., "side effects") – and *discontinuous phrases* – phrases whose parts more often occur separated from each other (e.g., "pick … off").

3.2 Identification of phrase continuity

Wiktionary and WordNet do not categorize phrases as continuous or discontinuous. So we need a heuristic to determine this automatically.

For each phrase "A_B", we compute [c_1, c_2, c_3, c_4, c_5] where $c_i, 1 \leq i \leq 5$, indicates there are c_i occurrences of A and B in that order with a distance

[1]`http://en.wiktionary.org`

of i. We compute these statistics for a corpus consisting of English Gigaword (Graff et al., 2003) and Wikipedia. We set the maximal distance to 5 because discontinuous phrases are rarely separated by more than 5 tokens.

If c_1 is 10 times higher than $(c_2+c_3+c_4+c_5)/4$, we classify "A_B" as *continuous*, otherwise as *discontinuous*. For example, [$c_1, \ldots, c_5$] is [1121, 632, 337, 348, 4052] for "pick_off", so c_1 is smaller than the average 1342.25 and "pick_off" is set as "discontinuous"; [$c_1, \ldots, c_5$] is [14831, 16, 177, 331, 3471] for "Cornell University", c_1 is 10 times larger than the average and this phrase is set to "continuous".

We found that that this heuristic for distinguishing between continuous and discontinuous phrases works well and leave the development of a more principled method for future work.

3.3 Sentence reformatting

Sentence "... A ... B ..." is

- reformatted as "... A_B ..." if A and B form a continuous phrase and no word intervenes between them and
- reformatted as "... A_B ... A_B ..." if A and B form a discontinuous phrase and are separated by 1 to 4 words. We replace each of the two component words with A_B to make the context of both constituents available to the phrase in learning.

This method of phrase detection will generate some false positives, e.g., if "pick" and "off" occur in a context like "she picked an island off the coast of Maine". However, our experimental results indicate that it is robust enough for our purposes.

We run word2vec (Mikolov et al., 2013) on the reformatted Wikipedia corpus to learn embeddings for all units. Embedding size is set to 200.

4 Measure of unit discriminativity

We will represent a sentence as the sum of the embeddings of its units. Building on Ji and Eisenstein (2013)'s TF-KLD, we want to weight units according to their ability to discriminate two sentences specific to the paraphrase task.

TF-KLD assumes a training set of sentence pairs in the form $\langle u_i, v_i, t_i \rangle$, where u_i and v_i denote the binary unit occurrence vectors for the sentences in the ith pair and $t_i \in \{0, 1\}$ is the gold tag. Then, we define p_k and q_k as follows.

- $p_k = P(u_{ik}|v_{ik} = 1, t_i = 1)$. This is the probability that unit w_k occurs in sentence u_i given that w_k occurs in its counterpart v_i and they are paraphrases.
- $q_k = P(u_{ik}|v_{ik} = 1, t_i = 0)$. This is the probability that unit w_k occurs in sentence u_i given that w_k occurs in its counterpart v_i and they are not paraphrases.

TF-KLD computes the discriminativity of unit w_k as the Kullback-Leibler divergence of the Bernoulli distributions (p_k, 1-p_k) and (q_k, 1-q_k)

TF-KLD has a serious shortcoming for unknown units. Unfortunately, the test data of the commonly used MSPR corpus in paraphrase task has about 6% unknown words and 62.5% of its sentences contain unknown words. It motivates us to design an improved scheme TF-KLD-KNN to reweight the features.

TF-KLD-KNN weights are the same as TF-KLD weights for known units. For a unit that did not occur in training, TF-KLD-KNN computes its weight as the average of the weights of its k nearest neighbors in embedding space, where unit similarity is calculated by cosine measure.[2]

Word2vec learns word embeddings based on the word context. The intuition of TF-KLD-KNN is that words with similar context have similar discriminativities. This enables us to transfer the weights of features in training data to the unknown features in test data, greatly helping to address problems of sparseness.

5 Experiments

5.1 Data and baselines

We use the MSRP corpus (Dolan et al., 2004) for evaluation. It consists of a training set of 2753 true paraphrase pairs and 1323 false paraphrase pairs and a test set of 1147 true and 578 false pairs.

[2]Unknown words without embeddings (only seven cases in our experiments) are ignored. This problem can be effectively relieved by training embedding on larger corpora.

For our new method, it is interesting to measure the improvement on the subset of those MSRP sentences that contain at least one phrase. In the standard MSRP corpus, 3027 training pairs (2123 true, 904 false) and 1273 test pairs (871 true, 402 false) contain phrases; we denote this subset as *subset*. We carry out experiments on *overall* (all MSRP sentences) as well as *subset* cases.

We compare six methods for paraphrase identification.

- **NOWEIGHT.** Following Blacoe and Lapata (2012), we simply represent a sentence as the unweighted sum of the embeddings of all its units.

- **MT** is the method proposed by Madnani et al. (2012): the sentence pair is represented as a vector of eight different machine translation metrics.

- **Ji and Eisenstein (2013)**. We reimplemented their "inductive" setup which is based on matrix factorization and is the top-performing system in paraphrasing task.[3]

  The following three methods not only use this vector of eight MT metrics, but use three kinds of additional features given two sentence representations s_1 and s_2: cosine similarity, element-wise sum s_1+s_2 and element-wise absolute difference $|s_1 - s_2|$. We now describe how each of the three methods computes the sentence vectors.

- **WORD.** The sentence is represented as the sum of all single-word embeddings, weighted by TF-KLD-KNN.

- **WORD+PHRASE.** The sentence is represented as the sum of the embeddings of all its units (including phrases), weighted by TF-KLD-KNN.

- **WORD+GOOGLE.** Mikolov et al. (2013) use a data-driven method to detect statistical phrases which are mostly continuous bigrams. We implement their system by first exploiting word2phrase[4] to reformat Wikipedia, then using word2vec skip-gram model to train phrase embeddings.

We use the same weighting scheme TF-KLD-KNN for the three weighted sum approaches: WORD, WORD+PHRASE and WORD+GOOGLE. Note however that there is an interaction between representation space and nearest neighbor search. We limit the neighbor range of unknown words for WORD to single words; in contrast, we search the space of all single words and linguistic (resp. Google) phrases for WORD+PHRASE (resp. WORD+GOOGLE).

We use LIBLINEAR (Fan et al., 2008) as our linear SVM implementation. 20% training data is used as development data. Parameter k is fine-tuned on development set and the best value 3 is finally used in following reported results.

5.2 Experimental results

Table 1 shows performance for the six methods as well as for the majority baseline. In the *overall* (resp. *subset*) setup, WORD+PHRASE performs best and outperforms (Ji and Eisenstein, 2013) by .009 (resp. .052) on accuracy. Interestingly, Ji and Eisenstein (2013)'s method obtains worse performance on $subset$. This can be explained by the effect of matrix factorization in their work: it works less well for smaller datasets like $subset$. This is a shortcoming of their approach. WORD+GOOGLE has a slightly worse performance than WORD+PHRASE; this suggests that linguistic phrases might be more effective than statistical phrases in identifying paraphrases.

Cases *overall* and *subset* both suggest that phrase embeddings improve sentence representations. The accuracy of WORD+PHRASE is lower on *overall* than on *subset* because WORD+PHRASE has no advantage over WORD for sentences without phrases.

5.3 Effectiveness of TF-KLD-KNN

The key contribution of TF-KLD-KNN is that it achieves full coverage of feature weights in the face of data sparseness. We now compare four weighting methods on overall corpus and with the combi-

[3]They report even better performance in a "transductive" setup that makes use of test data. We only address paraphrase identification for the case that the test data are not available for training the model in this paper.

[4]`https://code.google.com/p/word2vec/`

	overall		subset	
method	acc	F_1	acc	F_1
baseline	.665	.799	.684	.812
NOWEIGHT	.708	.809	.713	.823
MT	.774	.841	.772	.839
Ji and Eisenstein (2013)	.778	.843	.749	.827
WORD	.775	.839	.776	.843
WORD+GOOGLE	.780	.843	.795	.853
WORD+PHRASE	**.787**	**.848***	**.801**	**.857***

Table 1: Results on overall and subset corpus. Significant improvements over MT are marked with $*$ (approximate randomization test, Padó (2006), $p < .05$).

method	acc	F_1
NOWEIGHT	.746	.815
TF-IDF	.752	.821
TF-KLD	.774	.842
TF-KLD-KNN	**.787**	**.848**

Table 2: Effects of different reweighting methods on overall.

nation of MT features: NOWEIGHT, TF-IDF, TF-KLD, TF-KLD

Table 2 suggests that task-specific reweighting approaches (including TF-KLD and TF-KLD-KNN) are superior to unspecific schemes (NOWEIGHT and TF-IDF). Also, it demonstrates the effectiveness of our weight learning solution for unknown units in paraphrase task.

5.4 Reweighting schemes for unseen units

We compare our reweighting scheme **KNN** (i.e., TF-KLD-KNN) with three other reweighting schemes. **Zero**: zero weight, i.e., ignore unseen units; **Type-average**: take the average of weights of all known unit types in test set; **Context-average**: average of the weights of the adjacent known units of the unknown unit (two, one or defaulting to Zero, depending on how many there are). Figure 1 shows that KNN performs best.

6 Conclusion

This work introduced TF-KLD-KNN, a new reweighting scheme that learns the discriminativities of known as well as unknown units effectively. We further improved paraphrase identification per-

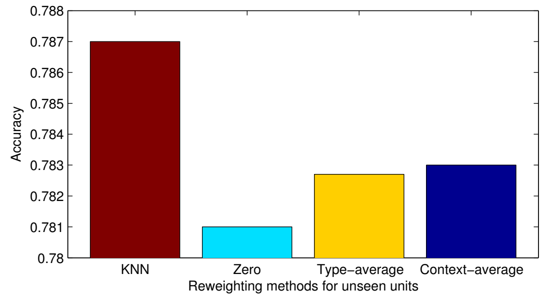


Figure 1: Performance of different reweighting schemes for unseen units on overall.

formance by the utilization of continuous and discontinuous phrase embeddings.

In future, we plan to do experiments in a cross-domain setup and enhance our algorithm for domain adaptation paraphrase identification.

Acknowledgments

We are grateful to members of CIS for comments on earlier versions of this paper. This work was supported by Baidu (through a Baidu scholarship awarded to Wenpeng Yin) and by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/8-2, SPP 1335).

References

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

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

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Pro-*

cessing of the AFNLP: Volume 1-Volume 1, pages 468–476. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 350–356. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Andrew Finch, Young-Sook Hwang, and Eiichiro Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 17–24.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zornitsa Kozareva and Andrés Montoyo. 2006. Paraphrase identification on the basis of supervised machine learning techniques. In *Advances in natural language processing*, pages 524–533. Springer.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

George Miller and Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26. Association for Computational Linguistics.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, volume 24, pages 801–809.

Zia Ul-Qayyum and Wasif Altaf. 2012. Paraphrase identification using semantic heuristic features. *Research Journal of Applied Sciences, Engineering and Technology*, 4(22):4894–4904.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006, pages 131–138.

Chapter 3

Convolutional Neural Network for Paraphrase Identification

Convolutional Neural Network for Paraphrase Identification

Wenpeng Yin and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
wenpeng@cis.uni-muenchen.de

Abstract

We present a new deep learning architecture Bi-CNN-MI for paraphrase identification (PI). Based on the insight that PI requires comparing two sentences *on multiple levels of granularity*, we learn multigranular sentence representations using convolutional neural network (CNN) and model interaction features at each level. These features are then the input to a logistic classifier for PI. All parameters of the model (for embeddings, convolution and classification) are directly optimized for PI. To address the lack of training data, we pretrain the network in a novel way using a language modeling task. Results on the MSRP corpus surpass that of previous NN competitors.

1 Introduction

In this paper, we address the problem of paraphrase identification. It is usually formalized as a binary classification task: for two sentences (S_1, S_2), determine whether they roughly have the same meaning.

Inspired by recent successes of deep neural networks (NNs) in fields like computer vision (Neverova et al., 2014), speech recognition (Deng et al., 2013) and natural language processing (Collobert and Weston, 2008), we adopt a deep learning approach to paraphrase identification in this paper.

The key observation that motivates our NN architecture is that the identification of a paraphrase relationship between S_1 and S_2 requires an analysis *at multiple levels of granularity.*

(A1) "Detroit manufacturers have raised vehicle prices by ten percent." – (A2) "GM, Ford and Chrysler have raised car prices by five percent."

Example A1/A2 shows that paraphrase identification requires comparison *at the word level.* A1 cannot be a paraphrase of A2 because the numbers "ten" and "five" are different.

(B1) "Mary gave birth to a son in 2000." – (B2) "He is 14 years old and his mother is Mary."

PI for B1/B2 can only succeed *at the sentence level* since B1/B2 express the same meaning using very different means.

Most work on paraphrase identification has focused on only one level of granularity: either on low-level features (e.g., Madnani et al. (2012)) or on the sentence level (e.g., ARC-I, Hu et al. (2014)).

An exception is the RAE model (Socher et al., 2011). It computes representations on all levels of a parse tree: each node – including nodes corresponding to words, phrases and the entire sentence – is represented as a vector. RAE then computes a $n_1 \times n_2$ comparison matrix of the two trees derived from S_1 and S_2 respectively, where n_1, n_2 are the number of nodes and each comparison is the Euclidean distance between two vectors. This is then the basis for paraphrase classification.

RAE (Socher et al., 2011) is one of three prior NN architectures that we draw on to design our system. It embodies the key insight that paraphrase identification involves analysis of information at multiple levels of granularity. However, relying on parsing has limitations for noisy text and for other applications in which highly accurate parsers are not available. We extend the basic idea of RAE by exploring stacked convolution layers which on one hand use sliding windows to split sentences into flexible phrases, furthermore, higher layers are able to extract more abstract features of longer-range phrases by combining phrases in lower layers.

A representative way of doing this in deep learning is the work by Kalchbrenner et al. (2014), the second prior NN architecture that we draw on. They use convolution to learn representations at multiple levels (Collobert and Weston, 2008). The motivation for convolution is that natural language consists of long sequences in which many short subsequences contribute in a stable way to the structure and meaning of the long sequence *regardless of the position of the subsequence within the long sequence*. Thus, it is advantageous to learn convolutional filters that detect a particular feature regardless of position. Kalchbrenner et al. (2014)'s architecture extends this idea in two important ways. First, k-max pooling extracts the k top values from a sequence of convolutional filter applications and guarantees a fixed length output. Second, they stack several levels of convolutional filters, thus achieving multigranularity. We incorporate this architecture as the part that analyzes an individual sentence.

The third prior NN architecture we draw on is ARC proposed by Hu et al. (2014) who also attempt to exploit convolution for paraphrase identification. Their key insight is that *we want to be able to directly optimize the entire system for the task we are addressing,* i.e., for paraphrase identification. Hu et al. (2014) do this by adopting a Siamese architecture: their NN consists of two shared-weight sentence analysis NNs that feed into a binary classifier that is directly trained on labeled sentence pairs. As we will show below, this is superior to separating the two steps: first learning sentence representations, then training binary classification for fixed, learned sentence representations as Bromley et al. (1993), Socher et al. (2011) and many others do.

We can now give an overview of our NN architecture (Figure 1). We call it Bi-CNN-MI: "Bi-CNN" stands for double CNNs used in Siamese framework, "MI" for *multigranular interaction* features. Bi-CNN-MI has three parts: (i) the sentence analysis network CNN-SM, (ii) the sentence interaction model CNN-IM and (iii) a logistic regression on top of the network that performs paraphrase identification. We now describe these three parts in detail.

(i) Following Kalchbrenner et al. (2014), we design CNN-SM, a convolutional sentence analysis NN that computes representations at four different levels: word, short ngram, long ngram and sentence. This multigranularity is important because paraphrase identification benefits from analyzing sentences at multiple levels.

(ii) Following Socher et al. (2011), CNN-IM, the interaction model, computes interaction features as $s_1 \times s_2$ matrices, where s_i is the number of items of a certain granularity in S_i. In contrast to Socher et al. (2011), CNN-IM computes these features at fixed levels and only for comparable units; e.g., we do not compare single words with entire sentences.

(iii) Following Hu et al. (2014), we integrate two copies of CNN-SM into a Siamese architecture that allows to optimize all parameters of the NN for paraphrase identification. In our case, these parameters include parameters for word embedding, for convolution filters, and for the classification of paraphrase candidate pairs. In contrast to Hu et al. (2014), the inputs to the final paraphrase candidate pair classification layer are *interaction feature matrices at multiple levels* – as opposed to *single-level features* that do not directly *compare an element of S_1 with a potentially corresponding element of S_2*.

There is one other problem we have to address to get good performance. Training sets for paraphrase identification are small in comparison with the high complexity of the task. Training a complex network like Bi-CNN-MI with a large number of parameters on a small training set is not promising due to sparseness and likely overfitting.

In order to make full use of the training data, we propose a new unsupervised training scheme CNN-LM (CNN Language Model) to pretrain the largest part of the model, the sentence analysis network CNN-SM. The key innovation is that we use a language modeling task in a setup similar to autoencoding for pretraining (see below for details). This means that embedding and convolutional parameters can be pretrained on very large corpora since no human labels are required for pretraining.

We will show below that this pretraining is critical for getting good performance in the paraphrase task. However, the general design principle of this type of unsupervised pretraining should be widely applicable given that next-word prediction training is possible in many NLP applications. Thus, this new way of unsupervised pretraining could be an important contribution of the paper independent of paraphrase identification.

Section 2 discusses related work. Sections 3 and 4 introduce the sentence model CNN-SM and the sentence interaction model CNN-IM. Section 5 describes the training regime. The experiments are presented in Section 6. Section 7 concludes.

2 Related work

Bi-CNN-MI is closely related to NN models for sentence representations and for text matching.

A pioneering work using CNN to model sentences is (Collobert and Weston, 2008). They conducted convolutions on sliding windows of a sentence and finally use max pooling to form a sentence representation. Kalchbrenner et al. (2014) introduce k-max pooling and stacking of several CNNs as discussed in Section 1.

Lu and Li (2013) developed a deep NN to match short texts, where interactions between components within the two objects were considered. These interactions were obtained via LDA (Blei et al., 2003). A two-dimensional interaction space is formed, then those local decisions will be sent to the corresponding neurons in upper layers to get rounds of fusion, finally logistic regression in the output layer produces the final matching score. Drawbacks of this approach are that LDA parameters are not optimized for the paraphrase task and that the interactions are formed on the level of single words only.

Gao et al. (2014) model *interestingness* between two documents with deep NNs. They map source-target document pairs to feature vectors in a latent space in such a way that the distance between the source document and its corresponding interesting target in that space is minimized. Interestingness is more like topic relevance, based mainly on the aggregate meaning of lots of keywords. Additionally, their model is a document-level model and is not multigranular.

Madnani et al. (2012) treated paraphrase relationship as a kind of mutual translation, hence combined eight kinds of machine translation metrics including BLEU (Papineni et al., 2002), NIST (Doddington, 2002), TER (Snover et al., 2006), TERp (Snover et al., 2009), METEOR (Denkowski and Lavie, 2010), SEPIA (Habash and Elkholy, 2008), BADGER (Parker, 2008) and MAXSIM (Chan and Ng, 2008). These features are not multigranular; rather they are low-level only; high-level features – e.g., a representation of the entire sentence – are not considered.

Bach et al. (2014) claimed that elementary discourse units obtained by segmenting sentences play an important role in paraphrasing. Their conclusion also endorses Socher et al. (2011)'s and our work, for both take similarities between component phrases into account.

We discussed Socher et al. (2011)'s RAE and Hu et al. (2014)'s ARC-I in Section 1. In addition to similarity matrices there are two other important aspects of (Socher et al., 2011). First, the similarity matrices are converted to a fixed size feature vector by *dynamic pooling*. We adopt this approach in Bi-CNN-MI; see Section 4.2 for details.

Second, (Socher et al., 2011) is partially based on parsing as is some other work on paraphrase identification (e.g., Wan et al. (2006), Ji and Eisenstein (2013)). Parsing is a potentially powerful tool for identifying the important meaning units of a sentence, which can then be the basis for determining meaning equivalence. However, reliance on parsing makes these approaches less flexible. For example, there are no high-quality parsers available for some domains and some languages. Our approach is in principle applicable for any domain and language. It is also unclear how we would identify comparable units in the parse trees of S_1 and S_2 if the parse trees have different height and the sentences different lengths. A key property of Bi-CNN-MI is that it is designed to produce units at fixed levels and only units at the same level are compared with each other.

3 Convolution sentence model CNN-SM

Our network Bi-CNN-MI for paraphrase detection (Figure 1) consists of four parts. On the left and on the right there are two multilayer NNs with seven layers, from "initialized word embeddings: sentence 1/2" to "k-max pooling". The weights of these two NNs are shared. This part of Bi-CNN-MI is based on (Kalchbrenner et al., 2014) and we refer to it as convolutional sentence model CNN-SM.

Between the two CNN-SMs there is the interaction model CNN-IM, consisting of four feature ma-

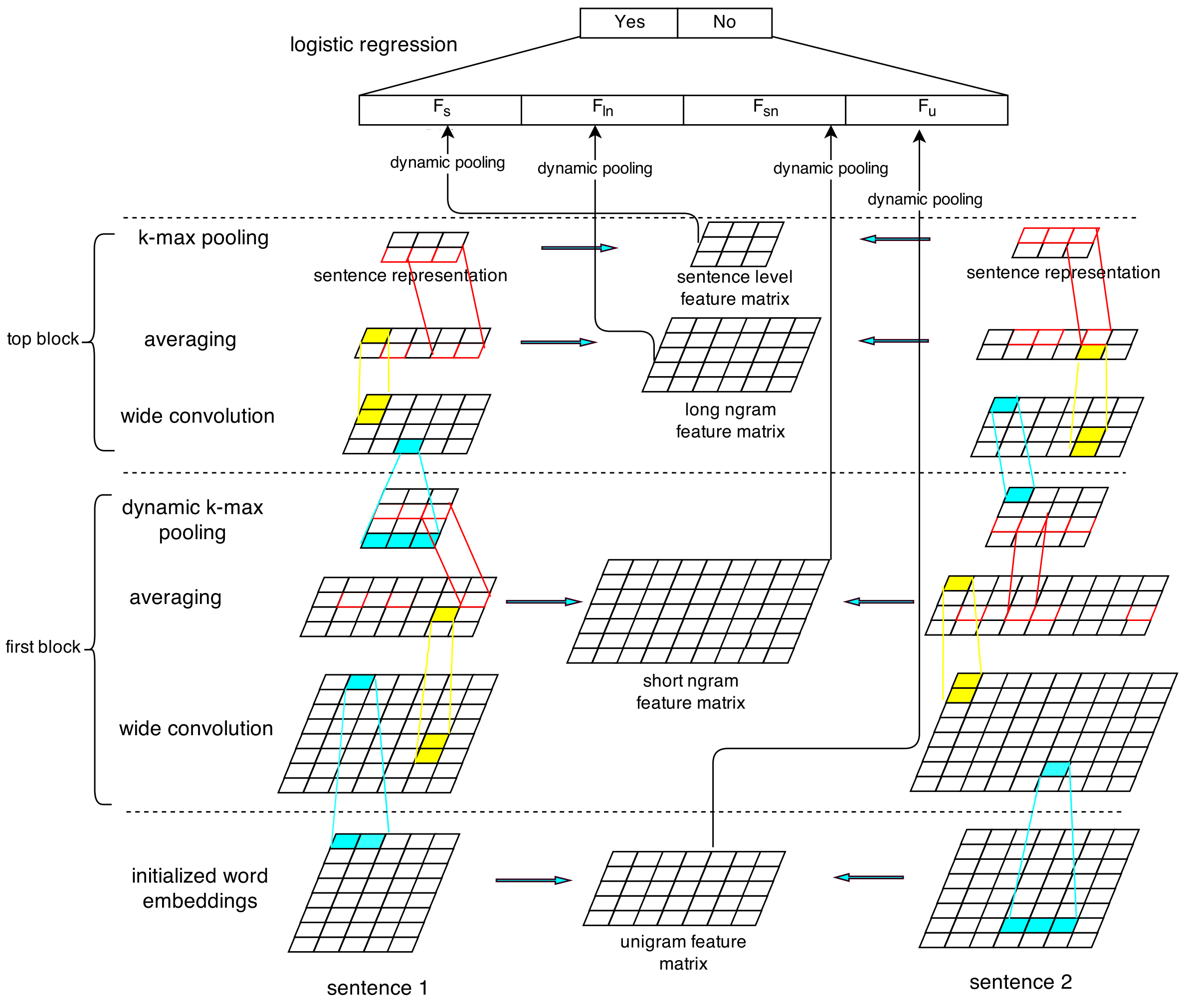


Figure 1: The paraphrase identification architecture Bi-CNN-MI

trices (unigram, short ngram, long ngram, sentence). CNN-IM feeds into a logistic classifier that performs paraphrase detection. See Sections 4 and 5 for these two parts of Bi-CNN-MI.

3.1 Wide convolution

We use Kalchbrenner et al. (2014)'s **wide one-dimensional convolution**. Denoting the number of tokens of S_i as $|S_i|$, we convolve weight matrix $\mathbf{M} \in \mathbb{R}^{d\times m}$ over sentence representation matrix $\mathbf{S} \in \mathbb{R}^{d\times|S_i|}$ and generate a matrix $\mathbf{C} \in \mathbb{R}^{d\times(|S_i|+m-1)}$ each column of which is the representation of an m-gram. d is the dimension of word (and also ngram) embeddings. m is filter width.

Our motivation for using convolution is that after training, a convolutional filter corresponds to a feature detector that learns to recognize a class of m-grams that is useful for paraphrase detection.

3.2 Averaging

After convolution, to build simple relations across rows, each odd row and the row behind immediately are *averaged*, generating matrix $\mathbf{A} \in \mathbb{R}^{\frac{d}{2}\times(|S_i|+m-1)}$. Namely:

$$\mathbf{A} = (\mathbf{C}_{\text{odd}} + \mathbf{C}_{\text{even}})/2 \qquad (1)$$

where $\mathbf{C}_{\text{odd}}$, $\mathbf{C}_{\text{even}}$ denote the odd and even rows of $\mathbf{C}$, respectively. Finally, this convolution layer will output matrix $\mathbf{B}$ whose jth column is defined thus:

$$\mathbf{B}_{:,j} = \tanh(\mathbf{A}_{:,j} + \mathbf{b}^T) \quad 0 \leq j < (|S_i| + m - 1) \qquad (2)$$

$\mathbf{b}$ is a bias vector with dimension $d/2$, same for each column.

3.3 Dynamic k-max pooling

We use Kalchbrenner et al. (2014)'s **dynamic k-max pooling** to extract features for variable-length sentences. It extracts k_{dy} top values from each dimension after the first layer of averaging and $k_{\text{top}} = 4$ top values after the top layer of averaging. We set

$$k_{\text{dy}} = \max(k_{\text{top}}, |S_i|/2 + 1) \qquad (3)$$

Thus, k_{dy} depends on the length of S_i.

The sequence of layers in (Kalchbrenner et al., 2014) is convolution, folding, k-max pooling, $\tanh$. We experimented with this sequence and found that after k-max pooling many $\tanh$ units had an input close to 1, in the nondynamic range of the function (since the input is the addition of several values). This makes learning difficult. We therefore changed the sequence to convolution, averaging, $\tanh$, k-max pooling. This makes it less likely that $\tanh$ units will be saturated.

We have described convolution, averaging and k-max pooling. We can stack several blocks of these three layers to form deep architectures, as the two blocks (marked "first block" and "top block") in Figure 1.

4 Convolution interaction model CNN-IM

After the introduction in the previous section of the CNN-SM part of our architecture for processing an *individual sentence*, we now turn to the CNN-IM interaction model that computes the four feature matrices in Figure 1 to assess the *interactions between the two sentences*.

4.1 Feature matrices

One key innovation of our approach is multigranularity: we compute similarity between the two paraphrase candidates on multiple levels. Specifically, we compute similarity at four levels in this paper: unigram, short ngram, long ngram and sentence. We use notation $l \in \{\text{u}, \text{sn}, \text{ln}, \text{s}\}$ to refer to the four levels, and use $\mathbf{S}^{i,l}$ to denote the matrix representing sentence S_i at level l. For level l, we compute feature matrices $\hat{\mathbf{F}}_l$ as follows:

$$\hat{\mathbf{F}}_l^{ij} = \exp\left(\frac{-||\mathbf{S}^{1,l}_{:,i} - \mathbf{S}^{2,l}_{:,j}||^2}{2\beta}\right) \qquad (4)$$

where $||\mathbf{S}^{1,l}_{:,i} - \mathbf{S}^{2,l}_{:,j}||^2$ is the Euclidean distance between the representations of the ith item of S_1 and the jth item of S_2 on level l. We set $\beta = 2$ (cf. Wu et al. (2013)).

We do not use cosine because the magnitude of the activations of hidden units is important, not just the overall direction; e.g., if $\mathbf{S}^{1,l}_{:,i}$ and $\mathbf{S}^{2,l}_{:,j}$ point in the same direction, but activations are much larger in $\mathbf{S}^{2,l}_{:,j}$, then the two vectors are very dissimilar.

The lowest level feature matrix (l = u) is the unigram similarity matrix $\hat{\mathbf{F}}_\text{u}$. It has size $|S_1| \times |S_2|$. The feature entry $\hat{\mathbf{F}}_\text{u}{}^{ij}$ is the similarity between the ith word of S_1 and the jth word of S_2 where each word is represented by a d-dimensional word embedding ($d = 100$ in our experiments).

The next level feature matrix is the short ngram similarity matrix $\hat{\mathbf{F}}_{\text{sn}}$. It has size $(|S_1| + m_{\text{sn}} - 1) \times (|S_2| + m_{\text{sn}} - 1)$ where $m_{\text{sn}} = 3$ is the filter width in this convolution layer and $|S_i| + m_{\text{sn}} - 1$ is the number of short ngrams in S_i. The feature entry $\hat{\mathbf{F}}_{\text{sn}}{}^{ij}$ is the similarity between two $d/2$-dimensional vectors representing two short ngrams from S_1 and S_2.

We use multiple feature maps to improve the system performance. Different feature maps are expected to extract different kinds of sentence features, and can be implemented in the same convolution layer in parallel. Specifically, we use $f_{\text{sn}} = 6$ feature maps on this level following Kalchbrenner et al. (2014). Thus, we actually compute six feature matrices $\hat{\mathbf{F}}_{\text{sn},i}$ ($i = 1, \cdots, f_{\text{sn}}$), one for each pair of feature maps that share convolution weights while derived from S_1 and S_2 respectively. (Figure 1 only shows one of those six matrices.)

The next level feature matrix is the long ngram similarity matrix $\hat{\mathbf{F}}_{\text{ln}}$. It has size $(k_{\text{dy},1} + m_{\text{ln}} - 1) \times (k_{\text{dy},2} + m_{\text{ln}} - 1)$ where $k_{\text{dy},i}$ (Equation 3) is the k value in dynamic k-max pooling for sentence i, $k_{\text{dy},i} + m_{\text{ln}} - 1$ is the number of long ngrams in S_i and $m_{\text{ln}} = 5$ is the filter width in this convolution layer. The feature entry $\hat{\mathbf{F}}_{\text{ln}}{}^{ij}$ is the similarity between two $d/4$-dimensional vectors representing two long ngrams from S_1 and S_2.

We use $f_{\text{ln}} = 14$ feature maps on this level following Kalchbrenner et al. (2014). Thus, we compute 14 feature matrices $\hat{\mathbf{F}}_{\text{ln},i}$ ($i = 1, \cdots, f_{\text{ln}}$), in a way analogous to the $f_{\text{sn}} = 6$ feature maps $\hat{\mathbf{F}}_{\text{sn},i}$.

The last feature matrix is the sentence similarity matrix $\hat{\mathbf{F}}_{\text{s}}$. It has size $k_{\text{top}} \times k_{\text{top}}$ where $k_{\text{top}} = 4$ is the parameter in k-max pooling at the last max pooling layer. The feature entry $\hat{\mathbf{F}}_{\text{s}}{}^{ij}$ is the similarity between two $d/4$-dimensional vectors computed by max pooling from S_1 and S_2.

For $l = \text{s}$, there are also $f_{\text{ln}} = 14$ feature matrices $\hat{\mathbf{F}}_{\text{s},i}$ ($i = 1, \cdots, f_{\text{ln}}$), analogous to the $f_{\text{ln}} = 14$ feature matrices $\hat{\mathbf{F}}_{\text{ln},i}$.

A general design principle of the architecture is that we compute each interaction feature matrix between two feature maps that share the same convolution weights. Two feature maps learned with the same filter will contain the same kinds of features derived from the input.

4.2 Dynamic pooling of feature matrices

As sentence lengths vary, feature matrices $\hat{\mathbf{F}}_l$ have different sizes, which makes it impossible to use them directly as input of the last layer.

That means we need to map $\hat{\mathbf{F}}_l \in \mathbb{R}^{r \times c}$ into a matrix $\mathbf{F}_l$ of fixed size $r' \times c'$ ($l \in \{\text{u}, \text{sn}, \text{ln}, \text{s}\}$; r', c' are parameters and are the same for all sentence pairs while r, c depend on $|S_1|$ and $|S_2|$). Dynamic pooling divides $\hat{\mathbf{F}}_l$ into $r' \times c'$ nonoverlapping *(dynamic) pools* and copies the maximum value in each dynamic pool to $\mathbf{F}_l$. Our method is similar to (Socher et al., 2011), but preserves locality better.

$\hat{\mathbf{F}}_l$ can be split into equal regions only if r (resp. c) is divisible by r' (resp. c'). Otherwise, for $r > r'$ and if $r \bmod r' = b$, the dynamic pools in the first $r' - b$ splits each have $\lfloor \frac{r}{r'} \rfloor$ rows while the remaining b splits each have $\lfloor \frac{r}{r'} \rfloor + 1$ rows. In Figure 2, a $r \times c = 4 \times 5$ matrix (left) is split into $r' \times c' = 3 \times 3$ dynamic pools (middle): each row is split into [1, 1, 2] and each column is split into [1, 2, 2].

If $r < r'$, we first repeat all rows until no fewer than r' rows remain. Then first r' rows are kept and split into r' dynamic pools. The same principle applies to the partitioning of columns. In Figure 2 (right), the areas with dashed lines and dotted lines are repeated parts for rows and columns, respectively; each cell is its own dynamic pool.

5 Training

5.1 Supervised training

Dynamic pooling gives us fixed size interaction feature matrices for sentence, ngram and unigram levels. As shown in Figure 1, the concatenation of these features ($\mathbf{F}_{\text{s}}$, $\mathbf{F}_{\text{ln}}$, $\mathbf{F}_{\text{sn}}$ and $\mathbf{F}_{\text{u}}$) is the input to a logistic regression layer for paraphrase classification. We have now described all three parts of Bi-CNN-MI: CNN-SM, CNN-IM and logistic regression.

Bi-CNN-MI with all its parameters – including word embeddings and convolution weights – is trained on MSRP. We initialize embeddings with those provided by Turian et al. (2010)[1] (based on Collobert and Weston (2008)). For layer sn, we have $f_{\text{sn}} = 6$ feature maps and set filter width $m_{\text{sn}} = 3$. For layer ln, we have $f_{\text{ln}} = 14$ feature maps and set filter width $m_{\text{ln}} = 5$ and $k_{\text{top}} = 4$. Dynamic pooling

[1]`metaoptimize.com/projects/wordreprs/`

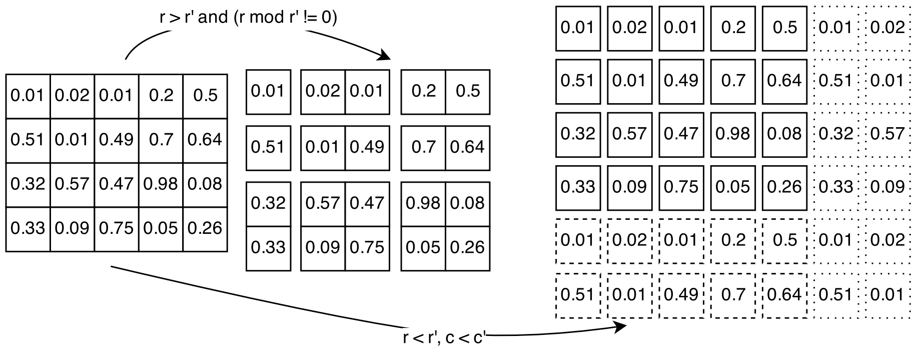


Figure 2: Partition methods in dynamic pooling. Original matrix with size 4×5 is mapped into matrix with size 3×3 and matrix with size 6×7, respectively. Each dynamic pool is distinguished by a border of empty white space around it.

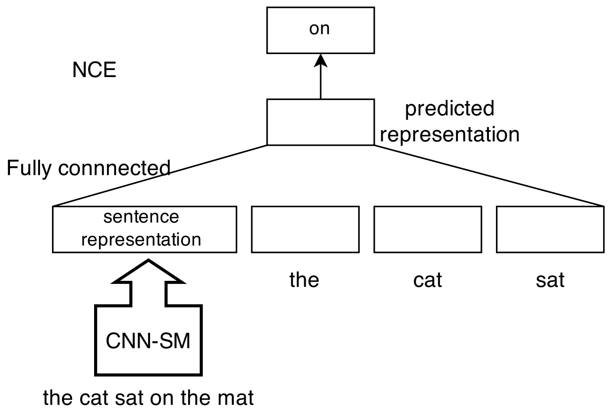


Figure 3: Unsupervised architecture: CNN-LM

sizes are 10×10, 10×10, 6×6, 2×2 for unigram, short ngram, long ngram and sentence, respectively. For training, we employ mini-batch of size 70, L_2 regularization with weight 5×10^{-4} and Adagrad (Duchi et al., 2011).

5.2 Unsupervised pretraining

One of the key contributions of this paper is the architecture CNN-LM shown in Figure 3. CNN-LM is used to pretrain the convolutional filters on unlabeled data. This addresses sparseness and limited training data for paraphrase identification.

The convolution sentence model CNN-SM (Section 3) is part of CNN-LM ("CNN-SM" in Figure 3). The input to CNN-SM is the entire sentence ("the cat sat on the mat"); its output ("sentence representation" in the leftmost rectangle in Figure 3 and the two grids labeled "sentence representation" in the top layer of the top block in Figure 1) is concatenated with a history consisting of the embeddings of the $h = 3$ preceding words ("the", "cat", "sat") as the input of a fully connected layer to generate a predicted representation for the next word ("on"). We employ noise-contrastive estimation (NCE) (Mnih and Teh, 2012; Mnih and Kavukcuoglu, 2013) to compute the cost: the model learns to discriminate between true next words and noise words. NCE allows us to fit unnormalized models making the training time effectively independent of the vocabulary size.

In experiments, CNN-LM is trained on unlabeled MSRP data and an additional 100,000 sentences from English Gigaword (Graff et al., 2003). In principle, sentences from any source, not just English Gigaword, can be used to train this model. In NCE, 20 noise words are sampled for each true example.

So training has two parts: unsupervised, CNN-LM (Figure 3) and supervised, Bi-CNN-MI (Figure 1). In the first phase, the unsupervised training phase, we adopt a language modeling approach because it does not require human labels and can use large corpora to pretrain word embeddings and convolution weights. The goal is to learn sentence features that are unbiased and reflect useful attributes of the input sentence. More importantly, pretraining is useful to relieve overfitting, which is a severe problem when building deep NNs on small corpora like MSRP (cf. Hu et al. (2014)).

In the second phase, the supervised training phase, pretrained word embeddings and convolution weights are tuned for optimal performance on PI.

In CNN-LM, we have combined several architectural elements to pretrain a high-quality sentence analysis NN despite the lack of training data. (i) Similar to PV-DM (Le and Mikolov, 2014), we integrate global context (CNN-SM) and local context (the history of size h) into one model – although our global context consists only of a sentence, not of a paragraph or document. (ii) Similar to work on autoencoding (Vincent et al., 2010), the output that is to be predicted is part of the input. Autoencoding is a successful approach to learning representations and we adapt it here to pretrain good sentence representations. (iii) A second successful approach to learning embeddings is neural network language modeling (Bengio et al., 2003; Mikolov, 2012). Again, we adopt this by including in CNN-LM an ngram language modeling part to predict the next word. The great advantage of this type of embedding learning is that no labels are needed. (iv) CNN-LM only adds one hidden layer over CNN-SM. It keeps simple architecture like PV-DM (Le and Mikolov, 2014), CBOW (Mikolov et al., 2013) and LBL (Mnih and Teh, 2012), enabling the CNN-SM as main training target.

In summary, the key contribution of CNN-LM is that we pretrain convolutional filters. Architectural elements from the literature are combined to support effective pretraining of convolutional filters.

6 Experiments

6.1 Data set and evaluation metrics

We use the Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2004; Das and Smith, 2009). The training set contains 2753 true and 1323 false paraphrase pairs; the test set contains 1147 and 578 pairs, respectively. For each triple (label, S_1, S_2) in the training set we also add (label, S_2, S_1) to make best use of the training data; these additions are nonredundant because the interaction feature matrices (Section 4.1) are asymmetric. Systems are evaluated by accuracy and F_1.

6.2 Paraphrase detection systems

Since we want to show that Bi-CNN-MI performs better than previous NN work, we compare with three NN approaches: NLM, ARC and RAE (Table 1).[2] We also include the majority baseline ("baseline") and MT (Madnani et al., 2012). **RAE** (Socher et al., 2011) and **MT** were discussed in Sections 1 and 2. We now briefly describe the other prior work.

Blacoe and Lapata (2012) compute the vector representation of a sentence from the neural language model (NLM) embeddings (computed based on (Collobert and Weston, 2008)) of the words of the sentence as the sum of the word embeddings (**NLM+**), as the element-wise multiplication of the word embeddings (**NLM**$\odot$), or by means of the recursive autoencoder (**NLM_RAE**, Socher et al. (2011)). The representations of the two paraphrase candidates are then concatenated as input to an SVM classifier. See Blacoe and Lapata (2012) for details.

The ARC model (Hu et al., 2014) is a convolutional architecture similar to (Collobert and Weston, 2008). **ARC-I** is a Siamese architecture in which two shared-weight convolutional sentence models are trained on the binary paraphrase detection task. Hu et al. (2014) find that ARC-I is suboptimal in that it defers the interaction between S_1 and S_2 to the very end of processing: only after the vectors representing S_1 and S_2 have been computed does an interaction occur. To remedy this problem, they propose **ARC-II** in which the Siamese architecture is replaced by a multilayer NN that processes a single representation produced by interleaving S_1 and S_2.

We also evaluate **Bi-CNN-MI–**, an NN identical to Bi-CNN-MI, except that it is not pretrained in unsupervised training.

6.3 Results

Table 1 shows that Bi-CNN-MI outperforms all other systems. The comparison with Bi-CNN-MI– indicates that this is partly due to one major innovation we introduced: unsupervised pretraining. Bi-CNN-MI–, the model without unsupervised pretraining, performs badly. Thus, unsupervised training is helpful to pretrain parameters in paraphrase

[2] A reviewer suggests an additional experiment to directly evaluate the importance of multigranularity: a "system that puts all unigrams, short ngrams, long ngrams, and sentence representations into one interaction matrix." This would indeed be an interesting baseline, but there is no obvious way to conduct this experiment since vectors from different levels are not comparable; e.g., they have different dimensionality.

method	acc	F_1
baseline	66.5	79.9
NLM+	69.0	80.1
NLM$\odot$	67.8	79.3
NLM_RAE	70.3	81.3
ARC-I	69.6	80.3
ARC-II	69.9	80.9
RAE	76.7	83.6
MT	77.4	84.1
Bi-CNN-MI–	72.5	81.4
Bi-CNN-MI	**78.1**	**84.4**

Table 1: Performance of different systems on MSRP

	features used	acc	F_1
1	no features	66.5	79.9
2	+ u: unigram	68.4	79.7
3	+ sn: short ngram	75.3	82.8
4	+ ln: long ngram	76.2	83.1
5	+ s: sentence	73.4	82.3
6	– u: unigram	77.8	84.3
7	– sn: short ngram	76.3	83.5
8	– ln: long ngram	75.6	83.2
9	– s: sentence	77.6	84.2
10	all features	78.1	84.4

Table 2: Analysis of impact of the four feature classes. Line 1: majority baseline. Line 10: Bi-CNN-MI result from Table 1. Lines 2–5: Bi-CNN-MI when only one feature class is used. Line 6–9: ablation experiment: on each line one feature class is removed.

detection, especially when the training set is small. RAE also uses pretraining, but not as effectively as Bi-CNN-MI as Table 1 indicates. Hu et al. (2014) also suggest that training complex NNs only with supervised training runs the risk of overfitting on the small MSRP corpus.

Table 2 looks at the relative importance of the four feature matrices shown in Figure 1. (The unsupervised part of the training regime is not changed for this experiment.) The results indicate that levels sn and ln are most informative: F_1 scores are highest if only these two levels are used (lines 3&4: 82.8, 83.1) and performance drops most when they are removed (lines 7&8: 83.5, 83.2).

Unigrams contribute little to overall performance (lines 2&6), probably because the paraphrases in the corpus typically do not involve individual words (replacing one word by its synonym); rather, the paraphrase relationship involves larger context, which can only be judged by the higher-level features.

Just using the sentence matrix by itself performs well (line 5), but less well than using only levels sn or ln (lines 3&4). Most prior NN work on PI has taken the sentence-level approach. Our results indicate that combining this with the more fine-grained comparison on the ngram-level is superior.

Removing the sentence matrix results in a small drop in performance (line 9). The reason is that sentence representations are computed by k-max pooling from level ln. Thus, we can roughly view the sentence-level feature matrix $\mathbf{F}_s$ as a subset of $\mathbf{F}_{\text{ln}}$.

Adding (Madnani et al., 2012)'s MT metrics as input to the Bi-CNN-MI logistic regression further improves performance: accuracy of 78.4 and F_1 of 84.6.

7 Conclusion and future work

We presented the deep learning architecture Bi-CNN-MI for paraphrase identification (PI). Based on the insight that PI requires comparing two sentences *on multiple levels of granularity*, we learn multigranular sentence representations using convolution and compute interaction feature matrices at each level. These matrices are then the input to a logistic classifier for PI. All parameters of the model (for embeddings, convolution and classification) are directly optimized for PI. To address the lack of training data, we pretrain the network in a novel way for a language modeling task. Results on MSRP are state of the art.

In the future, we plan to apply Bi-CNN-MI to sentence matching, question answering and other tasks.

Acknowledgments

We are grateful to Thang Vu, Irina Sergienya and Sebastian Ebert for comments on earlier versions of this paper. This work was supported by Baidu (through a Baidu scholarship awarded to Wenpeng Yin) and by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/8-2, SPP 1335).

References

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2014. Exploiting discourse information to identify paraphrases. *Expert Systems with Applications*, 41(6):2832–2841.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Yee Seng Chan and Hwee Tou Ng. 2008. Maxsim: A maximum similarity metric for machine translation evaluation. In *ACL*, pages 55–62.

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

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603.

Michael Denkowski and Alon Lavie. 2010. Extending the meteor machine translation evaluation metric to the phrase level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 250–253.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350.

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

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Nizar Habash and Ahmed Elkholy. 2008. Sepia: surface span extension to syntactic dependency precision-based mt evaluation. In *Proceedings of the NIST metrics for machine translation workshop at the association for machine translation in the Americas conference, AMTA-2008. Waikiki, HI*.

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

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th international conference on Machine learning*.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 NAACL-HLT*, pages 182–190.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

N Neverova, C Wolf, GW Taylor, and F Nebout. 2014. Multi-scale deep learning for gesture detection and localization. In *European Conference on Computer Vision (ECCV) 2014 ChaLearn Workshop. Zurich*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318.

Steven Parker. 2008. Badger: A new machine translation metric. *Metrics for Machine Translation Challenge*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006.

Pengcheng Wu, Steven CH Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 153–162. ACM.

Chapter 4

MultiGranCNN: An Architecture for General Matching of Text Chunks on Multiple Levels of Granularity

MultiGranCNN: An Architecture for General Matching of Text Chunks on Multiple Levels of Granularity

Wenpeng Yin and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
wenpeng@cis.uni-muenchen.de

Abstract

We present MultiGranCNN, a general deep learning architecture for matching text chunks. MultiGranCNN supports *multigranular comparability* of representations: shorter sequences in one chunk can be directly compared to longer sequences in the other chunk. MultiGranCNN also contains a *flexible and modularized match feature component* that is easily adaptable to different types of chunk matching. We demonstrate state-of-the-art performance of MultiGranCNN on clause coherence and paraphrase identification tasks.

1 Introduction

Many natural language processing (NLP) tasks can be posed as classifying the relationship between two TEXTCHUNKS (cf. Li et al. (2012), Bordes et al. (2014b)) where a TEXTCHUNK can be a sentence, a clause, a paragraph or any other sequence of words that forms a unit.

Paraphrasing (Figure 1, top) is one task that we address in this paper and that can be formalized as classifying a TEXTCHUNK relation. The two classes correspond to the sentences being (e.g., the pair <$\mathbf{p}$, $\mathbf{q}^+$>) or not being (e.g., the pair <$\mathbf{p}$, $\mathbf{q}^-$>) paraphrases of each other. Another task we look at is clause coherence (Figure 1, bottom). Here the two TEXTCHUNK relation classes correspond to the second clause being (e.g., the pair <$\mathbf{x}$, $\mathbf{y}^+$>) or not being (e.g., the pair <$\mathbf{x}$, $\mathbf{y}^-$>) a discourse-coherent continuation of the first clause. Other tasks that can be formalized as TEXTCHUNK relations are question answering (QA) (is the second chunk an answer to the first?), textual inference (does the first chunk imply the second?) and machine translation (are the two chunks translations of each other?).

$\mathbf{p}$	PDC will also almost certainly *fan the flames of* speculation about *Longhorn's release*.
$\mathbf{q}^+$	PDC will also almost certainly *reignite* speculation about *release dates of Microsoft 's new products*.
$\mathbf{q}^-$	PDC is indifferent to the release of Longhorn.
$\mathbf{x}$	The dollar suffered its worst one-day loss in a month,
$\mathbf{y}^+$	falling to 1.7717 marks ... from 1.7925 marks yesterday.
$\mathbf{y}^-$	up from 112.78 yen in late New York trading yesterday.

Figure 1: Examples for paraphrasing and clause coherence tasks

In this paper, we present MultiGranCNN, a general architecture for TEXTCHUNK relation classification. MultiGranCNN can be applied to a broad range of different TEXTCHUNK relations. This is a challenge because natural language has a complex structure – both sequential and hierarchical – and because this structure is usually not parallel in the two chunks that must be matched, further increasing the difficulty of the task. A successful detection algorithm therefore needs to capture not only the internal structure of TEXTCHUNKS, but also the rich pattern of their interactions.

MultiGranCNN is based on two innovations that are critical for successful TEXTCHUNK relation classification. First, the architecture is designed to ensure *multigranular comparability*. For general matching, we need the ability to match short sequences in one chunk with long sequences in the other chunk. For example, what is expressed by a single word in one chunk ("reignite" in $\mathbf{q}^+$ in the figure) may be expressed by a sequence of several words in its paraphrase ("fan the flames of" in $\mathbf{p}$). To meet this objective, we learn representations for words, phrases and the entire sentence that are all mutually comparable; in particular, these representations all have the same dimensionality and live in the same space.

Most prior work (e.g., Blacoe and Lapata (2012; Hu et al. (2014)) has neglected the need for multigranular comparability and performed matching within fixed levels only, e.g., only words were matched with words or only sentences with sentences. For a general solution to the problem of matching, we instead need the ability to match a unit on a lower level of granularity in one chunk with a unit on a higher level of granularity in the other chunk. Unlike (Socher et al., 2011), our model does not rely on parsing and it can more exhaustively search the hypothesis space of possible matchings, including matchings that correspond to conflicting segmentations of the input chunks (see Section 5).

Our second contribution is that MultiGranCNN contains a *flexible and modularized match feature component*. This component computes the basic features that measure how well phrases of the two chunks match. We investigate three different *match feature models* that demonstrate that a wide variety of different match feature models can be implemented. The match feature models can be swapped in and out of MultiGranCNN, depending on the characteristics of the task to be solved.

Prior work that has addressed matching tasks has usually focused on a single task like QA (Bordes et al., 2014a; Yu et al., 2014) or paraphrasing (Socher et al., 2011; Madnani et al., 2012; Ji and Eisenstein, 2013). The ARC architectures proposed by Hu et al. (2014) are intended to be more general, but seem to be somewhat limited in their flexibility to model different matching relations; e.g., they do not perform well for paraphrasing.

Different match feature models may also be required by factors other than the characteristics of the task. If the amount of labeled training data is small, then we may prefer a match feature model with few parameters that is robust against overfitting. If there is lots of training data, then a richer match feature model may be the right choice. This motivates the need for an architecture like MultiGranCNN that allows selection of the task-appropriate match feature model from a range of different models and its seamless integration into the architecture.

In remaining parts, Section 2 introduces some related work; Section 3 gives an overview of the proposed MultiGranCNN; Section 4 shows how to learn representations for generalized phrases (g-phrases); Section 5 describes the three matching models: DIRECTSIM, INDIRECTSIM and CONCAT; Section 6 describes the two 2D pooling methods: grid-based pooling and phrase-based pooling; Section 7 describes the match feature CNN; Section 8 summarizes the architecture of MultiGran CNN; and Section 9 presents experiments; finally, Section 10 concludes.

2 Related Work

Paraphrase identification (PI) is a typical task of sentence matching and it has been frequently studied (Qiu et al., 2006; Blacoe and Lapata, 2012; Madnani et al., 2012; Ji and Eisenstein, 2013). Socher et al. (2011) utilized parsing to model the hierarchical structure of sentences and uses unfolding recursive autoencoders to learn representations for single words and phrases acting as nonleaf nodes in the tree. The main difference to MultiGranCNN is that we stack multiple convolution layers to model flexible phrases and learn representations for them, and aim to address more general sentence correspondence. Bach et al. (2014) claimed that elementary discourse units obtained by segmenting sentences play an important role in paraphrasing. Their conclusion also endorses (Socher et al., 2011)'s and our work, for both take interactions between component phrases into account.

QA is another representative sentence matching problem. Yu et al. (2014) modeled sentence representations in a simplified CNN, finally finding the match score by projecting question and answer candidates into the same space. Other relevant QA work includes (Bordes et al., 2014c; Bordes et al., 2014a; Yang et al., 2014; Iyyer et al., 2014)

For more general matching, Chopra et al. (2005) and Liu (2013) used a Siamese architecture of shared-weight neural networks (NNs) to model two objects simultaneously, matching their representations and then learning a specific type of sentence relation. We adopt parts of their architecture, but we model phrase representations as well as sentence representations.

Li and Xu (2012) gave a comprehensive introduction to query-document matching and argued that query and document match at different levels: term, phrase, word sense, topic, structure etc. This also applies to sentence matching.

Lu and Li (2013) addressed matching of short texts. Interactions between the two texts were obtained via LDA (Blei et al., 2003) and were then the basis for computing a matching score. Compared to MultiGranCNN, drawbacks of this approach are that LDA parameters are not optimized for the specific task and that the interactions are formed on the level of single words only.

Gao et al. (2014) modeled interestingness between two documents with deep NNs. They mapped source-target document pairs to feature vectors in a latent space in such a way that the distance between the source document and its corresponding interesting target in that space was minimized. Interestingness is more like topic relevance, based mainly on the aggregated meaning of keywords, as opposed to more structural relationships as is the case for paraphrasing and clause coherence.

We briefly discussed (Hu et al., 2014)'s ARC in Section 1. MultiGranCNN is partially inspired by ARC, but introduces multigranular comparability (thus enabling crosslevel matching) and supports a wider range of match feature models.

Our unsupervised learning component (Section 4, last paragraph) resembles word2vec CBOW (Mikolov et al., 2013), but learns representations of TEXTCHUNKS as well as words. It also resembles PV-DM (Le and Mikolov, 2014), but our TEXTCHUNK representation is derived using a *hierarchical* architecture based on *convolution and pooling*.

3 Overview of MultiGranCNN

We use convolution-plus-pooling in two different components of MultiGranCNN. The first component, the *generalized phrase CNN* (gpCNN), will be introduced in Section 4. This component learns representations for *generalized phrases* (g-phrases) where a generalized phrase is a general term for subsequences of all granularities: words, short phrases, long phrases and the sentence itself. The gpCNN architecture has L layers of convolution, corresponding (for $L = 2$) to words, short phrases, long phrases and the sentence. We test different values of L in our experiments. We train gpCNN on large data in an unsupervised manner and then fine-tune it on labeled training data.

Using a *Siamese configuration*, two copies of gpCNN, one for each of the two input TEXTCHUNKS, are the input to the *match feature model*, presented in Section 5. This model produces $s_1 \times s_2$ matching features, one for each pair of g-phrases in the two chunks, where s_1, s_2 are the number of g-phrases in the two chunks, respectively.

The $s_1 \times s_2$ match feature matrix is first reduced to a fixed size by *dynamic 2D pooling*. The resulting fixed size matrix is then the input to the second convolution-plus-pooling component, the *match feature CNN* (mfCNN) whose output is fed to a *multilayer perceptron* (MLP) that produces the final match score. Section 6 will give details.

We use convolution-plus-pooling for both word sequences and match features because we want to compute increasingly abstract features at multiple levels of granularity. To ensure that g-phrases are mutually comparable when computing the $s_1 \times s_2$ match feature matrix, we impose the constraint that *all g-phrase representations live in the same space and have the same dimensionality*.

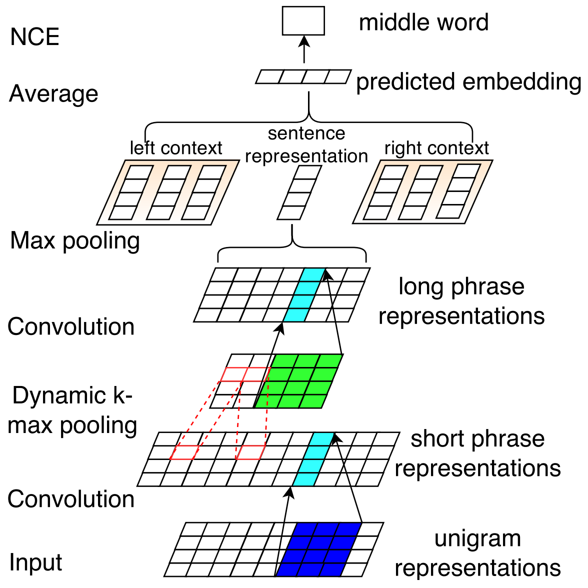


Figure 2: gpCNN: learning g-phrase representations. This figure only shows two convolution layers (i.e., $L = 2$) for saving space.

4 gpCNN: Learning Representations for g-Phrases

We use several stacked *blocks*, i.e., convolution-plus-pooling layers, to extract increasingly abstract features of the TEXTCHUNK. The input to the first block are the words of the TEXTCHUNK, represented by CW (Collobert and Weston, 2008) embeddings. Given a TEXTCHUNK of length $|S|$, let vector $\mathbf{c}_i \in \mathbb{R}^{wd}$ be the concatenated embeddings of words $v_{i-w+1}, \ldots, v_i$ where $w = 5$ is the filter width, $d = 50$ is the dimensionality of CW embeddings and $0 < i < |S| + w$. Embeddings for words v_i, $i < 1$ and $i > |S|$, are set to zero. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^d$ of the g-phrase $v_{i-w+1}, \ldots, v_i$ using the convolution matrix $\mathbf{W}_l \in \mathbb{R}^{d \times wd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W}_l \mathbf{c}_i + \mathbf{b}_l) \quad (1)$$

where block index $l = 1$, bias $\mathbf{b}_l \in \mathbb{R}^d$. We use *wide convolution* (i.e., we apply the convolution matrix $\mathbf{W}_l$ to words v_i, $i < 1$ and $i > |S|$) because this makes sure that each word v_i, $1 \leq i \leq |S|$, can be detected by all weights of $\mathbf{W}_l$ – as opposed to only the rightmost (resp. leftmost) weights for initial (resp. final) words in narrow convolution.

The configuration of convolution layers in following blocks ($l > 1$) is exactly the same except that the input vectors $\mathbf{c}_i$ are not words, but the output of pooling from the previous layer of convolution – as we will explain presently. The configuration is the same (e.g., all $\mathbf{W}_l \in \mathbb{R}^{d \times wd}$) because, by design, all g-phrase representations have the same dimensionality d. This also ensures that each g-phrase representation can be directly compared with each other g-phrase representation.

We use *dynamic k-max pooling* to extract the k_l top values from each dimension after convolution in the l^{th} block and the k_L top values in the final block. We set

$$k_l = \max(\alpha, \lceil \frac{L - l}{L} |S| \rceil) \quad (2)$$

where $l = 1, \cdots, L$ is the block index, and $\alpha = 4$ is a constant (cf. Kalchbrenner et al. (2014)) that makes sure a reasonable minimum number of values is passed on to the next layer. We set $k_L = 1$ (not 4, cf. Kalchbrenner et al. (2014)) because our design dictates that all g-phrase representations, including the representation of the TEXTCHUNK itself, have the same dimensionality. Example: for $L = 4, |S| = 20$, the k_i are $[15, 10, 5, 1]$.

Dynamic k-max pooling keeps the most important features and allows us to stack multiple blocks to extract hiearchical features: units on consecutive layers correspond to larger and larger parts of the TEXTCHUNK thanks to the subset selection property of pooling.

For many tasks, labeled data for training gpCNN is limited. We therefore employ *unsupervised training* to initialize gpCNN as shown in Figure 2. Similar to CBOW (Mikolov et al., 2013), we predict a sampled middle word v_i from the average of seven vectors: the TEXTCHUNK representation (the final output of gpCNN) and the three words to the left and to the right of v_i. We use noise-contrastive estimation (Mnih and Teh, 2012) for training: 10 noise words are sampled for each true example.

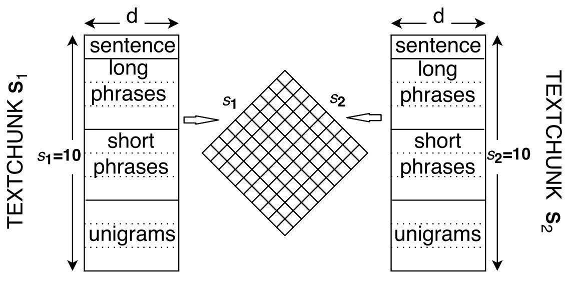


Figure 3: General illustration of match feature model. In this example, both S_1 and S_2 have 10 g-phrases, so the match feature matrix $\hat{\mathbf{F}} \in \mathbb{R}^{s_1 \times s_2}$ has size 10×10.

5 Match Feature Models

Let $g_1, \ldots, g_{s_k}$ be an enumeration of the s_k g-phrases of TEXTCHUNK S_k. Let $\mathbf{S}_k \in \mathbb{R}^{s_k \times d}$ be the matrix, constructed by concatenating the four matrices of unigram, short phrase, long phrase and sentence representations shown in Figure 2 that contain the learned representations from Section 4 for these s_k g-phrases; i.e., row $\mathbf{S}_{ki}$ is the learned representation of g_i.

The basic design of a match feature model is that we produce an $s_1 \times s_2$ matrix $\hat{\mathbf{F}}$ for a pair of TEXTCHUNKS S_1 and S_2, shown in Figure 3. $\hat{\mathbf{F}}_{i,j}$ is a score that assesses the relationship between g-phrase g_i of S_1 and g-phrase g_j of S_2 *with respect to the* TEXTCHUNK *relation of interest* (paraphrasing, clause coherence etc). This score $\hat{\mathbf{F}}_{i,j}$ is computed based on the vector representations $\mathbf{S}_{1i}$ and $\mathbf{S}_{2j}$ of the two g-phrases.[1]

We experiment with three different feature models to compute the match score $\hat{\mathbf{F}}_{i,j}$ because we would like our architecture to address a wide variety of different TEXTCHUNK relations. We can model a TEXTCHUNK relation like paraphrasing as "for each meaning element in one sentence, there must be a similar meaning element in the other sentence"; thus, a good candidate for the match score $\hat{\mathbf{F}}_{i,j}$ is simply vector similarity. In contrast, similarity is a less promising match score for clause coherence; for clause coherence, we want a score that models how good a continuation one g-phrase is for the other. These considerations motivate us to define three different match feature models that we will introduce now.

The first match feature model is DIRECTSIM.

[1]In response to a reviewer question, recall that s_i is the total number of g-phrases of S_i, so there is only one $s_1 \times s_2$ matrix, not several on different levels of granularity.

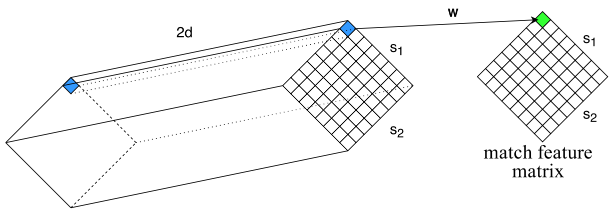


Figure 4: CONCAT match feature model

This model computes the match score of two g-phrases as their similarity using a radial basis function kernel:

$$\hat{\mathbf{F}}_{i,j} = \exp\left(\frac{-||\mathbf{S}_{1i} - \mathbf{S}_{2j}||^2}{2\beta}\right) \tag{3}$$

where we set $\beta = 2$ (cf. Wu et al. (2013)). DIRECTSIM is an appropriate feature model for TEXTCHUNK relations like paraphrasing because in that case direct similarity features are helpful in assessing meaning equivalence.

The second match feature model is INDIRECTSIM. Instead of computing the similarity directly as we do for DIRECTSIM, we first transform the representation of the g-phrase in one TEXTCHUNK using a transformation matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, then compute the match score by inner product and sigmoid activation:

$$\hat{\mathbf{F}}_{i,j} = \sigma(\mathbf{S}_{1i}\mathbf{M}\mathbf{S}_{2j}^{\mathrm{T}} + b), \tag{4}$$

Our motivation is that for a TEXTCHUNK relation like clause coherence, the two TEXTCHUNKS need not have any direct similarity. However, if we map the representations of TEXTCHUNK S_1 into an appropriate space then we can hope that similarity between these transformed representations of S_1 and the representations of TEXTCHUNK S_2 do yield useful features. We will see that this hope is borne out by our experiments.

The third match feature model is CONCAT. This is a general model that can learn any weighted combination of the values of the two vectors:

$$\hat{\mathbf{F}}_{i,j} = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{e}_{i,j} + b) \tag{5}$$

where $\mathbf{e}_{i,j} \in \mathbb{R}^{2d}$ is the concatenation of $\mathbf{S}_{1i}$ and $\mathbf{S}_{2j}$. We can learn different combination weights $\mathbf{w}$ to solve different types of TEXTCHUNK matching.

We call this match feature model CONCAT because we implement it by concatenating g-phrase vectors to form a tensor as shown in Figure 4.

The match feature models implement multi-granular comparability: they match all units in one TEXTCHUNK with all units in the other TEXTCHUNK. This is necessary because a general solution to matching must match a low-level unit like "reignite" to a higher-level unit like "fan the flames of" (Figure 1). Unlike (Socher et al., 2011), our model does not rely on parsing; therefore, it can more exhaustively search the hypothesis space of possible matchings: mfCNN covers a wide variety of different, possibly overlapping units, not just those of a single parse tree.

6 Dynamic 2D Pooling

The match feature models generate an $s_1 \times s_2$ matrix. Since it has variable size, we apply two different dynamic 2D pooling methods, *grid-based pooling* and *phrase-focused pooling*, to transform it to a fixed size matrix.

6.1 Grid-based pooling

We need to map $\hat{\mathbf{F}} \in \mathbb{R}^{s_1 \times s_2}$ into a matrix $\mathbf{F}$ of fixed size $s^* \times s^*$ where s^* is a parameter. Grid-based pooling divides $\hat{\mathbf{F}}$ into $s^* \times s^*$ nonoverlapping *(dynamic) pools* and copies the maximum value in each dynamic pool to $\mathbf{F}$. This method is similar to (Socher et al., 2011), but preserves locality better.

$\hat{\mathbf{F}}$ can be split into equal regions only if both s_1 and s_2 are divisible by s^*. Otherwise, for $s_1 > s^*$ and if $s_1 \bmod s^* = b$, the dynamic pools in the first $s^* - b$ splits each have $\left\lfloor \frac{s_1}{s^*} \right\rfloor$ rows while the remaining b splits each have $\left\lfloor \frac{s_1}{s^*} \right\rfloor + 1$ rows. In Figure 5, a $s_1 \times s_2 = 4 \times 5$ matrix (left) is split into $s^* \times s^* = 3 \times 3$ dynamic pools (middle): each row is split into [1, 1, 2] and each column is split into [1, 2, 2].

If $s_1 < s^*$, we first repeat all rows in batch style with size s_1 until no fewer than s^* rows remain. Then the first s^* rows are kept and split into s^* dynamic pools. The same principle applies to the partitioning of columns. In Figure 5 (right), the areas with dashed lines and dotted lines are repeated parts for rows and columns, respectively; each cell is its own dynamic pool.

6.2 Phrase-focused pooling

In the match feature matrix $\hat{\mathbf{F}} \in \mathbb{R}^{s_1 \times s_2}$, row i (resp. column j) contains all feature values for g-phrase g_i of S_1 (resp. g_j of S_2). Phrase-focused pooling attempts to pick the largest match features

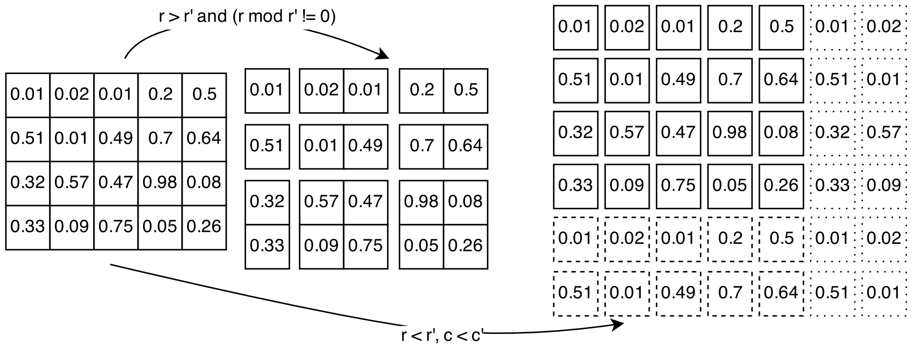


Figure 5: Partition methods in grid-based pooling. Original matrix with size 4×5 is mapped into matrix with size 3×3 and matrix with size 6×7, respectively. Each dynamic pool is distinguished by a border of empty white space around it.

for a g-phrase g on the assumption that they are the best basis for assessing the relation of g with other g-phrases. To implement this, we sort the values of each row i (resp. each column j) in decreasing order giving us a matrix $\hat{\mathbf{F}}_r \in \mathbb{R}^{s_1 \times s_2}$ with sorted rows (resp. $\hat{\mathbf{F}}_c \in \mathbb{R}^{s_1 \times s_2}$ with sorted columns). Then we concatenate the columns of $\hat{\mathbf{F}}_r$ (resp. the rows of $\hat{\mathbf{F}}_c$) resulting in list $F_r = \{f^r_1, \dots, f^r_{s_1 s_2}\}$ (resp. $F_c = \{f^c_1, \dots, f^c_{s_1 s_2}\}$) where each f^r (f^c) is an element of $\hat{\mathbf{F}}_r$ ($\hat{\mathbf{F}}_c$). These two lists are merged into a list F by interleaving them so that members from F_r and F_c alternate. F is then used to fill the rows of $\mathbf{F}$ from top to bottom with each row being filled from left to right.[2]

7 mfCNN: Match feature CNN

The output of dynamic 2D pooling is further processed by the match feature CNN (mfCNN) as depicted in Figure 6. mfCNN extracts increasingly abstract interaction features from lower-level interaction features, using several layers of 2D wide convolution and fixed-size 2D pooling.

We call the combination of a 2D wide convolution layer and a fixed-size 2D pooling layer a *block*, denoted by index b ($b = 1, 2 \dots$). In general, let tensor $\mathbf{T}^b \in \mathbb{R}^{c_b \times s_b \times s_b}$ denote the feature maps in block b; block b has c_b feature maps, each of size $s_b \times s_b$ ($\mathbf{T}^1 = \mathbf{F} \in \mathbb{R}^{1 \times s^* \times s^*}$). Let $\mathbf{W}^b \in \mathbb{R}^{c_{b+1} \times c_b \times f_b \times f_b}$ be the filter weights of 2D wide convolution in block b, $f_b \times f_b$ is then the size of sliding convolution regions. Then the convolution is performed as element-wise multiplication between $\mathbf{W}^b$ and $\mathbf{T}^b$ as follows:

$$\hat{\mathbf{T}}^{b+1}_{m,i-1,j-1} = \sigma(\sum \mathbf{W}^b_{m,:,:,:} \mathbf{T}^b_{:,i-f_b:i,j-f_b:j} + \mathbf{b}^b_m) \tag{6}$$

where $0 \le m < c_{b+1}$, $1 \le i, j < s_b + f_b$, $\mathbf{b}^b \in \mathbb{R}^{c_{b+1}}$.

Subsequently, fixed-size 2D pooling selects dominant features from $k_b \times k_b$ non-overlapping windows of $\hat{\mathbf{T}}^{b+1}$ to form a tensor as input of block $b + 1$:

$$\mathbf{T}^{b+1}_{m,i,j} = \max(\hat{\mathbf{T}}^{b+1}_{m,ik_b:(i+1)k_b,jk_b:(j+1)k_b}) \tag{7}$$

where $0 \le i, j < \lfloor \frac{s_b + f_b - 1}{k_b} \rfloor$.

Hu et al. (2014) used narrow convolution which would limit the number of blocks. 2D wide convolution in this work enables to stack multiple blocks of convolution and pooling to extract higher-level interaction features. We will study the influence of the number of blocks on performance below.

For the experiments, we set $s^* = 40$, $c_b = 50$, $f_b = 5$, $k_b = 2$ ($b = 1, 2, \cdots$).

8 MultiGranCNN

We can now describe the overall architecture of MultiGranCNN. First, using a Siamese configuration, two copies of gpCNN, one for each of the two input TEXTCHUNKS, produce g-phrase representations on different levels of abstraction (Figure 2). Then one of the three match feature models (DIRECTSIM, CONCAT or INDIRECTSIM) produces an $s_1 \times s_2$ match feature matrix, each cell of which assesses the match of a pair of g-phrases from the two chunks. This match feature matrix is reduced to a fixed size matrix by dynamic 2D pooling (Section 6). As shown in Figure 6, the resulting fixed size matrix is the input for mfCNN, which extracts interaction features of

[2]If $\hat{\mathbf{F}}$ has fewer cells than $\mathbf{F}$, then we simply repeat the filling procedure to fill all cells.

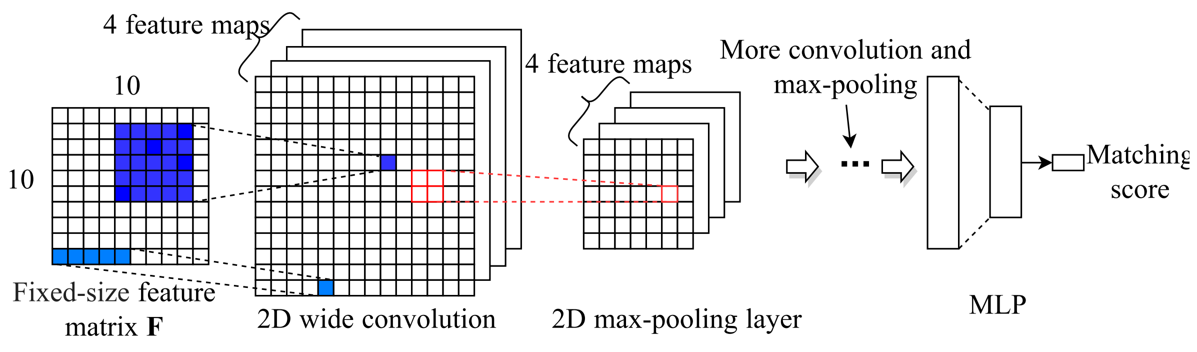


Figure 6: mfCNN & MLP for matching score learning. $s^* = 10$, $f_b = 5$, $k_b = 2$, $c_b = 4$ in this example.

increasing complexity from the basic interaction features computed by the match feature model. Finally, the output of the last block of mfCNN is the input to an MLP that computes the match score.

MultiGranCNN bears resemblance to previous work on clause and sentence matching (e.g., Hu et al. (2014), Socher et al. (2011)), but it is more general and more flexible. It learns representations of g-phrases, i.e., representations of parts of the TEXTCHUNK at multiple granularities, not just for a single level such as the sentence as ARC-I does (Hu et al., 2014). MultiGranCNN explores the space of interactions between the two chunks more exhaustively by considering interactions between every unit in one chunk with every other unit in the other chunk, at all levels of granularity. Finally, MultiGranCNN supports a number of different match feature models; the corresponding module can be instantiated in a way that ensures that match features are best suited to support accurate decisions on the TEXTCHUNK relation task that needs to be addressed.

9 Experimental Setup and Results

9.1 Training

Suppose the triple $(\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-)$ is given and $\mathbf{x}$ matches $\mathbf{y}^+$ better than $\mathbf{y}^-$. Then our objective is the minimization of the following ranking loss:

$$l(\mathbf{x}, \mathbf{y}^+, \mathbf{y}^-) = \max(0, 1 + s(\mathbf{x}, \mathbf{y}^-) - s(\mathbf{x}, \mathbf{y}^+))$$

where $s(\mathbf{x}, \mathbf{y})$ is the predicted match score for $(\mathbf{x}, \mathbf{y})$. We use stochastic gradient descent with Adagrad (Duchi et al., 2011), L_2 regularization and minibatch training.

We set initial learning rate to 0.05, batch size to 70, L_2 weight to $5 \cdot 10^{-4}$.

Recall that we employ unsupervised pretraining of representations for g-phrases. We can either *freeze* these representations in subsequent supervised training; or we can *fine-tune* them. We study the performance of both regimes.

9.2 Clause Coherence Task

As introduced by Hu et al. (2014), the *clause coherence* task determines for a pair $(\mathbf{x}, \mathbf{y})$ of clauses if the sentence "$\mathbf{xy}$" is a coherent sentence. We construct a clause coherence dataset as follows (the set used by Hu et al. (2014) is not yet available). We consider all sentences from English Gigaword (Parker et al., 2009) that consist of two comma-separated clauses $\mathbf{x}$ and $\mathbf{y}$, with each clause having between five and 30 words. For each $\mathbf{y}$, we choose four clauses $\mathbf{y}' \ldots \mathbf{y}''''$ randomly from the 1000 second clauses that have the highest similarity to $\mathbf{y}$, where similarity is cosine similarity of TF-IDF vectors of the clauses; restricting the alternatives to similar clauses ensures that the task is hard. The clause coherence task then is to select $\mathbf{y}$ from the set $\mathbf{y}, \mathbf{y}', \ldots, \mathbf{y}''''$ as the correct continuation of $\mathbf{x}$. We create 21 million examples, each consisting of a first clause $\mathbf{x}$ and five second clauses. This set is divided into a training set of 19 million and development and test sets of one million each. An example from the training set is given in Figure 1.

Then, we study the performance variance of different MultiGranCNN setups from three perspectives: a) layers of CNN in both unsupervised (gpCNN) and supervised (mfCNN) training phases; b) different approaches for clause relation feature modeling; c) dynamic pooling methods for generating same-sized feature matrices.

Figure 7 (top table) shows that (Hu et al., 2014)'s parameters are good choices for our setup as well. We get best result when both gpCNN and mfCNN have three blocks of convolution and pooling. This suggests that multiple layers of convolution succeed in extracting high-level features that are beneficial for clause coherence.

Figure 7 (2nd table) shows that INDIRECTSIM and CONCAT have comparable performance and both outperform DIRECTSIM. DIRECTSIM is expected to perform poorly because the contents in the two clauses usually have little or no overlapping meaning. In contrast, we can imagine that INDIRECTSIM first transforms the first clause $\mathbf{x}$ into a counterpart and then matches this counterpart with the second clause $\mathbf{y}$. In CONCAT, each of $s_1 \times s_2$ pairs of g-phrases is concatentated and supervised training can then learn an unrestricted function to assess the importance of this pair for clause coherence (cf. Eq. 5). Again, this is clearly a more promising TEXTCHUNK relation model for clause coherence than one that relies on DIRECTSIM.

acc		mfCNN 0	mfCNN 1	mfCNN 2	mfCNN 3
gpCNN	0	38.02	44.08	47.81	48.43
	1	40.91	45.31	51.73	52.13
	2	43.10	48.06	54.14	54.86
	3	45.62	51.77	55.97	**56.31**

match feature model	acc
DIRECTSIM	25.40
INDIRECTSIM	**56.31**
CONCAT	56.12

freeze g-phrase represenations or not	acc
MultiGranCNN (freeze)	55.79
MultiGranCNN (fine-tune)	**56.31**

pooling method	acc
dynamic (Socher et al., 2011)	55.91
grid-based	56.07
phrase-focused	**56.31**

Figure 7: Effect on dev acc (clause coherence) of different factors: # convolution blocks, match feature model, freeze vs. fine-tune, pooling method.

Figure 7 (3rd table) demonstrates that fine-tuning g-phrase representations gives better performance than freezing them. Also, grid-based and phrase-focused pooling outperform dynamic pooling (Socher et al., 2011) (4th table). Phrase-focused pooling performs best.

Table 1 compares MultiGranCNN to ARC-I and ARC-II, the architectures proposed by Hu et al. (2014). We also test the five baseline systems from their paper: DeepMatch, WordEmbed, SENMLP, SENNA+MLP, URAE+MLP. For MultiGranCNN, we use the best dev set settings: number of convolution layers in gpCNN and mfCNN is 3; INDIRECTSIM; phrase-focused pooling. Table 1 shows that MultiGranCNN outperforms all other approaches on clause coherence test set.

9.3 Paraphrase Identification Task

We evaluate paraphrase identification (PI) on the PAN corpus (http://bit.ly/mt-para, (Madnani et al., 2012)), consisting of training and test sets of 10,000 and 3000 sentence pairs, respectively. Sentences are about 40 words long on average.

Since PI is a binary classification task, we replace the MLP with a logistic regression layer. As phrase-focused pooling was proven to be optimal, we directly use phrase-focused pooling in PI task without comparison, assuming that the choice of dynamic pooling is task independent.

For parameter selection, we split the PAN training set into a core training set (core) of size 9000 and a development set (dev) of size 1000. We then train models on core and select parameters based on best performance on dev. The best results on dev are obtained for the following parameters: freezing g-phrase representations, DIRECTSIM, two convolution layers in gpCNN, no convolution layers in mfCNN. We use these parameter settings to train a model on the entire training set and report performance in Table 2.

We compare MultiGranCNN to ARC-I/II (Hu et al., 2014), and two previous papers reporting performance on PAN. Madnani et al. (2012) used a combination of three *basic MT metrics* (BLEU, NIST and TER) and five complex MT metrics (TERp, METEOR, BADGER, MAXISIM,

model	acc
Random Guess	20.00
DeepMatch	34.17
WordEmbed	38.28
SENMLP	34.57
SENNA+MLP	42.09
URAE+MLP	27.41
ARC-I	45.04
ARC-II	50.18
MultiGranCNN	**56.27**

Table 1: Performance on clause coherence test set.

SEPIA), computed on entire sentences. Bach et al. (2014) applied MT metrics to elementary discourse units. We integrate these eight MT metrics from prior work.

method	acc	F_1
ARC-I	61.4	60.3
ARC-II	64.9	63.5
basic MT metrics	88.6	87.8
+ TERp	91.5	91.2
+ METEOR	92.0	91.8
+ Others	92.3	92.1
(Bach et al., 2014)	93.4	93.3
8MT+MultiGranCNN (fine-tune)	94.1	94.0
8MT+MultiGranCNN (freeze)	**94.9**	**94.7**

Table 2: Results on PAN. "8MT" = 8 MT metrics

Table 2 shows that MultiGranCNN in combination with MT metrics obtains state-of-the-art performance on PAN. *Freezing* weights learned in unsupervised training (Figure 2) performs better than *fine-tuning* them; also, Table 3 shows that the best result is achieved if *no convolution* is used in mfCNN. Thus, the best configuration for paraphrase identification is to "forward" fixed-size interaction matrices as input to the logistic regression, without any intermediate convolution layers.

Freezing weights learned in unsupervised training and no convolution layers in mfCNN both protect against overfitting. Complex deep neural networks are in particular danger of overfitting when training sets are small as in the case of PAN (cf. Hu et al. (2014)). In contrast, fine-tuning weights and several convolution layers were the optimal setup for clause coherence. For clause coherence, we have a much larger training set and therefore can successfully train a much larger number of parameters.

Table 3 shows that CONCAT performs badly for PI while DIRECTSIM and INDIRECTSIM perform well. We can conceptualize PI as the task of determining if each meaning element in S_1 has a similar meaning element in S_2. The $s_1 \times s_2$ DIRECTSIM feature model directly models this task and the $s_1 \times s_2$ INDIRECTSIM feature model also models it, but learning a transformation of g-phrase representations before applying similarity. In contrast, CONCAT can learn arbitrary relations between parts of the two sentences, a model that seems to be too unconstrained for PI if insufficient training resources are available.

In contrast, for the clause coherence task, concatentation worked well and DIRECTSIM worked poorly and we provided an explanation based on the specific properties of clause coherence (see discussion of Figure 7). We conclude from these results that it is dependent on the task what the best feature model is for matching two linguistic objects. Interestingly, INDIRECTSIM performs well on both tasks. This suggests that INDIRECTSIM is a general feature model for matching, applicable to tasks with very different properties.

10 Conclusion

In this paper, we present MultiGranCNN, a general deep learning architecture for classifying the relation between two TEXTCHUNKS. MultiGranCNN supports *multigranular comparability* of representations: shorter sequences in one TEXTCHUNK can be directly compared to longer sequences in the other TEXTCHUNK. MultiGranCNN also contains a *flexible and modularized match feature component* that is easily adaptable to different TEXTCHUNK relations. We demonstrated state-of-the-art performance of MultiGranCNN on paraphrase identification and clause coherence tasks.

Acknowledgments

Thanks to CIS members and anonymous reviewers for constructive comments. This work was supported by Baidu (through a Baidu scholarship awarded to Wenpeng Yin) and by Deutsche Forschungsgemeinschaft (grant DFG SCHU 2246/8-2, SPP 1335).

F_1		mfCNN 0	mfCNN 1	mfCNN 2	mfCNN 3
gpCNN	0	92.7	92.9	92.9	93.9
gpCNN	1	93.2	93.5	93.9	93.5
gpCNN	2	**94.7**	94.2	93.7	93.3
gpCNN	3	94.5	94.0	93.6	92.9

match feature model	acc	F_1
DIRECTSIM	**94.9**	**94.7**
INDIRECTSIM	94.7	94.5
CONCAT	93.0	92.9

Table 3: Effect on dev F_1 (PI) of different factors: # convolution blocks, match feature model.

References

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2014. Exploiting discourse information to identify paraphrases. *Expert Systems with Applications*, 41(6):2832–2841.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014b. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014c. Open question answering with weakly supervised embedding models. *Proceedings of 2014 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

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

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

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

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

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 633–644.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896.

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

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Proceedings of The 31st International Conference on Machine Learning*, pages 1188–1196.

Hang Li and Jun Xu. 2012. Beyond bag-of-words: machine learning for query-document matching in web search. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1177–1177. ACM.

Xutao Li, Michael K Ng, and Yunming Ye. 2012. Har: Hub, authority and relevance scores in multi-relational data for query search. In *Proceedings of the 12th SIAM International Conference on Data Mining*, pages 141–152. SIAM.

Chen Liu. 2013. *Probabilistic Siamese Network for Learning Representations*. Ph.D. thesis, University of Toronto.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

Robert Parker, Linguistic Data Consortium, et al. 2009. *English gigaword fourth edition*. Linguistic Data Consortium.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 18–26. Association for Computational Linguistics.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Pengcheng Wu, Steven CH Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 153–162. ACM.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 645–650.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS deep learning workshop*.

Chapter 5

ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs

ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs

Wenpeng Yin, Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
wenpeng@cis.lmu.de

Bing Xiang, Bowen Zhou
IBM Watson
Yorktown Heights, NY, USA
bingxia,zhou@us.ibm.com

Abstract

How to model a pair of sentences is a critical issue in many NLP tasks such as answer selection (AS), paraphrase identification (PI) and textual entailment (TE). Most prior work (i) deals with one individual task by fine-tuning a specific system; (ii) models each sentence's representation separately, rarely considering the impact of the other sentence; or (iii) relies fully on manually designed, task-specific linguistic features. This work presents a general **A**ttention **B**ased **C**onvolutional **N**eural **N**etwork (ABCNN) for modeling a pair of sentences. We make three contributions. (i) The ABCNN can be applied to a wide variety of tasks that require modeling of sentence pairs. (ii) We propose three attention schemes that integrate mutual influence between sentences into CNNs; thus, the representation of each sentence takes into consideration its counterpart. These interdependent sentence pair representations are more powerful than isolated sentence representations. (iii) ABCNNs achieve state-of-the-art performance on AS, PI and TE tasks. We release code at: `https://github.com/yinwenpeng/Answer_Selection`.

1 Introduction

How to model a pair of sentences is a critical issue in many NLP tasks such as answer selection (AS) (Yu et al., 2014; Feng et al., 2015), paraphrase identification (PI) (Madnani et al., 2012; Yin and Schütze, 2015a), textual entailment (TE) (Marelli et al., 2014a; Bowman et al., 2015a) etc.

AS	$\mathbf{s_0}$	how much did Waterboy *gross*?
	$\mathbf{s_1^+}$	the movie *earned* \$161.5 million
	$\mathbf{s_1^-}$	this was Jerry Reed's final film appearance
PI	$\mathbf{s_0}$	she struck a deal with RH to pen a book *today*
	$\mathbf{s_1^+}$	she signed a contract with RH to write a book
	$\mathbf{s_1^-}$	she denied *today* that she struck a deal with RH
TE	$\mathbf{s_0}$	an ice skating rink placed *outdoors* is *full of people*
	$\mathbf{s_1^+}$	a *lot of people* are in an ice skating park
	$\mathbf{s_1^-}$	an ice skating rink placed *indoors* is *full of people*

Figure 1: Positive ($<s_0, s_1^+>$) and negative ($<s_0, s_1^->$) examples for AS, PI and TE tasks. RH = Random House

Most prior work derives each sentence's representation separately, rarely considering the impact of the other sentence. This neglects the mutual influence of the two sentences in the context of the task. It also contradicts what humans do when comparing two sentences. We usually focus on key parts of one sentence by extracting parts from the other sentence that are related by identity, synonymy, antonymy and other relations. Thus, human beings model the two sentences together, using the content of one sentence to guide the representation of the other.

Figure 1 demonstrates that each sentence of a pair partially determines which parts of the other sentence we must focus on. For AS, correctly answering s_0 requires attention on "gross": s_1^+ contains a corresponding unit ("earned") while s_1^- does not. For PI, focus should be removed from "today" to correctly recognize $< s_0, s_1^+ >$ as paraphrases and $< s_0, s_1^- >$ as non-paraphrases. For TE, we need to focus on "full of people" (to recognize TE for $<s_0, s_1^+>$) and on "outdoors" / "indoors" (to recognize non-TE for $<s_0, s_1^->$). These examples show the need for an architecture that computes different representations of s_i for different s_{1-i} ($i \in \{0, 1\}$).

Convolutional Neural Networks (CNNs) (LeCun et al., 1998) are widely used to model sentences (Kalchbrenner et al., 2014; Kim, 2014) and sentence pairs (Socher et al., 2011; Yin and Schütze, 2015a), especially in classification tasks. CNNs are supposed to be good at extracting robust and abstract features of input. This work presents the ABCNN, an attention-based convolutional neural network, that has a powerful mechanism for modeling a sentence pair by taking into account the interdependence between the two sentences. The ABCNN is a general architecture that can handle a wide variety of sentence pair modeling tasks.

Some prior work proposes simple mechanisms that can be interpreted as controlling varying attention; e.g., Yih et al. (2013) employ word alignment to match related parts of the two sentences. In contrast, our attention scheme based on CNNs models relatedness between two parts fully automatically. Moreover, attention at multiple levels of granularity, not only at word level, is achieved as we stack multiple convolution layers that increase abstraction.

Prior work on attention in deep learning (DL) mostly addresses long short-term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997). LSTMs achieve attention usually in a word-to-word scheme, and word representations mostly encode the *whole context* within the sentence (Bahdanau et al., 2015; Rocktäschel et al., 2016). It is not clear whether this is the best strategy; e.g., in the AS example in Figure 1, it is possible to determine that "how much" in s_0 matches "\$161.5 million" in s_1 without taking the entire sentence contexts into account. This observation was also investigated by Yao et al. (2013b) where an information retrieval system retrieves sentences with tokens labeled as DATE by named entity recognition or as CD by POS tagging if there is a "when" question. However, labels or POS tags require extra tools. CNNs benefit from incorporating attention into representations of *local phrases* detected by filters; in contrast, LSTMs encode the *whole context* to form attention-based word representations – a strategy that is more complex than the CNN strategy and (as our experiments suggest) performs less well for some tasks.

Apart from these differences, it is clear that attention has as much potential for CNNs as it does for LSTMs. As far as we know, this is the first NLP paper that incorporates attention into CNNs. Our ABCNNs get state-of-the-art in AS and TE tasks, and competitive performance in PI, then obtains further improvements over all three tasks when linguistic features are used.

2 Related Work

Non-DL on Sentence Pair Modeling. Sentence pair modeling has attracted lots of attention in the past decades. Many tasks can be reduced to a semantic text matching problem. Due to the variety of word choices and inherent ambiguities in natural language, bag-of-word approaches with simple surface-form word matching tend to produce brittle results with poor prediction accuracy (Bilotti et al., 2007). As a result, researchers put more emphasis on exploiting syntactic and semantic structure. Representative examples include methods based on deeper semantic analysis (Shen and Lapata, 2007; Moldovan et al., 2007), tree edit-distance (Punyakanok et al., 2004; Heilman and Smith, 2010) and quasi-synchronous grammars (Wang et al., 2007) that match the dependency parse trees of the two sentences. Instead of focusing on the high-level semantic representation, Yih et al. (2013) turn their attention to improving the shallow semantic component, lexical semantics, by performing semantic matching based on a latent word-alignment structure (cf. Chang et al. (2010)). Lai and Hockenmaier (2014) explore finer-grained word overlap and alignment between two sentences using negation, hypernym, synonym and antonym relations. Yao et al. (2013a) extend word-to-word alignment to phrase-to-phrase alignment by a semi-Markov CRF. However, such approaches often require more computational resources. In addition, employing syntactic or semantic parsers – which produce errors on many sentences – to find the best match between the structured representations of two sentences is not trivial.

DL on Sentence Pair Modeling. To address some of the challenges of non-DL work, much recent work uses neural networks to model sentence pairs for AS, PI and TE.

For AS, Yu et al. (2014) present a bigram CNN to model question and answer candidates. Yang et al. (2015) extend this method and get state-of-the-art performance on the WikiQA dataset (Section 5.1).

Feng et al. (2015) test various setups of a bi-CNN architecture on an insurance domain QA dataset. Tan et al. (2016) explore bidirectional LSTMs on the same dataset. Our approach is different because we do not model the sentences by two independent neural networks in parallel, but instead as an interdependent sentence pair, using attention.

For PI, Blacoe and Lapata (2012) form sentence representations by summing up word embeddings. Socher et al. (2011) use recursive autoencoders (RAEs) to model representations of local phrases in sentences, then pool similarity values of phrases from the two sentences as features for binary classification. Yin and Schütze (2015a) similarly replace an RAE with a CNN. In all three papers, the representation of one sentence is not influenced by the other – in contrast to our attention-based model.

For TE, Bowman et al. (2015b) use recursive neural networks to encode entailment on SICK (Marelli et al., 2014b). Rocktäschel et al. (2016) present an attention-based LSTM for the Stanford natural language inference corpus (Bowman et al., 2015a). Our system is the first CNN-based work on TE.

Some prior work aims to solve a general sentence matching problem. Hu et al. (2014) present two CNN architectures, ARC-I and ARC-II, for sentence matching. ARC-I focuses on sentence representation learning while ARC-II focuses on matching features on phrase level. Both systems were tested on PI, sentence completion (SC) and tweet-response matching. Yin and Schütze (2015b) propose the MultiGranCNN architecture to model general sentence matching based on phrase matching on multiple levels of granularity and get promising results for PI and SC. Wan et al. (2016) try to match two sentences in AS and SC by multiple sentence representations, each coming from the local representations of two LSTMs. Our work is the first one to investigate attention for the general sentence matching task.

Attention-Based DL in Non-NLP Domains. Even though there is little if any work on attention mechanisms in CNNs for NLP, attention-based CNNs have been used in computer vision for visual question answering (Chen et al., 2015), image classification (Xiao et al., 2015), caption generation (Xu et al., 2015), image segmentation (Hong et al., 2016) and object localization (Cao et al., 2015).

symbol	description
s, s_0, s_1	sentence or sentence length
v	word
w	filter width
d_i	dimensionality of input to layer $i+1$
$\mathbf{W}$	weight matrix

Table 1: Notation

Mnih et al. (2014) apply attention in recurrent neural networks (RNNs) to extract information from an image or video by adaptively selecting a sequence of regions or locations and only processing the selected regions at high resolution. Gregor et al. (2015) combine a spatial attention mechanism with RNNs for image generation. Ba et al. (2015) investigate attention-based RNNs for recognizing multiple objects in images. Chorowski et al. (2014) and Chorowski et al. (2015) use attention in RNNs for speech recognition.

Attention-Based DL in NLP. Attention-based DL systems have been applied to NLP after their success in computer vision and speech recognition. They mainly rely on RNNs and end-to-end encoder-decoders for tasks such as machine translation (Bahdanau et al., 2015; Luong et al., 2015) and text reconstruction (Li et al., 2015; Rush et al., 2015). Our work takes the lead in exploring attention mechanisms in CNNs for NLP tasks.

3 BCNN: Basic Bi-CNN

We now introduce our basic (non-attention) CNN that is based on the Siamese architecture (Bromley et al., 1993), i.e., it consists of two weight-sharing CNNs, each processing one of the two sentences, and a final layer that solves the sentence pair task. See Figure 2. We refer to this architecture as the *BCNN*. The next section will then introduce the ABCNN, an attention architecture that extends the BCNN. Table 1 gives our notational conventions.

In our implementation and also in the mathematical formalization of the model given below, we pad the two sentences to have the same length $s = \max(s_0, s_1)$. However, in the figures we show different lengths because this gives a better intuition of how the model works.

We now describe the BCNN's four types of layers: input, convolution, average pooling and output.

Input layer. In the example in the figure, the two input sentences have 5 and 7 words, respectively.

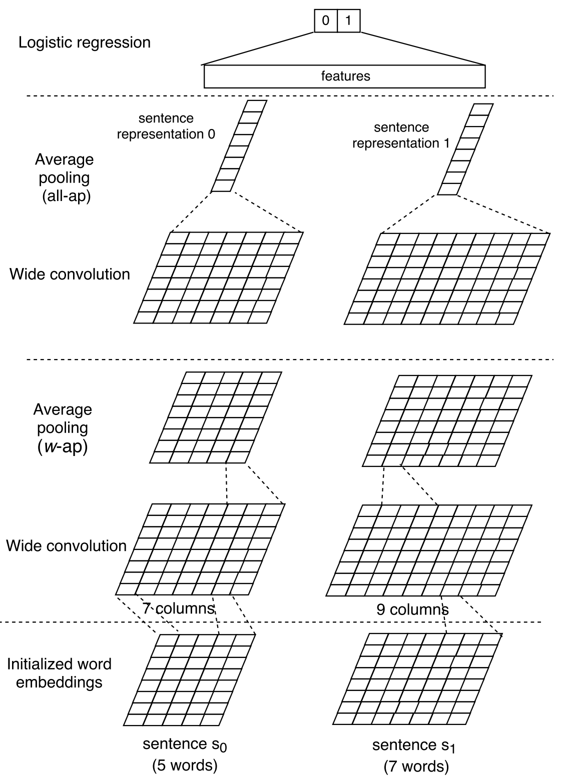


Figure 2: BCNN: ABCNN without Attention

Each word is represented as a d_0-dimensional precomputed word2vec (Mikolov et al., 2013) embedding, $d_0 = 300$. As a result, each sentence is represented as a feature map of dimension $d_0 \times s$.

Convolution layer. Let $v_1, v_2, \ldots, v_s$ be the words of a sentence and $\mathbf{c}_i \in \mathbb{R}^{w \cdot d_0}$, $0 < i < s + w$, the concatenated embeddings of $v_{i-w+1}, \ldots, v_i$ where embeddings for v_j are set to zero when $j < 1$ or $j > s$. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^{d_1}$ for the *phrase* $v_{i-w+1}, \ldots, v_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d_1 \times w d_0}$ as follows:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b})$$

where $\mathbf{b} \in \mathbb{R}^{d_1}$ is the bias.

Average pooling layer. Pooling (including min, max, average pooling) is commonly used to extract robust features from convolution. In this paper, we introduce attention weighting as an alternative, but use average pooling as a baseline as follows.

For the output feature map of the last convolution layer, we do column-wise averaging over *all columns*, denoted as *all-ap*. This generates a representation vector for each of the two sentences, shown as the top "Average pooling (*all-ap*)" layer below "Logistic regression" in Figure 2. These two vectors are the basis for the sentence pair decision.

For the output feature map of non-final convolution layers, we do column-wise averaging over *windows of w consecutive columns*, denoted as *w-ap*; shown as the lower "Average pooling (*w-ap*)" layer in Figure 2. For filter width w, a convolution layer transforms an input feature map of s columns into a new feature map of $s + w - 1$ columns; average pooling transforms this back to s columns. This architecture supports stacking an arbitrary number of convolution-pooling blocks to extract increasingly abstract features. Input features to the bottom layer are words, input features to the next layer are short phrases and so on. Each level generates more abstract features of higher granularity.

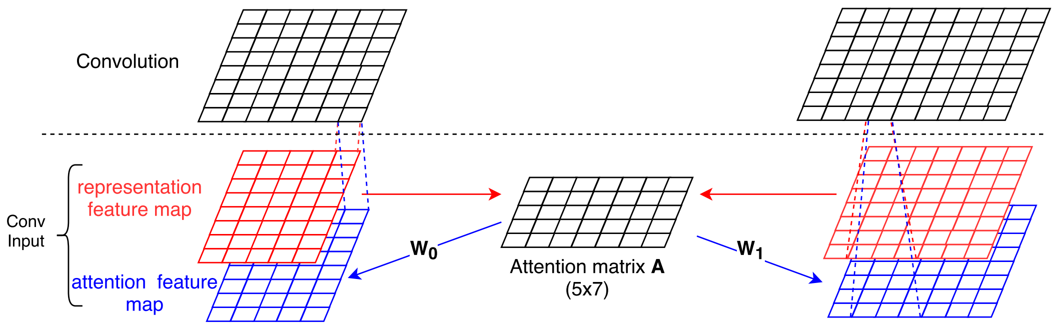
The last layer is an **output layer**, chosen according to the task; e.g., for binary classification tasks, this layer is logistic regression (see Figure 2). Other types of output layers are introduced below.

We found that in most cases, performance is boosted if we provide the output of *all pooling layers* as input to the output layer. For each non-final average pooling layer, we perform *w-ap* (pooling over windows of w columns) as described above, but we also perform *all-ap* (pooling over all columns) and forward the result to the output layer. This improves performance because representations from different layers cover the properties of the sentences at different levels of abstraction and all of these levels can be important for a particular sentence pair.

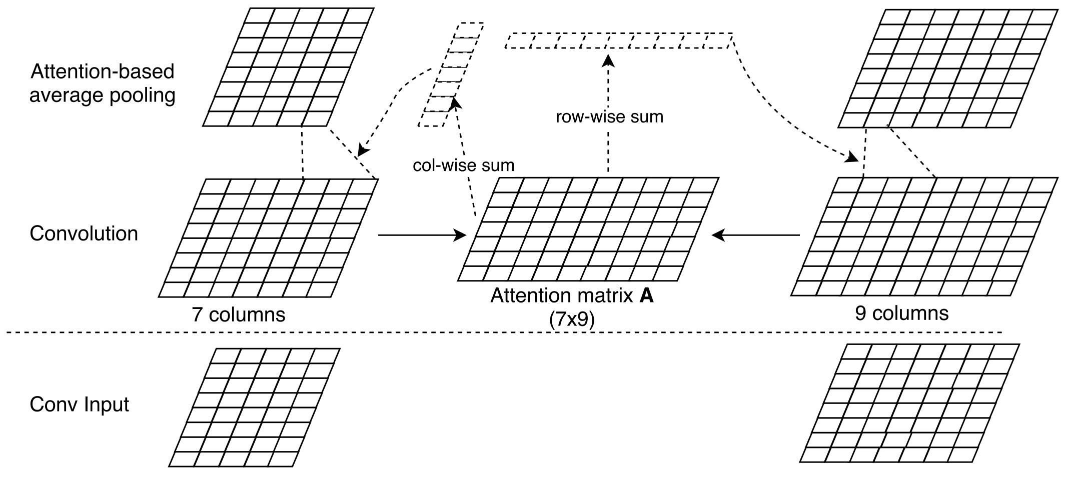
4 ABCNN: Attention-Based BCNN

We now describe three architectures based on the BCNN, the ABCNN-1, the ABCNN-2 and the ABCNN-3, that each introduces an attention mechanism for modeling sentence pairs; see Figure 3.

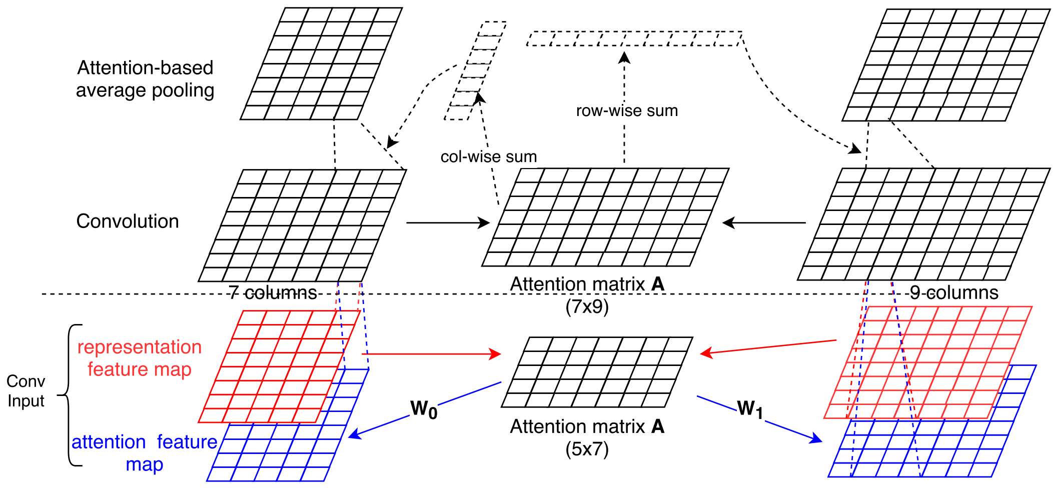
ABCNN-1. The ABCNN-1 (Figure 3(a)) employs an attention feature matrix $\mathbf{A}$ to influence convolution. Attention features are intended to weight those units of s_i more highly in convolution that are relevant to a unit of s_{1-i} ($i \in \{0, 1\}$); we use the term "unit" here to refer to words on the lowest level and to phrases on higher levels of the network. Figure 3(a) shows two *unit representation feature maps* in red: this part of the ABCNN-1 is the same as in the BCNN (see Figure 2). Each column is the



(a) One block in ABCNN-1



(b) One block in ABCNN-2



(c) One block in ABCNN-3

Figure 3: Three ABCNN architectures

representation of a unit, a word on the lowest level and a phrase on higher levels. We first describe the attention feature matrix $\mathbf{A}$ informally (layer "Conv input", middle column, in Figure 3(a)). $\mathbf{A}$ is generated by matching units of the left representation feature map with units of the right representation feature map such that the attention values of row i in $\mathbf{A}$ denote the attention distribution of the i-th unit of s_0 with respect to s_1, and the attention values of column j in $\mathbf{A}$ denote the attention distribution of the j-th unit of s_1 with respect to s_0. $\mathbf{A}$ can be viewed as a new feature map of s_0 (resp. s_1) in row (resp. column) direction because each row (resp. column) is a new feature vector of a unit in s_0 (resp. s_1). Thus, it makes sense to combine this new feature map with the representation feature maps and use both as input to the convolution operation. We achieve this by transforming $\mathbf{A}$ into the two blue matrices in Figure 3(a) that have the same format as the representation feature maps. As a result, the new input of convolution has two feature maps for each sentence (shown in red and blue). Our motivation is that the attention feature map will guide the convolution to learn "counterpart-biased" sentence representations.

More formally, let $\mathbf{F}_{i,r} \in \mathbf{R}^{d\times s}$ be the *representation feature map* of sentence i ($i \in \{0, 1\}$). Then we define the attention matrix $\mathbf{A} \in \mathbf{R}^{s\times s}$ as follows:

$$\mathbf{A}_{i,j} = \text{match-score}(\mathbf{F}_{0,r}[:, i], \mathbf{F}_{1,r}[:, j]) \qquad (1)$$

The function match-score can be defined in a variety of ways. We found that $1/(1 + |x - y|)$ works well where $|\cdot|$ is Euclidean distance.

Given attention matrix $\mathbf{A}$, we generate the *attention feature map* $\mathbf{F}_{i,a}$ for s_i as follows:

$$\mathbf{F}_{0,a} = \mathbf{W}_0 \cdot \mathbf{A}^\top, \quad \mathbf{F}_{1,a} = \mathbf{W}_1 \cdot \mathbf{A}$$

The weight matrices $\mathbf{W}_0 \in \mathbf{R}^{d\times s}$, $\mathbf{W}_1 \in \mathbf{R}^{d\times s}$ are parameters of the model to be learned in training.[1]

We stack the representation feature map $\mathbf{F}_{i,r}$ and the attention feature map $\mathbf{F}_{i,a}$ as an order 3 tensor and feed it into convolution to generate a higher-level representation feature map for s_i ($i \in \{0, 1\}$). In Figure 3(a), s_0 has 5 units, s_1 has 7. The output of convolution (shown in the top layer, filter width $w = 3$) is a higher-level representation feature map with 7 columns for s_0 and 9 columns for s_1.

[1]The weights of the two matrices are shared in our implementation to reduce the number of parameters of the model.

ABCNN-2. The ABCNN-1 computes attention weights *directly on the input representation* with the aim of *improving the features computed by convolution*. The ABCNN-2 (Figure 3(b)) instead computes attention weights *on the output of convolution* with the aim of *reweighting this convolution output*. In the example shown in Figure 3(b), the feature maps output by convolution for s_0 and s_1 (layer marked "Convolution" in Figure 3(b)) have 7 and 9 columns, respectively; each column is the representation of a unit. The attention matrix $\mathbf{A}$ compares all units in s_0 with all units of s_1. We sum all attention values for a unit to derive a single attention weight for that unit. This corresponds to summing all values in a row of $\mathbf{A}$ for s_0 ("col-wise sum", resulting in the column vector of size 7 shown) and summing all values in a column for s_1 ("row-wise sum", resulting in the row vector of size 9 shown).

More formally, let $\mathbf{A} \in \mathbf{R}^{s\times s}$ be the attention matrix, $a_{0,j} = \sum \mathbf{A}[j, :]$ the attention weight of unit j in s_0, $a_{1,j} = \sum \mathbf{A}[:, j]$ the attention weight of unit j in s_1 and $\mathbf{F}^c_{i,r} \in \mathbf{R}^{d\times(s_i+w-1)}$ the output of convolution for s_i. Then the j-th column of the new feature map $\mathbf{F}^p_{i,r}$ generated by *w-ap* is derived by:

$$\mathbf{F}^p_{i,r}[:, j] = \sum_{k=j:j+w} a_{i,k}\mathbf{F}^c_{i,r}[:, k], \quad j = 1 \ldots s_i$$

Note that $\mathbf{F}^p_{i,r} \in \mathbf{R}^{d\times s_i}$, i.e., ABCNN-2 pooling generates an output feature map of the same size as the input feature map of convolution. This allows us to stack multiple convolution-pooling blocks to extract features of increasing abstraction.

There are three main differences between the ABCNN-1 and the ABCNN-2. (i) Attention in the ABCNN-1 impacts *convolution indirectly* while attention in the ABCNN-2 influences *pooling* through *direct* attention weighting. (ii) The ABCNN-1 requires the two matrices $\mathbf{W}_i$ to convert the attention matrix into attention feature maps; and the input to convolution has two times as many feature maps. Thus, the ABCNN-1 has more parameters than the ABCNN-2 and is more vulnerable to overfitting. (iii) As pooling is performed after convolution, pooling handles larger-granularity units than convolution; e.g., if the input to convolution has word level granularity, then the input to pooling has phrase level granularity, the phrase size being equal to filter size w. Thus, the ABCNN-1 and the ABCNN-2 implement attention mechanisms for linguistic units of different granularity. The complementarity of the ABCNN-1 and the ABCNN-2 motivates us to propose the ABCNN-3, a third architecture that combines elements of the two.

ABCNN-3 (Figure 3(c)) combines the ABCNN-1 and the ABCNN-2 by stacking them; it combines the strengths of the ABCNN-1 and -2 by allowing the attention mechanism to operate (i) both on the convolution and on the pooling parts of a convolution-pooling block and (ii) both on the input granularity and on the more abstract output granularity.

5 Experiments

We test the proposed architectures on three tasks: answer selection (AS), paraphrase identification (PI) and textual entailment (TE).

Common Training Setup. Words are initialized by 300-dimensional word2vec embeddings and not changed during training. A single randomly initialized embedding is created for all unknown words by uniform sampling from [-.01,.01]. We employ Adagrad (Duchi et al., 2011) and L_2 regularization.

Network Configuration. Each network in the experiments below consists of (i) an initialization block b_1 that initializes words by word2vec embeddings, (ii) a stack of $k-1$ convolution-pooling blocks $b_2, \dots, b_k$, computing increasingly abstract features, and (iii) one final *LR layer* (logistic regression layer) as shown in Figure 2.

The input to the LR layer consists of kn features – each block provides n similarity scores, e.g., n cosine similarity scores. Figure 2 shows the two sentence vectors output by the final block b_k of the stack ("sentence representation 0", "sentence representation 1"); this is the basis of the last n similarity scores. As we explained in the final paragraph of Section 3, we perform *all-ap* pooling for *all blocks*, not just for b_k. Thus we get one sentence representation each for s_0 and s_1 for each block $b_1, \dots, b_k$. We compute n similarity scores for each block (based on the block's two sentence representations). Thus, we compute a total of kn similarity scores and these scores are input to the LR layer.

		AS			PI			TE		
	#CL	lr	w	L_2	lr	w	L_2	lr	w	L_2
ABCNN-1	1	.08	4	.0004	.08	3	.0002	.08	3	.0006
ABCNN-1	2	.085	4	.0006	.085	3	.0003	.085	3	.0006
ABCNN-2	1	.05	4	.0003	.085	3	.0001	.09	3	.00065
ABCNN-2	2	.06	4	.0006	.085	3	.0001	.085	3	.0007
ABCNN-3	1	.05	4	.0003	.05	3	.0003	.09	3	.0007
ABCNN-3	2	.06	4	.0006	.055	3	.0005	.09	3	.0007

Table 2: Hyperparameters. lr: learning rate. #CL: number convolution layers. w: filter width. The number of convolution kernels d_i ($i > 0$) is 50 throughout.

Depending on the task, we use different methods for computing the similarity score: see below.

Layerwise Training. In our training regime, we first train a network consisting of just one convolution-pooling block b_2. We then create a new network by adding a block b_3, initialize its b_2 block with the previously learned weights for b_2 and train b_3 keeping the previously learned weights for b_2 fixed. We repeat this procedure until all $k-1$ convolution-pooling blocks are trained. We found that this training regime gives us good performance and shortens training times considerably. Since similarity scores of lower blocks are kept unchanged once they have been learned, this also has the nice effect that "simple" similarity scores (those based on surface features) are learned first and subsequent training phases can focus on complementary scores derived from more complex abstract features.

Classifier. We found that performance increases if we do not use the output of the LR layer as the final decision, but instead train a linear SVM or a logistic regression with default parameters[2] directly on the input to the LR layer (i.e., on the kn similarity scores that are generated by the k-block stack after network training is completed). Direct training of SVMs/LR seems to get closer to the global optimum than gradient descent training of CNNs.

Table 2 shows hyperparameters, tuned on dev.

We use addition and LSTMs as two **shared baselines** for all three tasks, i.e., for AS, PI and TE. We now describe these two shared baselines.

(i) **Addition**. We sum up word embeddings element-wise to form each sentence representation. The classifier input is then the concatenation of the two sentence representations. (ii) **A-LSTM**. Before this work, most attention mechanisms in NLP

[2] `http://scikit-learn.org/stable/` for both.

were implemented in recurrent neural networks for text generation tasks such as machine translation (e.g., Bahdanau et al. (2015), Luong et al. (2015)). Rocktäschel et al. (2016) present an attention-LSTM for natural language inference. Since this model is the pioneering attention based RNN system for sentence pair classification, we consider it as a baseline system ("A-LSTM") for all our three tasks. The A-LSTM has the same configuration as our ABCNNs in terms of word initialization (300-dimensional word2vec embeddings) and the dimensionality of all hidden layers (50).

5.1 Answer Selection

We use WikiQA,[3] an open domain question-answer dataset. We use the subtask that assumes that there is at least one correct answer for a question. The corresponding dataset consists of 20,360 question-candidate pairs in train, 1,130 pairs in dev and 2,352 pairs in test where we adopt the standard setup of only considering questions with correct answers in test. Following Yang et al. (2015), we truncate answers to 40 tokens.

The task is to rank the candidate answers based on their relatedness to the question. Evaluation measures are mean average precision (MAP) and mean reciprocal rank (MRR).

Task-Specific Setup. We use cosine similarity as the similarity score for AS. In addition, we use sentence lengths, *WordCnt* (count of the number of non-stopwords in the question that also occur in the answer) and *WgtWordCnt* (reweight the counts by the IDF values of the question words). Thus, the final input to the LR layer has size $k + 4$: one cosine for each of the k blocks and the four additional features.

We compare with seven **baselines**. The first three are considered by Yang et al. (2015): (i) WordCnt; (ii) WgtWordCnt; (iii) CNN-Cnt (the state-of-the-art system): combine CNN with (i) and (ii). Apart from the baselines considered by Yang et al. (2015), we compare with two Addition baselines and two LSTM baselines. Addition and A-LSTM are the *shared baselines* described before. We also combine both with the four extra features; this gives us two additional baselines that we refer to as Addition(+) and A-LSTM(+).

[3]`http://aka.ms/WikiQA` (Yang et al., 2015)

	method	MAP	MRR
Baselines	WordCnt	0.4891	0.4924
	WgtWordCnt	0.5099	0.5132
	CNN-Cnt	<u>0.6520</u>	<u>0.6652</u>
	Addition	0.5021	0.5069
	Addition(+)	0.5888	0.5929
	A-LSTM	0.5347	0.5483
	A-LSTM(+)	0.6381	0.6537
BCNN	one-conv	0.6629	0.6813
	two-conv	0.6593	0.6738
ABCNN-1	one-conv	0.6810*	0.6979*
	two-conv	0.6855*	0.7023*
ABCNN-2	one-conv	0.6885*	0.7054*
	two-conv	0.6879*	0.7068*
ABCNN-3	one-conv	0.6914*	**0.7127***
	two-conv	**0.6921***	0.7108*

Table 3: Results on WikiQA. Best result per column is bold. Significant improvements over state-of-the-art baselines (underlined) are marked with $*$ (t-test, p $<$ .05).

Results. Table 3 shows performance of the baselines, of the BCNN and of the three ABCNNs. For CNNs, we test one (one-conv) and two (two-conv) convolution-pooling blocks.

The non-attention network BCNN already performs better than the baselines. If we add attention mechanisms, then the performance further improves by several points. Comparing the ABCNN-2 with the ABCNN-1, we find the ABCNN-2 is slightly better even though the ABCNN-2 is the simpler architecture. If we combine the ABCNN-1 and the ABCNN-2 to form the ABCNN-3, we get further improvement.[4]

This can be explained by the ABCNN-3's ability to take attention of finer-grained granularity into consideration in each convolution-pooling block while the ABCNN-1 and the ABCNN-2 consider attention only at convolution input or only at pooling input, respectively. We also find that stacking two convolution-pooling blocks does not bring consistent improvement and therefore do not test deeper architectures.

5.2 Paraphrase Identification

We use the Microsoft Research Paraphrase (MSRP) corpus (Dolan et al., 2004). The training set contains 2753 true / 1323 false and the test set 1147 true / 578 false paraphrase pairs. We randomly select 400

[4]If we limit the input to the LR layer to the k similarity scores in the ABCNN-3 (two-conv), results are .660 (MAP) / .677 (MRR).

pairs from train and use them as dev; but we still report results for training on the entire training set. For each triple (label, s_0, s_1) in the training set, we also add (label, s_1, s_0) to the training set to make best use of the training data. Systems are evaluated by accuracy and F_1.

Task-Specific Setup. In this task, we add the 15 MT features from (Madnani et al., 2012) and the lengths of the two sentences. In addition, we compute ROUGE-1, ROUGE-2 and ROUGE-SU4 (Lin, 2004), which are scores measuring the match between the two sentences on (i) unigrams, (ii) bigrams and (iii) unigrams and skip-bigrams (maximum skip distance of four), respectively. In this task, we found transforming Euclidean distance into similarity score by $1/(1 + |x - y|)$ performs better than cosine similarity. Additionally, we use dynamic pooling (Yin and Schütze, 2015a) of the attention matrix $\mathbf{A}$ in Equation (1) and forward pooled values of all blocks to the classifier. This gives us better performance than only forwarding sentence-level matching features.

We compare our system with representative DL approaches: (i) A-LSTM; (ii) A-LSTM(+): A-LSTM plus handcrafted features; (iii) RAE (Socher et al., 2011), recursive autoencoder; (iv) Bi-CNN-MI (Yin and Schütze, 2015a), a bi-CNN architecture; and (v) MPSSM-CNN (He et al., 2015), the state-of-the-art NN system for PI, and the following four non-DL systems: (vi) Addition; (vii) Addition(+): Addition plus handcrafted features; (viii) MT (Madnani et al., 2012), a system that combines machine translation metrics;[5] (ix) MF-TF-KLD (Ji and Eisenstein, 2013), the state-of-the-art non-NN system.

Results. Table 4 shows that the BCNN is slightly worse than the state-of-the-art whereas the ABCNN-1 roughly matches it. The ABCNN-2 is slightly above the state-of-the-art. The ABCNN-3 outperforms the state-of-the-art in accuracy and F_1.[6] Two convolution layers only bring small improvements over one.

[5]For better comparability of approaches in our experiments, we use a simple SVM classifier, which performs slightly worse than Madnani et al. (2012)'s more complex meta-classifier.

[6]Improvement of .3 (acc) and .1 (F_1) over state-of-the-art is not significant. The ABCNN-3 (two-conv) without "linguistic" features (i.e., MT and ROUGE) achieves 75.1/82.7.

method		acc	F_1
Baselines	majority voting	66.5	79.9
	RAE	76.8	83.6
	Bi-CNN-MI	78.4	84.6
	MPSSM-CNN	78.6	84.7
	MT	76.8	83.8
	MF-TF-KLD	78.6	84.6
	Addition	70.8	80.9
	Addition (+)	77.3	84.1
	A-LSTM	69.5	80.1
	A-LSTM (+)	77.1	84.0
BCNN	one-conv	78.1	84.1
	two-conv	78.3	84.3
ABCNN-1	one-conv	78.5	84.5
	two-conv	78.5	84.6
ABCNN-2	one-conv	78.6	84.7
	two-conv	78.8	84.7
ABCNN-3	one-conv	78.8	**84.8**
	two-conv	**78.9**	**84.8**

Table 4: Results for PI on MSRP

5.3 Textual Entailment

SemEval 2014 Task 1 (Marelli et al., 2014a) evaluates system predictions of textual entailment (TE) relations on sentence pairs from the SICK dataset (Marelli et al., 2014b). The three classes are entailment, contradiction and neutral. The sizes of SICK train, dev and test sets are 4439, 495 and 4906 pairs, respectively. We call this dataset ORIG.

We also create NONOVER, a copy of ORIG in which *words occurring in both sentences are removed*. A sentence in NONOVER is denoted by the special token <empty> if all words are removed. Table 5 shows three pairs from ORIG and their transformation in NONOVER. We observe that focusing on the non-overlapping parts provides clearer hints for TE than ORIG. In this task, we run two copies of each network, one for ORIG, one for NONOVER; these two networks have a single common LR layer.

Like Lai and Hockenmaier (2014), we train our final system (after fixing hyperparameters) on train and dev (4934 pairs). Eval measure is accuracy.

Task-Specific Setup. We found that for this task forwarding two similarity scores from each block (instead of just one) is helpful. We use cosine similarity and Euclidean distance. As we did for paraphrase identification, we add the 15 MT features for each sentence pair for this task as well; our motivation is that entailed sentences resemble paraphrases more than contradictory sentences do.

	ORIG	NONOVER
0	children in red shirts are playing in the leaves	children red shirts playing
	three kids are sitting in the leaves	three kids sitting
1	three boys are jumping in the leaves	boys
	three kids are jumping in the leaves	kids
2	a man is jumping into an empty pool	an empty
	a man is jumping into a full pool	a full

Table 5: SICK data: Converting the original sentences (ORIG) into the NONOVER format

We use the following linguistic features. **Negation** is important for detecting contradiction. Feature NEG is set to 1 if either sentence contains "no", "not", "nobody", "isn't" and to 0 otherwise. Following Lai and Hockenmaier (2014), we use WordNet (Miller, 1995) to detect **nyms**: synonyms, hypernyms and antonyms in the pairs. But we do this on NONOVER (not on ORIG) to focus on what is critical for TE. Specifically, feature SYN is the number of word pairs in s_0 and s_1 that are synonyms. HYP0 (resp. HYP1) is the number of words in s_0 (resp. s_1) that have a hypernym in s_1 (resp. s_0). In addition, we collect all *potential antonym pairs* (PAP) in NONOVER. We identify the matched chunks that occur in *contradictory* and *neutral*, but not in *entailed* pairs. We exclude synonyms and hypernyms and apply a frequency filter of $n = 2$. In contrast to Lai and Hockenmaier (2014), we constrain the PAP pairs to cosine similarity above 0.4 in word2vec embedding space as this discards many noise pairs. Feature ANT is the number of matched PAP antonyms in a sentence pair. As before we use sentence **lengths**, both for ORIG (LEN0O: length s_0, LEN1O: length s_1) and for NONOVER (LEN0N: length s_0, LEN1N: length s_1).

On the whole, we have 24 extra features: 15 MT metrics, NEG, SYN, HYP0, HYP1, ANT, LEN0O, LEN1O, LEN0N and LEN1N.

Apart from the Addition and LSTM baselines, we further compare with the top-3 systems in SemEval and TrRNTN (Bowman et al., 2015b), a recursive neural network developed for this SICK task.

Results. Table 6 shows that our CNNs outperform A-LSTM (with or without linguistic features added) and the top three SemEval systems. Comparing ABCNNs with the BCNN, attention mechanisms consistently improve performance. The ABCNN-1 has performance comparable to the ABCNN-2 while the ABCNN-3 is better still: a boost of 1.6 points compared to the previous state of the art.[7]

method		acc
Sem-Eval Top3	(Jimenez et al., 2014)	83.1
	(Zhao et al., 2014)	83.6
	(Lai and Hockenmaier, 2014)	84.6
TrRNTN	(Bowman et al., 2015b)	76.9
Addition	no features	73.1
	plus features	79.4
A-LSTM	no features	78.0
	plus features	81.7
BCNN	one-conv	84.8
	two-conv	85.0
ABCNN-1	one-conv	85.6
	two-conv	85.8
ABCNN-2	one-conv	85.7
	two-conv	85.8
ABCNN-3	one-conv	86.0*
	two-conv	**86.2***

Table 6: Results on SICK. Significant improvements over (Lai and Hockenmaier, 2014) are marked with $*$ (test of equal proportions, $p < .05$).

Visual Analysis. Figure 4 visualizes the attention matrices for one TE sentence pair in the ABCNN-2 for blocks b_1 (unigrams), b_2 (first convolutional layer) and b_3 (second convolutional layer). Darker shades of blue indicate stronger attention values.

In Figure 4 (top), each word corresponds to exactly one row or column. We can see that words in s_i with semantic equivalents in s_{1-i} get high attention while words without semantic equivalents get low attention, e.g., "walking" and "murals" in s_0 and "front" and "colorful" in s_1. This behavior seems reasonable for the unigram level.

Rows/columns of the attention matrix in Figure 4 (middle) correspond to phrases of length three since filter width $w = 3$. High attention values generally correlate with close semantic correspondence: the phrase "people are" in s_0 matches "several people are" in s_1; both "are walking outside" and "walking outside the" in s_0 match "are in front" in s_1; "the building that" in s_0 matches "a colorful building" in s_1. More interestingly, looking at the bottom right corner, both "on it" and "it" in s_0 match "building" in s_1; this indicates that ABCNNs are able to detect some coreference across sentences. "building" in s_1 has two places in which higher attentions appear, one is with "it" in s_0, the other is with "the building

[7]If we run the ABCNN-3 (two-conv) without the 24 linguistic features, performance is 84.6.

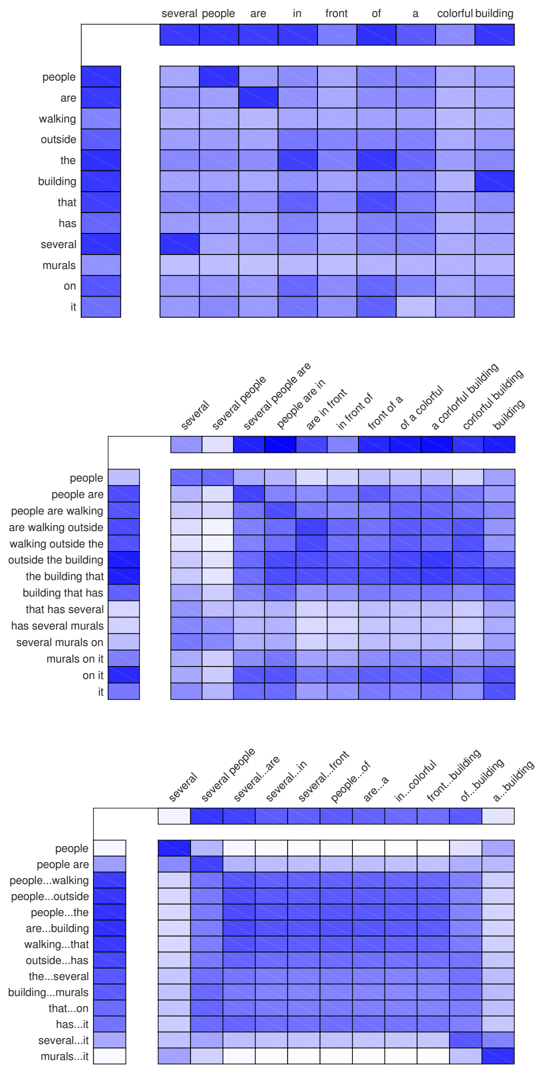


Figure 4: Attention visualization for TE. Top: unigrams, b_1. Middle: conv1, b_2. Bottom: conv2, b_3.

that" in s_0. This may indicate that ABCNNs recognize that "building" in s_1 and "the building that" / "it" in s_0 refer to the same object. Hence, coreference resolution across sentences as well as within a sentence both are detected. For the attention vectors on the left and the top, we can see that attention has focused on the key parts: "people are walking outside the building that" in s_0, "several people are in" and "of a colorful building" in s_1.

Rows/columns of the attention matrix in Figure 4 (bottom, second layer of convolution) correspond to phrases of length 5 since filter width $w = 3$ in both convolution layers ($5 = 1 + 2 * (3 - 1)$). We use "..." to denote words in the middle if a phrase like "several...front" has more than two words. We can see that attention distribution in the matrix has focused on some local regions. As granularity of phrases is larger, it makes sense that the attention values are smoother. But we still can find some interesting clues: at the two ends of the main diagonal, higher attentions hint that the first part of s_0 matches well with the first part of s_1; "several murals on it" in s_0 matches well with "of a colorful building" in s_1, which satisfies the intuition that these two phrases are crucial for making a decision on TE in this case. This again shows the potential strength of our system in figuring out which parts of the two sentences refer to the same object. In addition, in the central part of the matrix, we can see that the long phrase "people are walking outside the building" in s_0 matches well with the long phrase "are in front of a colorful building" in s_1.

6 Summary

We presented three mechanisms to integrate attention into CNNs for general sentence pair modeling tasks.

Our experiments on AS, PI and TE show that attention-based CNNs perform better than CNNs without attention mechanisms. The ABCNN-2 generally outperforms the ABCNN-1 and the ABCNN-3 surpasses both.

In all tasks, we did not find any big improvement of two layers of convolution over one layer. This is probably due to the limited size of training data. We expect that, as larger training sets become available, deep ABCNNs will show even better performance.

In addition, linguistic features contribute in all three tasks: improvements by 0.0321 (MAP) and 0.0338 (MRR) for AS, improvements by 3.8 (acc) and 2.1 (F_1) for PI and an improvement by 1.6 (acc) for TE. But our ABCNNs can still reach or surpass state-of-the-art even without those features in AS and TE tasks. This indicates that ABCNNs are generally strong NN systems.

Attention-based LSTMs are especially successful in tasks with a strong *generation component* like machine translation (discussed in Sec. 2). CNNs have not been used for this type of task. This is an interesting area of future work for attention-based CNNs.

Acknowledgments

We gratefully acknowledge the support of Deutsche Forschungsgemeinschaft (DFG): grant SCHU 2246/8-2.

We would like to thank the anonymous reviewers for their helpful comments.

References

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *Proceedings of ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Matthew W. Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of SIGIR*, pages 351–358.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*, pages 546–556.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of CVSC Workshop*, pages 12–21.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using A “siamese” time delay neural network. *IJPRAI*, 7(4):669–688.

Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, Deva Ramanan, and Thomas S. Huang. 2015. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of ICCV*, pages 2956–2964.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of NAACL-HLT*, pages 429–437.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. 2015. ABC-CNN: An attention based convolutional neural network for visual question answering. *CoRR*, abs/1511.05960.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: First results. In *Proceedings of Deep Learning and Representation Learning Workshop, NIPS*.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Proceedings of NIPS*, pages 577–585.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of COLING*, pages 350–356.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Proceedings of IEEE ASRU Workshop*, pages 813–820.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *Proceedings of ICML*, pages 1462–1471.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of EMNLP*, pages 1576–1586.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*, pages 1011–1019.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Seunghoon Hong, Junhyuk Oh, Honglak Lee, and Bohyung Han. 2016. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *Proceedings of CVPR*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*, pages 2042–2050.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of EMNLP*, pages 891–896.

Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of SemEval*, pages 732–742.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of SemEval*, pages 329–334.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*, pages 1106–1115.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL Text Summarization Workshop*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of NAACL-HLT*, pages 182–190.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of SemEval*, pages 1–8.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

George A. Miller. 1995. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Proceedings of NIPS*, pages 2204–2212.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Daniel Hodges. 2007. Cogex: A semantically and contextually enriched logic prover for question answering. *Journal of Applied Logic*, 5(1):49–69.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004 (Special session: Intelligent Text Processing)*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, pages 12–21.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809.

Ming Tan, Bing Xiang, and Bowen Zhou. 2016. LSTM-based deep learning models for non-factoid answer selection. In *Proceedings of ICLR Workshop*.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of AAAI*, pages 2835–2841.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*, pages 22–32.

Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. 2015. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proceedings of CVPR*, pages 842–850.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*, pages 2048–2057.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. Semi-markov phrase-based monolingual alignment. In *Proceedings of EMNLP*, pages 590–600.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. 2013b. Automatic coupling of answer extraction and information retrieval. In *Proceedings of ACL*, pages 159–165.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*, pages 1744–1753.

Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proceedings of NAACL-HLT*, pages 901–911.

Wenpeng Yin and Hinrich Schütze. 2015b. Multi-GranCNN: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of ACL-IJCNLP*, pages 63–73.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *Proceedings of Deep Learning and Representation Learning Workshop, NIPS*.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of SemEval*, pages 271–277.

Chapter 6

Task-Specific Attentive Pooling of Phrase Alignments Contributes to Sentence Matching

Task-Specific Attentive Pooling of Phrase Alignments Contributes to Sentence Matching

Wenpeng Yin, Hinrich Schütze
The Center for Information and Language Processing
LMU Munich, Germany
wenpeng@cis.lmu.de

Abstract

This work studies comparatively two typical sentence matching tasks: textual entailment (TE) and answer selection (AS), observing that weaker phrase alignments are more critical in TE, while stronger phrase alignments deserve more attention in AS. The key to reach this observation lies in phrase detection, phrase representation, phrase alignment, and more importantly how to connect those aligned phrases of different matching degrees with the final classifier.

Prior work (i) has limitations in phrase generation and representation, or (ii) conducts alignment at word and phrase levels by handcrafted features or (iii) utilizes a single framework of alignment without considering the characteristics of specific tasks, which limits the framework's effectiveness across tasks.

We propose an architecture based on Gated Recurrent Unit that supports (i) representation learning of phrases of *arbitrary granularity* and (ii) task-specific attentive pooling of phrase alignments between two sentences. Experimental results on TE and AS match our observation and show the effectiveness of our approach.

1 Introduction

How to model a pair of sentences is a critical issue in many NLP tasks, including textual entailment (Marelli et al., 2014a; Bowman et al., 2015a; Yin et al., 2016a) and answer selection (Yu et al., 2014; Yang et al., 2015; Santos et al., 2016). A key challenge common to these tasks is the lack of explicit alignment annotation between the sentences of the pair. Thus, inferring and assessing the semantic relations between words and phrases in the two sentences is a core issue.

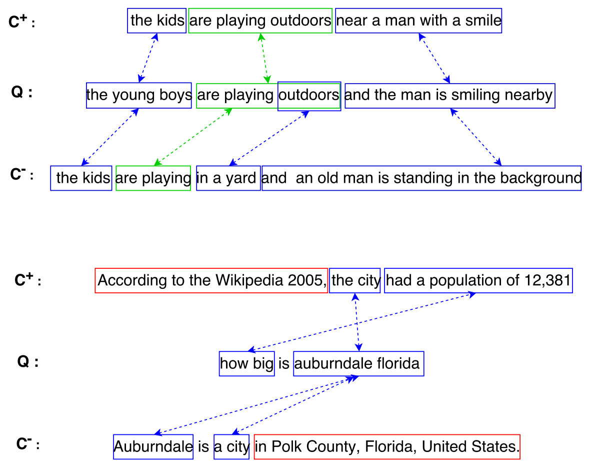


Figure 1: Alignment examples in TE (top) and AS (bottom). Green color: identical (subset) alignment; blue color: relatedness alignment; red color: unrelated alignment. Q: the first sentence in TE or the question in AS; C^+, C^-: the correct or incorrect counterpart in the sentence pair (Q, C).

Figure 1 shows examples of human annotated phrase alignments. In the TE example, we try to figure out Q entails C^+ (positive) or C^- (negative). As human beings, we discover the relationship of two sentences by studying the alignments between linguistic units. We see that some phrases are kept: "are playing outdoors" (between Q and C^+), "are playing " (between Q and C^-). Some phrases are changed into related semantics on purpose: "the young boys" (Q) $\rightarrow$ "the kids" (C^+ & C^-), "the man is smiling nearby" (Q) $\rightarrow$ "near a man with a smile" (C^+) or $\rightarrow$ "an old man is standing in the background" (C^-) . We can see that the kept parts have stronger alignments (green color), and changed parts have weaker alignments (blue color). Here, by "strong" / "weak" we mean how semantically close the two aligned phrases are. To successfully identify the relationships of (Q, C^+) or (Q, C^-), studying the changed parts is crucial. *Hence, we argue that TE should pay more attention to weaker alignments.*

In AS, we try to figure out: does sentence C^+ or sentence C^- answer question Q? Roughly, the content in candidates C^+ and C^- can be classified into aligned part (e.g., repeated or relevant parts) and negligible part. This differs from TE, in which it is hard to claim that some parts are negligible or play a minor role, as TE requires to make clear that each part can entail or be entailed. Hence, TE is considerably sensitive to those "unseen" parts. In contrast, *AS is more tolerant of negligible parts and less related parts.* From the AS example in Figure 1, we see that "Auburndale Florida" (Q) can find related part "the city" (C^+), and "Auburndale", "a city" (C^-) ; "how big" (Q) also matches "had a population of 12,381" (C^+) very well. And some unaligned parts exist, denoted by red color. *Hence, we argue that stronger alignments in AS deserve more attention.*

The above analysis suggests that: (i) alignments connecting two sentences can happen between phrases of arbitrary granularity; (ii) phrase alignments can have different intensities; (iii) tasks of different properties require paying different attention to alignments of different intensities.

Alignments at word level (Yih et al., 2013) or phrase level (Yao et al., 2013) both have been studied before. For example, Yih et al. (2013) make use of WordNet (Miller, 1995) and Probase (Wu et al., 2012) for identifying hyper- and hyponymy. Yao et al. (2013) use POS tags, WordNet and paraphrase database for alignment identification. Their approaches rely on manual feature design and linguistic resources. We develop a deep neural network (DNN) to learn representations of phrases of arbitrary lengths. As a result, alignments can be searched in a more automatic and exhaustive way.

DNNs have been intensively investigated in sentence pair classifications (Blacoe and Lapata, 2012; Socher et al., 2011; Yin and Schütze, 2015b), and attention mechanisms are also applied to individual tasks (Santos et al., 2016; Rocktäschel et al., 2016; Wang and Jiang, 2016); however, most attention-based DNNs have implicit assumption that stronger alignments deserve more attention (Yin et al., 2016a; Santos et al., 2016; Yin et al., 2016b). Our examples in Figure 1, instead, show that this assumption does not hold invariably. Weaker alignments in certain tasks such as TE can be the indicator of the final decision. Our inspiration comes from the analysis of some prior work. For TE, Yin et al. (2016a) show that considering the pairs in which overlapping tokens are removed can give a boost. This simple trick matches our motivation that weaker alignment should be given more attention in TE. However, Yin et al. (2016a) remove overlapping tokens completely, potentially obscuring complex alignment configurations. In addition, Yin et al. (2016a) use the same attention mechanism for TE and AS, which is less optimal based on our observations.

This motivates us in this work to introduce DNNs with a flexible attention mechanism that is adaptable for specific tasks. For TE, it can make our system pay more attention to weaker alignments; for AS, it enables our system to focus on stronger alignments. We can treat the pre-processing in (Yin et al., 2016a) as a hard way, and ours as a soft way, as our phrases have more flexible lengths and the existence of overlapping phrases decreases the risk of losing important alignments. In experiments, we will show that this attention scheme is very effective for different tasks.

We make the following contributions. (i) We use GRU (Gated Recurrent Unit (Cho et al., 2014)) to learn representations for phrases of arbitrary granularity. Based on phrase representations, we can detect phrase alignments of different intensities. (ii) We propose attentive pooling to achieve flexible choice among alignments, depending on the characteristics of the task. (iii) We achieve state-of-the-art on TE task.

2 Related Work

Non-DNN for sentence pair modeling. Heilman and Smith (2010) describe tree edit models that generalize tree edit distance by allowing operations that better account for complex reordering phenomena and by learning from data how different edits should affect the model's decisions about sentence relations. Wang and Manning (2010) cope with the alignment between a sentence pair by using a probabilistic model that models tree-edit operations on dependency parse trees. Their model treats alignments as structured latent variables, and offers a principled framework for incorporating complex linguistic features. Guo and Diab (2012) identify the degree of sentence similarity by modeling the missing words (words that are not in the sentence) so as to relieve the sparseness issue of sentence modeling. Yih et al. (2013) try to improve the shallow semantic component, lexical semantics, by formulating sentence pair as a semantic matching problem with a latent word-alignment structure as in (Chang et al., 2010). More fine-grained word overlap and alignment between two sentences are explored in (Lai and Hockenmaier, 2014), in which negation, hypernym/hyponym, synonym and antonym relations are used. Yao et al. (2013) extend word-to-word alignment to phrase-to-phrase alignment by a semi-Markov CRF. Such approaches often require more computational resources. In addition, using syntactic/semantic parsing during run-time to find the best matching between structured representation of sentences is not trivial.

DNN for sentence pair classification. There recently has been great interest in using DNNs for classifying sentence pairs as they can reduce the burden of feature engineering.

For TE, Bowman et al. (2015b) employ recursive DNN to encode entailment on SICK (Marelli et al., 2014b). Rocktäschel et al. (2016) present an attention-based LSTM (long short-term memory, Hochreiter and Schmidhuber (1997)) for the SNLI corpus (Bowman et al., 2015a).

For AS, Yu et al. (2014) present a bigram CNN (convolutional neural network (LeCun et al., 1998)) to model question and answer candidates. Yang et al. (2015) extend this method and get state-of-the-art performance on the WikiQA dataset. Feng et al. (2015) test various setups of a bi-CNN architecture on an insurance domain QA dataset. Tan et al. (2015) explore bidirectional LSTM on the same dataset. Other sentence matching tasks such as paraphrase identification (Socher et al., 2011; Yin and Schütze, 2015a), question – Freebase fact matching (Yin et al., 2016b) etc. are also investigated.

Some prior work aims to solve a general sentence matching problem. Hu et al. (2014) present two CNN architectures for paraphrasing, sentence completion (SC), tweet-response matching tasks. Yin and Schütze (2015b) propose the MultiGranCNN architecture to model general sentence matching based on phrase matching on multiple levels of granularity. Wan et al. (2016) try to match two sentences in AS and SC by multiple sentence representations, each coming from the local representations of two LSTMs.

Attention-based DNN for alignment. DNNs have been successfully developed to detect alignments, e.g., in machine translation (Bahdanau et al., 2015; Luong et al., 2015) and text reconstruction (Li et al., 2015; Rush et al., 2015). In addition, attention-based alignment is also applied in natural language inference (e.g., Rocktäschel et al. (2016),Wang and Jiang (2016)). However, most of this work aligns word-by-word. As Figure 1 shows, many sentence relations can be better identified through phrase level alignments. This is one motivation of our work.

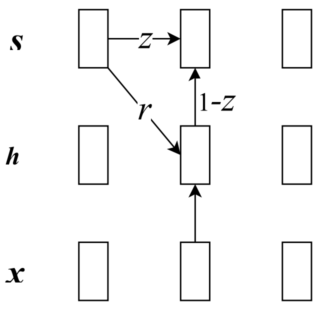


Figure 2: Gated Recurrent Unit

3 Model

This section first gives a brief introduction of GRU and how it performs phrase representation learning, then describes the different attentive poolings for phrase alignments w.r.t TE and AS tasks.

3.1 GRU Introduction

GRU is a simplified version of LSTM. Both are found effective in sequence modeling, as they are order-sensitive and can capture long-range context. The tradeoffs between GRU and its competitor LSTM have not been fully explored yet. According to empirical evaluations in (Chung et al., 2014; Jozefowicz et al., 2015), there is not a clear winner. In many tasks both architectures yield comparable performance and tuning hyper-parameters like layer size is probably more important than picking the ideal architecture. GRU have fewer parameters and thus may train a bit faster or need less data to generalize. Hence, we use GRU, as shown in Figure 2, to model text:

$$\mathbf{z} = \sigma(\mathbf{x}_t\mathbf{U}^z + \mathbf{s}_{t-1}\mathbf{W}^z) \quad (1)$$
$$\mathbf{r} = \sigma(\mathbf{x}_t\mathbf{U}^r + \mathbf{s}_{t-1}\mathbf{W}^r) \quad (2)$$
$$\mathbf{h}_t = \tanh(\mathbf{x}_t\mathbf{U}^h + (\mathbf{s}_{t-1} \circ \mathbf{r})\mathbf{W}^h) \quad (3)$$
$$\mathbf{s}_t = (1 - \mathbf{z}) \circ \mathbf{h}_t + \mathbf{z} \circ \mathbf{s}_{t-1} \quad (4)$$

x is the input sentence with token $\mathbf{x}_t \in \mathbb{R}^d$ at position t, $\mathbf{s}_t \in \mathbb{R}^h$ is the hidden state at t, supposed to

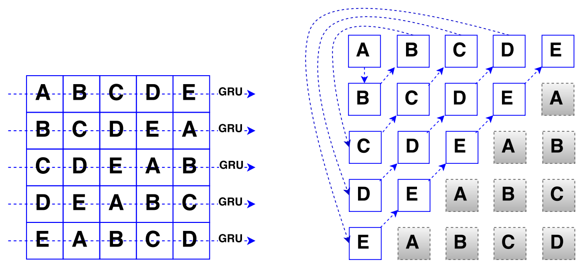


Figure 3: Phrase representation learning by GRU (left), sentence reformatting (right)

encode the history $x_1, \cdots, x_{t-1}$. $\mathbf{z}$ and $\mathbf{r}$ are two gates. All $\mathbf{U} \in \mathbb{R}^{d\times h}$,$\mathbf{W} \in \mathbb{R}^{h\times h}$ are parameters in GRU.

3.2 Representation Learning for Phrases

For a general sentence s with five consecutive words: ABCDE, with each word represented by a word embedding of dimensionality d, we first create four fake sentences, s^1: "BCDEA", s^2: "CDEAB", s^3: "DEABC" and s^4: "EABCD", then put them in a matrix (Figure 3, left).

We run GRUs on each row of this matrix in parallel. As GRU is able to encode the whole sequence up to current position, this step generates representations for any consecutive phrases in original sentence s. For example, the GRU hidden state at position "E" at coordinates (1,5) (i.e., 1st row, 5th column) denotes the representation of the phrase "ABCDE" which in fact is s itself, the hidden state at "E" (2,4) denotes the representation of phrase "BCDE", ... , the hidden state of "E" (5,1) denotes phrase representation of "E" itself. *Hence, for each token, we can learn the representations for all phrases ending with this token.* Finally, all phrases of any lengths in s can get a representation vector. GRUs in those rows are set to share weights so that all phrase representations are comparable in the same space.

Now, we reformat sentence "ABCDE" into $s^* =$ "(A) (B) (AB) (C) (BC) (ABC) (D) (CD) (BCD) (ABCD) (E) (DE) (CDE) (BCDE) (ABCDE)", as shown by arrows in Figure 3 (right), the arrow direction means phrase order. Each sequence in parentheses is a phrase (we use parentheses just for making the phrase boundaries clear). Randomly taking a phrase "CDE" as an example, its representation comes from the hidden state at "E" (3,3) in Figure 3 (left). Shaded parts are discarded. The main advantage of reformatting sentence "ABCDE" into the *new sentence* s^* is to create phrase-level semantic units, but at the same time we maintain the order information.

Hence, the sentence "how big is Auburndale Florida" in Figure 1 will be reformatted into "(how) (big) (how big) (is) (big is) (how big is) (Auburndale) (is Auburndale) (big is Auburndale) (how big is Auburndale) (Florida) (Auburndale Florida) (is Auburndale Florida) (big is Auburndale Florida) (how big is Auburndale Florida)". We can see that phrases are exhaustively detected and represented.

In the experiments of this work, we explore the phrases of maximal length 7 instead of arbitrary lengths.

3.3 Attentive Pooling

As each sentence s^* consists of a sequence of phrases, and each phrase is denoted by a representation vector generated by GRU, we can compute an *alignment matrix* $\mathbf{A}$ between two sentences s^*_1 and s^*_2, by comparing each two phrases, one from s^*_1 and one from s^*_2. Let s^*_1 and s^*_2 also denote lengths respectively, thus $\mathbf{A} \in \mathbb{R}^{s^*_1 \times s^*_2}$. While there are many ways of computing the entries of $\mathbf{A}$, we found that cosine works well in our setting.

The first step then is to detect the best alignment for each phrase by leveraging $\mathbf{A}$. To be concrete, for sentence s^*_1, we do row-wise max-pooling over $\mathbf{A}$ as attention vector $\mathbf{a_1}$:

$$\mathbf{a}_{1,i} = \max(\mathbf{A}[\mathrm{i}, :]) \tag{5}$$

In $\mathbf{a_1}$, the entry $\mathbf{a}_{1,i}$ denotes the best alignment for i^{th} phrase in sentence s^*_1. Similarly, we can do column-wise max-pooling to generate attention vector $\mathbf{a}_2$ for sentence s^*_2.

Now, the problem is that we need to pay most attention to the phrases aligned very well or phrases aligned badly. According to the analysis of the two examples in Figure 1, we need to pay more attention to weaker (resp. stronger) alignments in TE (resp. AS). To this end, we adopt different second step over attention vector $\mathbf{a}_i$ ($i = 1, 2$) for TE and AS.

For TE, in which weaker alignments are supposed to contribute more, we do *k-min-pooling* over $\mathbf{a}_i$, i.e., we only keep the k phrases which are aligned worst. For the (Q, C^+) pair in TE example of Figure 1, we expect this step is able to put most of our attention to the phrases "the kids", "the young boys", "near a man with a smile" and "and the man is smiling nearby" as they have rela-

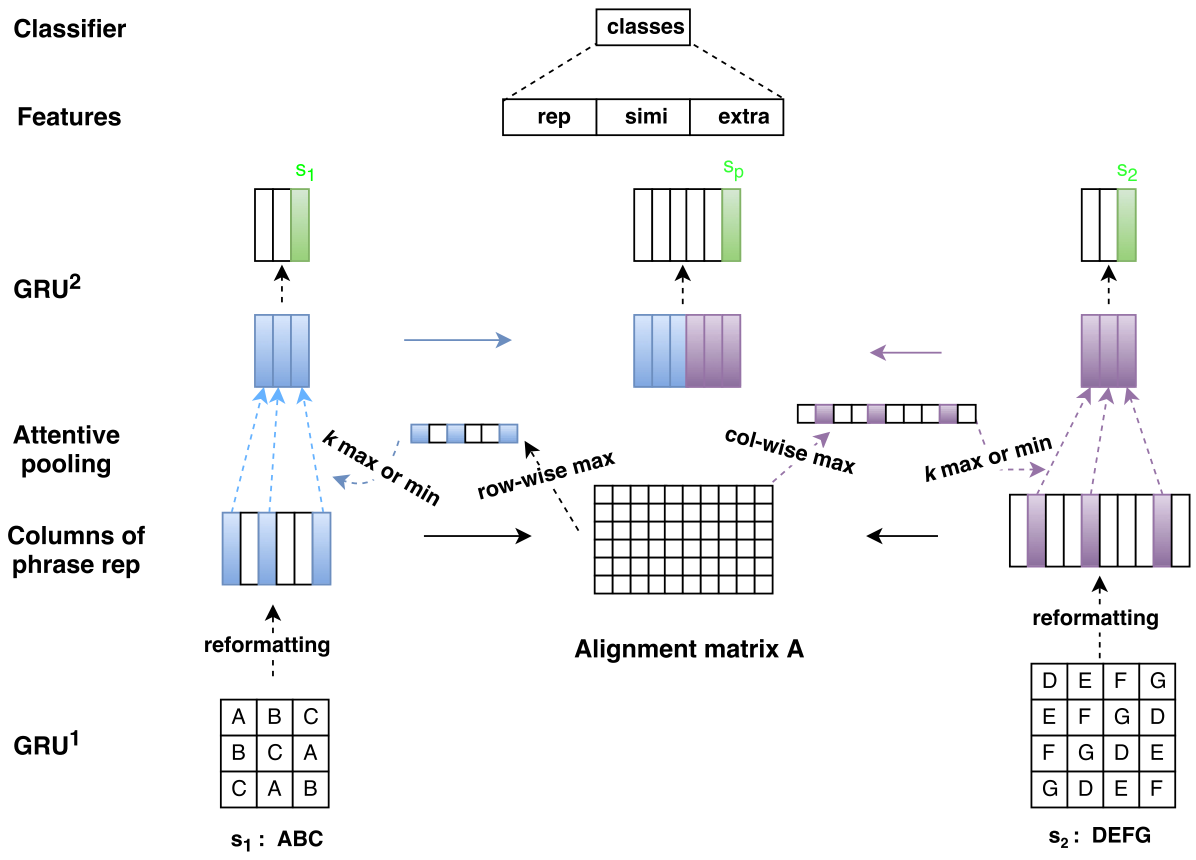


Figure 4: The whole architecture

tively weaker alignments while their relations are the indicator of the final decision.

For AS, in which stronger alignments are supposed to contribute more, we do *k-max-pooling* over $\mathbf{a}_i$, i.e., we only keep the k phrases which are aligned best. For the (Q, C^+) pair in AS example of Figure 1, we expect this k-max-pooling is able to put most of our attention to the phrases "how big" "Auburndale Florida", "the city" and "had a population of 12,381" as they have relatively stronger alignments and their relations are the indicator of the final decision. We keep the original order of extracted phrases after k-min/max-pooling.

In summary, for TE, we first do row-wise max-pooling over alignment matrix, then do k-min-pooling over generated alignment vector; we use *k-min-max-pooling* to denote the whole process. In contrast, we use *k-max-max-pooling* for AS. We refer to this method of using two successive min or max pooling steps as *attentive pooling*.

3.4 The Whole Architecture

Now, we present the whole system in Figure 4. We take sentences s_1 "ABC" and s_2 "DEFG" as illustration. Each token, i.e., A to F, in the figure is denoted by an embedding vector, hence each sentence is represented as an order-3 tensor as input (they are depicted as rectangles just for simplicity). Based on tensor-style sentence input, we have described the phrase representation learning by GRU^1 in Section 3.2 and attentive pooling in Section 3.3.

Attentive pooling generates a new feature map for each sentence, as shown in Figure 4 (the third layer from the bottom), and each column representation in the feature map denotes a key phrase in this sentence that, based on our modeling assumptions, should be a good basis for the correct final decision. For instance, we expect such a feature map to contain representations of "the young boys", "outdoors" and "and the man is smiling nearby" for the sentence Q in the TE example of Figure 1.

Now, we do another GRU^2 step for: 1) the new

	d	lr	bs	L_2	div	k
TE	[256,256]	.0001	1	.0006	.06	5
AS	[50,50]	.0001	1	.0006	.06	6

Table 1: Hyperparameters. d: dimensionality of hidden states in GRU layers; lr: learning rate; bs: mini-batch size; L_2: L_2 normalization; div: diversity regularizer; k: k-min/max-pooling.

feature map of each sentence mentioned above, to encode all the key phrases as the sentence representation; 2) a concatenated feature map of the two new sentence feature maps, to encode all the key phrases in the two sentences sequentially as the representation of the *sentence pair*. As GRU generates a hidden state at each position, we always choose the last hidden state as the representation of the sentence or sentence pair. In Figure 4 (the fourth layer), these final GRU-generated representations for sentence s_1, s_2 and the sentence pair are depicted as green columns: s_1, s_2 and s_p respectively.

As for the input of the final classifier, it can be flexible, such as representation vectors (*rep*), similarity scores between s_1 and s_2 (*simi*), and extra linguistic features (*extra*). This can vary based on the specific tasks. We give details in Section 4.

4 Experiments

We test the proposed architectures on TE and AS benchmark datasets.

4.1 Common Setup

For both TE and AS, words are initialized by 300-dimensional GloVe embeddings[1] (Pennington et al., 2014) and not changed during training. A single randomly initialized embedding is created for all unknown words by uniform sampling from $[-.01, .01]$. We use ADAM (Kingma and Ba, 2015), with a first momentum coefficient of 0.9 and a second momentum coefficient of 0.999,[2] L_2 regularization and Diversity Regularization (Xie et al., 2015). Table 1 shows the values of the hyperparameters, tuned on dev.

Classifier. Following Yin et al. (2016a), we use three classifiers – logistic regression in DNN, logistic regression and linear SVM with default parameters[3] directly on the feature vector – and report performance of the best.

[1]`nlp.stanford.edu/projects/glove/`

[2]Standard configuration recommended by Kingma and Ba

[3]`http://scikit-learn.org/stable/` for both.

Common Baselines. (i) **Addition**. We sum up word embeddings element-wise to form sentence representation, then concatenate two sentence representation vectors (s_1^0, s_2^0) as classifier input. (ii) **A-LSTM**. The pioneering attention based LSTM system for a specific sentence pair classification task "natural language inference" (Rocktäschel et al., 2016). A-LSTM has the same dimensionality as our GRU system in terms of initialized word representations and the hidden states. (iii) **ABCNN** (Yin et al., 2016a). The state-of-the-art system in both TE and AS.

Based on the motivation in Section 1, the main hypothesis to be tested in experiments is: k-min-max-pooling is superior for TE and k-max-max-pooling is superior for AS. In addition, we would like to determine whether the second pooling step in attention pooling, i.e., the k-min/max-pooling, is more effective than a "full-pooling" in which *all* the generated phrases are forwarded into the next layer.

4.2 Textual Entailment

SemEval 2014 Task 1 (Marelli et al., 2014a) evaluates system predictions of textual entailment (TE) relations on sentence pairs from the SICK dataset (Marelli et al., 2014b). The three classes are entailment, contradiction and neutral. The sizes of SICK train, dev and test sets are 4439, 495 and 4906 pairs, respectively. *We choose SICK benchmark dataset so that our result is directly comparable with that of (Yin et al., 2016a), in which non-overlapping text are utilized explicitly to boost the performance. That trick inspires this work.*

Following Lai and Hockenmaier (2014), we train our final system (after fixing of hyperparameters) on train and dev (4,934 pairs). Our evaluation measure is accuracy.

4.2.1 Feature Vector

The final feature vector as input of classifier contains three parts: *rep, simi, extra.*

Rep. Totally five vectors, three are the top sentence representation s_1, s_2 and the top sentence pair representation s_p (shown in green in Figure 4), two are s_1^0, s_2^0 from *Addition* baseline.

Simi. Four similarity scores, cosine similarity and euclidean distance between s_1 and s_2, cosine similarity and euclidean distance between s_1^0 and s_2^0. Euclidean distance $\|\cdot\|$ is transformed into $1/(1+\|\cdot\|)$.

	method	acc
SemEval Top3	(Jimenez et al., 2014)	83.1
	(Zhao et al., 2014)	83.6
	(Lai and Hockenmaier, 2014)	84.6
TrRNTN	(Bowman et al., 2015b)	76.9
Addition	no features	73.1
	plus features	79.4
A-LSTM	no features	78.0
	plus features	81.7
ABCNN	(Yin et al., 2016a)	86.2
GRU k-min-max ablation	– rep	86.4
	– simi	85.1
	– extra	85.5
GRU	k-max-max-pooling	84.9
	full-pooling	85.2
	k-min-max-pooling	**87.1***

Table 2: Results on SICK. Significant improvement over both k-max-max-pooling and full-pooling is marked with $*$ (test of equal proportions, p $<$ .05).

Extra. We include the same 22 linguistic features as Yin et al. (2016a). They cover 15 machine translation metrics between the two sentences; whether or not the two sentences contain negation tokens like "no", "not" etc; whether or not they contain synonyms, hypernyms or antonyms; two sentence lengths. See Yin et al. (2016a) for details.

4.2.2 Results

Table 2 shows that GRU with k-min-max-pooling gets state-of-the-art performance on SICK and significantly outperforms k-max-max-pooling and full-pooling. Full-pooling has more phrase input than the combination of k-max-max-pooling and k-min-max-pooling, this might bring two problems: (i) noisy alignments increase; (ii) sentence pair representation s_p is no longer discriminative – s_p does not know its semantics comes from phrases of s_1 or s_2: as different sentences have different lengths, the boundary location separating two sentences varies across pairs. However, this is crucial to determine whether s_1 entails s_2.

ABCNN (Yin et al., 2016a) is based on assumptions similar to k-max-max-pooling: words/phrases with higher matching values should contribute more in this task. However, ABCNN gets the optimal performance by combining a reformatted SICK version in which overlapping tokens in two sentences are removed. This instead hints that non-overlapping units can do a big favor for this task, which is indeed the superiority of our "k-min-max-pooling".

	method	MAP	MRR
Baselines	CNN-Cnt	0.6520	0.6652
	Addition	0.5021	0.5069
	Addition-Cnt	0.5888	0.5929
	A-LSTM	0.5321	0.5469
	A-LSTM-Cnt	0.6388	0.6529
	AP-CNN	0.6886	0.6957
	ABCNN	0.6921	0.7127
GRU k-max-max ablation	– rep	0.6913	0.6994
	– simi	0.6764	0.6875
	– extra	0.6802	0.6899
GRU	k-min-max-pooling	0.6674	0.6791
	full-pooling	0.6693	0.6785
	k-max-max-pooling	**0.7124***	**0.7237***

Table 3: Results on WikiQA. Significant improvement over both k-min-max-pooling and full-pooling is marked with $*$ (t-test, p $<$ .05). STOA: 74.17 (MAP)/75.88 (MRR) in (Tymoshenko et al., 2016)

4.3 Answer Selection

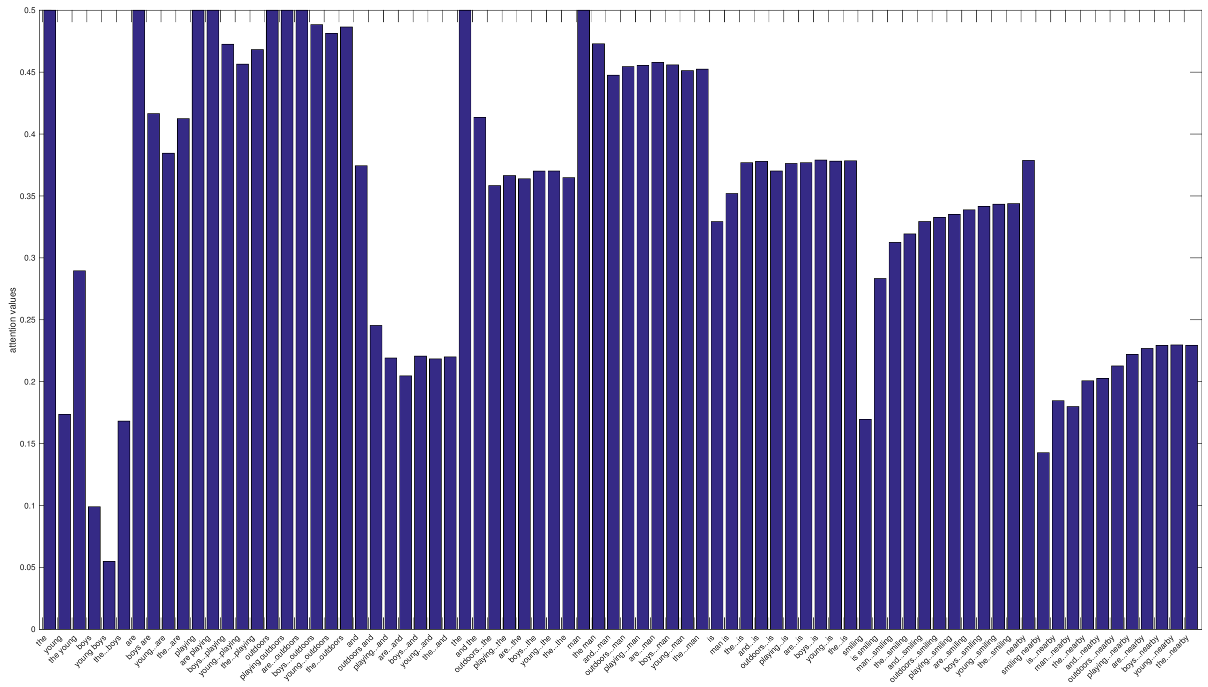
We use WikiQA[4] subtask that assumes there is at least one correct answer for a question. This dataset consists of 20,360, 1130 and 2352 question-candidate pairs in train, dev and test, respectively. Following Yang et al. (2015), we truncate answers to 40 tokens and report mean average precision (MAP) and mean reciprocal rank (MRR).

Apart from the common baselines Addition, A-LSTM and ABCNN, we compare further with: (i) **CNN-Cnt** (Yang et al., 2015): combine CNN with two linguistic features "WordCnt" (the number of non-stopwords in the question that also occur in the answer) and "WgtWordCnt" (reweight the counts by the IDF values of the question words); (ii) **AP-CNN** (Santos et al., 2016).

4.3.1 Feature Vector

The final feature vector in AS has the same (*rep, simi, extra*) structure as TE, except that **simi** consists of only two cosine similarity scores, and **extra** consists of four entries: two sentence lengths, WordCnt and WgtWordCnt.

[4]`http://aka.ms/WikiQA` (Yang et al., 2015)


(a) Attention distribution for phrases in "Q" of TE example in Figure 1

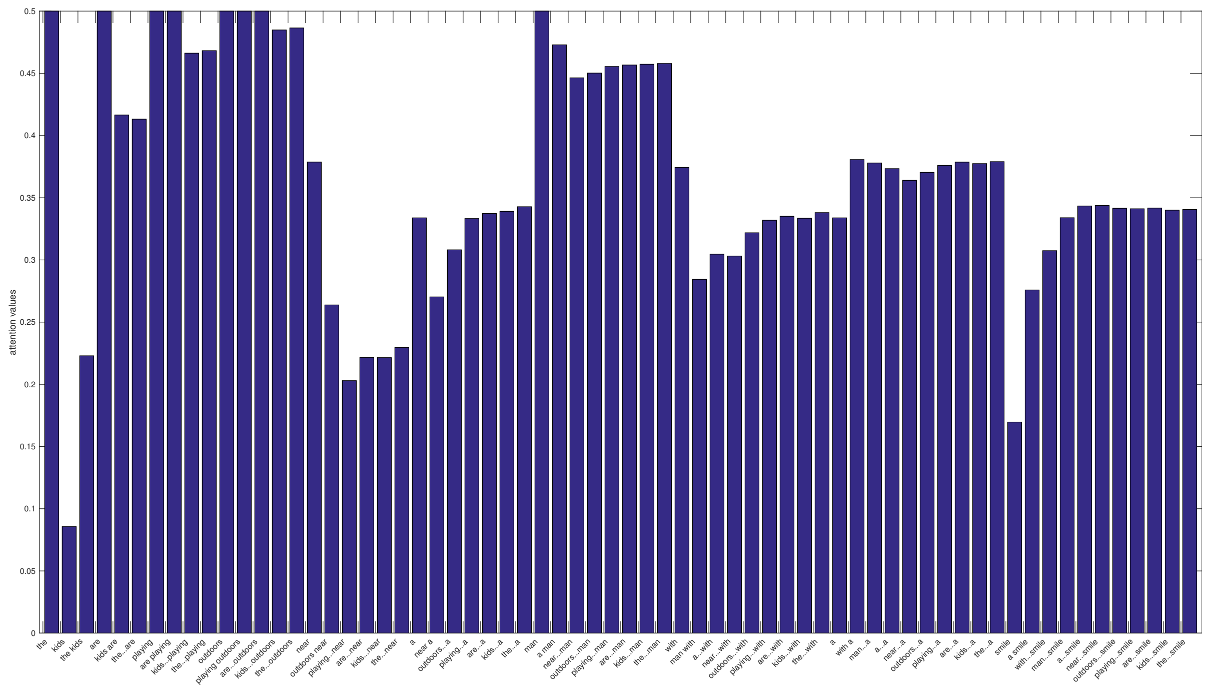

(b) Attention distribution for phrases in "C^+" of TE example in Figure 1

Figure 5: Attention Visualization

4.3.2 Results

Table 3 shows that GRU with k-max-max-pooling is significantly better than its k-min-max-pooling and full-pooling versions. GRU with k-max-max-pooling has similar assumption with ABCNN (Yin et al., 2016a) and AP-CNN (Santos et al., 2016): units with higher matching scores are supposed to contribute more in this task. Our improvement can be due to that: i) our linguistic units cover more exhaustive phrases, it enables alignments in a wider range; ii) we have two max-pooling steps in our attention pooling, especially the second one is able to remove some noisily aligned phrases. Both ABCNN and AP-CNN are based on convolutional layers, the phrase detection is constrained by filter sizes. Even though ABCNN tries a second

CNN layer to detect bigger-granular phrases, their phrases in different CNN layers cannot be aligned directly as they are in different spaces. GRU in this work uses the same weights to learn representations of arbitrary-granular phrases, hence, all phrases can share the representations in the same space and can be compared directly.

4.4 Visual Analysis

In this subsection, we visualize the attention distributions over phrases, i.e., $\mathbf{a_i}$ in Equation 5, of example sentences in Figure 1 (for space limit, we only show this for TE example). Figures 5(a)-5(b) respectively show the attention values of each phrase in (Q, C^+) pair in TE example in Figure 1. We can find that k-min-pooling over this distributions can indeed detect some key phrases that are supposed to determine the pair relations. Taking Figure 5(a) as an example, phrases "young boys", phrases ending with "and", phrases "smiling", "is smiling", "nearby" and a couple of phrases ending with "nearby" have lowest attention values. According to our k-min-pooling step, these phrases will be detected as key phrases. Considering further the Figure 5(b), phrases "kids", phrases ending with "near", and a couple of phrases ending with "smile" are detected as key phrases.

If we look at the key phrases in both sentences, we can find that the discovering of those key phrases matches our analysis in Section 1 for TE example: "kids" corresponds to "young boys", "smiling nearby" corresponds to "near...smile".

Another interesting phenomenon is that, taking Figure 5(b) as example, even though "are playing outdoors" can be well aligned as it appears in both sentences, nevertheless the visualization figures show that the attention values of "are playing outdoors and" in Q and "are playing outdoors near" drop dramatically. This hints that our model can get rid of some surface matching, as the key token "and" or "near" makes the semantics of "are playing outdoors and" and "are playing outdoors near" be pretty different with their sub-phrase "are playing outdoors". This is important as "and" or "near" is crucial unit to connect the following key phrases "smiling nearby" in Q or "a smile" in C^+. If we connect those key phrases sequentially as a new fake sentence, as we did in attentive pooling layer of Figure 4, we can see that the fake sentence roughly "reconstructs" the meaning of the original sentence while it is composed of phrase-level semantic units now.

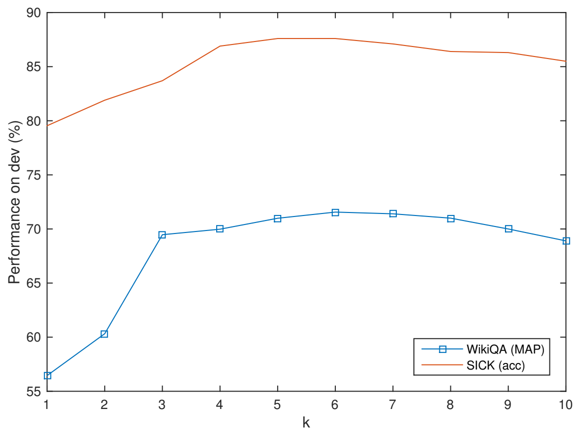


Figure 6: Effects of pooling size k (cf. Table Table 1)

4.5 Effects of Pooling Size k

The key idea of the proposed method is achieved by the k-min/max pooling. We show how the hyperparameter k influences the results by tuning on the dev sets.

In Figure 6, we can see the performance trends of changing k value between 1 and 10 in the two tasks. Roughly $k > 4$ can give competitive results, but larger values bring performance drop.

5 Conclusion

In this work, we investigate the contribution of different intensities of phrase alignments for different tasks. We argue that it is not true that stronger alignments always matter more. We found TE task prefers weaker alignments while AS task prefers stronger alignments. We proposed flexible attentive poolings in GRU system to satisfy the different requirements of different tasks. Experimental results show the soundness of our argument and the effectiveness of our attention pooling based GRU systems.

As future work, we plan to investigate phrase representation learning in context and how to conduct the attentive pooling automatically regardless of the categories of the tasks.

Acknowledgments

We gratefully acknowledge the support of Deutsche Forschungsgemeinschaft for this work (SCHU 2246/8-2).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*, pages 546–556.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2015b. Recursive neural networks can learn logical semantics. In *Proceedings of CVSC workshop*, pages 12–21.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of NAACL-HLT*, pages 429–437.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *Proceedings of IEEE ASRU Workshop*.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of ACL*, pages 864–872.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*, pages 1011–1019.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS*, pages 2042–2050.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. *SemEval*, pages 732–742.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*, pages 2342–2350.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. *SemEval*, pages 329–334.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of ACL*, pages 1106–1115.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, pages 1412–1421.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval*, pages 1–8.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, pages 379–389.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL-HLT*, pages 1268–1278.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of AAAI*, pages 2835–2841.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of NAACL*, pages 1442–1451.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of Coling*, pages 1164–1172.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of SIGMOD*, pages 481–492.

Pengtao Xie, Yuntian Deng, and Eric Xing. 2015. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Semi-markov phrase-based monolingual alignment. In *Proceedings of EMNLP*, pages 590–600.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*, pages 1744–1753.

Wenpeng Yin and Hinrich Schütze. 2015a. Convolutional neural network for paraphrase identification. In *Proceedings of NAACL*, pages 901–911, May–June.

Wenpeng Yin and Hinrich Schütze. 2015b. Multi-grancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of ACL-IJCNLP*, pages 63–73.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016a. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016b. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING*, pages 1746–1756.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS Deep Learning Workshop*.

Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. *SemEval*, pages 271–277.

Chapter 7

Simple Question Answering by Attentive Convolutional Neural Network

Simple Question Answering by Attentive Convolutional Neural Network[*]

Wenpeng Yin[*], **Mo Yu**[†], **Bing Xiang**[†], **Bowen Zhou**[†], **Hinrich Schütze**[*]
[*]Center for Information and Language Processing
LMU Munich, Germany
wenpeng@cis.lmu.de
[†]IBM Watson
Yorktown Heights, NY, USA
{yum,bingxia,zhou}@us.ibm.com

Abstract

This work focuses on answering single-relation factoid questions over Freebase. Each question can acquire the answer from a single fact of form (subject, predicate, object) in Freebase. This task, simple question answering (SimpleQA), can be addressed via a two-step pipeline: entity linking and fact selection. In fact selection, we match the *subject entity in a fact candidate* with the entity mention in the question by a *character-level* convolutional neural network (char-CNN), and match the *predicate in that fact* with the question by a *word-level* CNN (word-CNN). This work makes two main contributions. (i) A simple and effective entity linker over Freebase is proposed. Our entity linker outperforms the state-of-the-art entity linker over SimpleQA task. [1] (ii) A novel attentive maxpooling is stacked over word-CNN, so that the predicate representation can be matched with the predicate-focused question representation more effectively. Experiments show that our system sets new state-of-the-art in this task.

1 Introduction

Factoid question answering (QA) over knowledge bases such as Freebase (Bollacker et al., 2008) has been intensively studied recently (e.g., Bordes et al. (2014), Yao et al. (2014), Bast and Haussmann (2015), Yih et al. (2015), Xu et al. (2016)). Answering a question can require reference to multiple related facts in Freebase or reference to a single fact. This work studies simple question answering (SimpleQA) based on the *SimpleQuestions* benchmark (Bordes et al., 2015) in which answering a question does not require reasoning over multiple facts. Single-relation factual questions are the most common type of question observed in various community QA sites (Fader et al., 2013) and in search query logs. Even though this task is called "simple", it is in reality not simple at all and far from solved.

In SimpleQA, a question, such as "what's the hometown of Obama?", asks a single and direct topic of an entity. In this example, the entity is "Obama" and the topic is hometown. So our task is reduced to finding one fact (subject, predicate, object) in Freebase that answers the question, which roughly means the *subject* and *predicate* are the best matches for the topical entity "Obama" and for the topic description "what's the hometown of", respectively. Thus, we aim to design a method that picks a fact from Freebase, so that this fact matches the question best. This procedure resembles answer selection (Yu et al., 2014) in which a system, given a question, is asked to choose the best answer from a list of candidates. In this work, we formulate the SimpleQA task as a fact selection problem and the key issue lies in the system design for how to match a fact candidate to the question.

The first obstacle is that Freebase has an overwhelming number of facts. A common and effective way is to first conduct entity linking of a question over Freebase, so that only a small subset of facts remain as candidates. Prior work achieves entity linking by searching word n-grams of a question among all entity names (Bordes et al., 2015; Golub and He, 2016). Then, facts whose subject entities match those n-grams are kept. Our first contribution in this work is to present a simple while effective entity linker

[*]This work was conducted during the first author's internship at IBM Watson Group.
[1]*We release our entity linking results at*: https://github.com/Gorov/SimpleQuestions-EntityLinking

to this task. Our entity linker first uses each word of a question (or of an entity mention in the question) to search in the entity vocabulary, all entities are kept if their names *contain one of the query words*. Then, we design three simple factors to give a raw ranking score for each entity candidate: (i) the ratio of words in the entity name that are covered by the question; (ii) the ratio of words in the question that are covered by the entity name; (iii) the position of the entity mention in the question. We choose top-N ranked entities as candidates. Our entity linker does not consider the semantics or topic of an entity; it considers only the string surface. Nevertheless, experiments show that these three factors are the basis for a top-performing entity linker for SimpleQA.

Based on entity linking results, we consider each fact as a fact candidate that has one of the entity candidates as subject. Then our system solves the task of *fact selection*, i.e., matching the question with each fact candidate and picking the best one. Our system is built based on two observations. (i) Surface-form match between a subject entity and its mention in the question provides more straight-forward and effective clue than their semantic match. For example, "Barack Obama" matches with "Obama" in surface-form, which acts as a fundamental indicator that the corresponding fact and the question are possibly about the same "Obama". (ii) Predicate in a fact is a paraphrase of the question's pattern where we define the *pattern* to be the topic asked by the question about the entity, and represent it as the question in which the entity mention has been replaced by a special symbol. Often the predicate corresponds to a keyword or a rephrased token of the pattern, this means we need to create a flexible model to handle this relationship.

These observations motivate us to include two kinds of convolutional neural networks (CNN, LeCun et al. (1998)) in our deep learning system. (i) A character-level CNN (*char-CNN*) that models the match between an Freebase entity and its mention in the question on surface-form. We consider CNN over character-level rather than the commonly-used word-level, so that the generated representation is more robust even in the presence of typos, spaces and other character violations. (ii) A word-level CNN (*word-CNN*) with attentive maxpooling that learns the match of the Freebase predicate with the question's pattern. A Freebase predicate is a predefined relation, mostly consisting of a few words: "place of birth", "nationality", "author editor" etc. In contrast, a pattern is highly variable in length and word choice, i.e., the subsequence of the question that represents the predicate in a question can take many different forms. Convolution-maxpooling slides a window over the input and identifies the best matching subsequence for a task, using a number of filters that support flexible matching. Thus, convolution-maxpooling is an appropriate method for finding the pattern subsequence that best matches the predicate description. *We add attention to this basic operation of convolution-maxpooling.* Attentions are guided by the predicate over all n-gram phrases in the pattern, finally system pools phrase features by considering the feature values as well as the attentions towards those features. *Phrases more similar to the predicate, i.e., with higher attention values, will be selected with higher probability than other phrases to represent the pattern.*[2]

Our overall approach is for the entity linker to identify top-N entity candidates for a question. All facts that contain one of these entities as subject are then the fact search space for this question. Char-CNN and word-CNN decompose each question-fact match into an entity-mention surface-form match and a predicate-pattern semantic match. Our approach has a simple architecture, but it outperforms the state-of-the-art, a system that has a much more complicated structure.

2 Related Work

As mentioned in Section 1, factoid QA against Freebase can be categorized into single-relation QA and multi-relation QA. Much work has been done on multi-relation QA in the past decade, especially after the release of benchmark WebQuestions (Berant et al., 2013). Most state-of-the-art approaches (Berant et al., 2013; Yahya et al., 2013; Yao and Van Durme, 2014; Yih et al., 2015) are based on semantic parsing, where a question is mapped to its formal meaning representation (e.g., logical form) and then translated to a knowledge base (KB) query. The answers to the question can then be retrieved simply

[2]Surface-form entity linking has limitations in candidate collection as some entities have the same names. We tried another word-CNN to match the pattern to the entity description provided by Freebase, but no improvement is observed.

by executing the query. Other approaches retrieve a set of candidate answers from KB using relation extraction (Yao and Van Durme, 2014; Yih et al., 2014; Yao, 2015; Bast and Haussmann, 2015) or distributed representations (Bordes et al., 2014; Dong et al., 2015; Xu et al., 2016). Our method in this work explores CNN to learn distributed representations for Freebase facts and questions.

SimpleQA was first investigated in (Fader et al., 2013) through PARALEX dataset against knowledge base Reverb (Fader et al., 2011). Yih et al. (2014) also investigate PARALEX by a system with some similarity to ours – they employ CNNs to match entity-mention and predicate-pattern. Our model differs in two-fold. (i) They use the same CNN architecture based on a word-hashing technique (Huang et al., 2013) for both entity-mention and predicate-pattern matches. Each word is first preprocessed into a count vector of character-*trigram* vocabulary, then forwarded into the CNN as input. We treat entities and mentions as character sequences. Our char-CNN for entity-mention match is more end-to-end without data preprocessing. (ii) We introduce attentive maxpooling for better predicate-pattern match.

The latest benchmark SimpleQuestions in SimpleQA is introduced by Bordes et al. (2015). Bordes et al. (2015) tackle this problem by an embedding-based QA system developed under the framework of Memory Networks (Weston et al., 2015; Sukhbaatar et al., 2015). The setting of the SimpleQA corresponds to the elementary operation of performing a single lookup in the memory. They investigate the performance of training on the combination of SimpleQuestions, WebQuestions and Reverb training sets. Golub and He (2016) propose a character-level attention-based encoder-decoder framework to encode the question and subsequently decode into (subject, predicate) tuple. Our model in this work is much simpler than these prior systems. Dai et al. (2016) combine a unified conditional probabilistic framework with deep recurrent neural networks and neural embeddings to get state-of-the-art performance.

Treating SimpleQA as fact selection is inspired by work on answer selection (e.g., Yu et al. (2014), Yin et al. (2016b), Santos et al. (2016)) that looks for the correct answer(s) from some candidates for a given question. The answer candidates in those tasks are raw text, not structured information as facts in Freebase are. We are also inspired by work that generates natural language questions given knowledge graph facts (Seyler et al., 2015; Serban et al., 2016). It hints that there exists a kind of match between natural language questions and FB facts.

3 Task Definition and Data Introduction

We first describe the Freebase (Bollacker et al., 2008) and SimpleQuestions task (Berant et al., 2013).

Freebase is a structured knowledge base in which entities are connected by predefined predicates or "relations". All predicates are directional, connecting from the subject to the object. A triple (subject, predicate, object), denoted as (h, p, t), describes a fact; e.g., (U.S. Route 2, major_cities, Kalispell) refers to the fact that U.S. Route 2 runs through the city of Kalispell.

SimpleQuestions benchmark, a typical SimpleQA task, provides a set of single-relation questions; each question is accompanied by a ground truth fact. The object entity in the fact is the answer by default. The dataset is split into train (75,910), dev (10,845) and test (21,687) sets. This benchmark also provides two subsets of Freebase: FB2M (2,150,604 entities, 6,701 predicates, 14,180,937 atomic facts), FB5M (4,904,397 entities, 7,523 predicates, 22,441,880 atomic facts). While single-relation questions are easier to handle than questions with more complex and multiple relations, single-relation question answering is still far from being solved. Even in this restricted domain there are a large number of paraphrases of the same question. Thus, the problem of mapping from a question to a particular predicate and entity in Freebase is hard.

The task assumes that single-relation questions can be answered by querying a knowledge base such as Freebase with a single subject and predicate argument. Hence, only the tuple (h, p) is used to match the question. The evaluation metric is accuracy. Only a fact that matches the ground truth in both subject and predicate is counted as correct.

4 Entity Linking

Given a question, the entity linker provides a set of top-N entity candidates. The input of our deep learning model are (subject, predicate) and (mention, pattern) pairs. Thus, given a question, two problems we have to solve are (i) identifying candidate entities in Freebase that the question refers to and (ii) identifying the span (i.e., mention) in the question that refers to the entity. Each problem can be handled before the other, which results in two entity linkers. (i) **Passive Entity Linker**: First search for entity candidates *by all question words*, then use returned entities to guide the mention detection; (ii) **Active Entity Linker**: First identify the entity mention in the question, then *use the mention span* to search for entity candidates. We now introduce them in detail.

Passive Entity Linker. We perform entity linking by deriving the *longest consecutive common subsequence* (LCCS) between a question and entity candidates and refer to it as σ. Given a question q and all entity names from Freebase, we perform the following three steps.

(i) Lowercase/tokenize entity names and question

(ii) Use each component word of q to retrieve entities whose names contain this word. We refer to the set of all these entities as C_e.

(iii) For each entity candidate e in C_e, compute its LCCS σ with the question q. Let p be the position of the last token of σ in q. Compute $a = |\sigma|/|q|$, $b = |\sigma|/|e|$ and $c = p/|q|$ where $|\cdot|$ is length in words. Finally, entity candidate e is scored by the weighted sum $s_e = \alpha a + \beta b + (1 - \alpha - \beta)c$. Parameters α and β are tuned on dev. Top-N ranked entities are kept for each question.

Discussion. Factor $a = |\sigma|/|q|$ means we prefer candidates that cover *more consecutive words* of the question. Factor $b = |\sigma|/|e|$ means that we prefer the candidates that cover *more consecutive words of the entity*. Factor $c = p/|q|$ means that we prefer candidates that appear *close to the end of the question*; this is based on the observation that most entity mentions are far from the beginning of the question. Despite the simplicity of this passive entity linker, it outperforms other state-of-the-art entity linkers of this SimpleQuestions task by a big margin. Besides, this entity linker is unsupervised and runs fast. We will show its promise and investigate the individual contributions of the three factors in experiments.

Each question q is provided top-N entity candidates from Freebase by entity linker. Then for *mention detection*, we first compute the LCCS σ *on word level* between q and entity e. If the entity is longer than σ and has l (resp. r) words on the left (resp. right) of σ, then we extend σ in the question by l left (resp. r right) words and select this subsequence as the candidate mention. For example, entity "U.S. Route 2" and question "what major cities does us route 2 run through" have LCCS σ "route 2". The FB entity "U.S. Route 2" has one extra word "u.s." on the left of σ, so we extend σ by one left word and the candidate mention is "us route 2". We do this so that the mention has the same word size as the entity string.[3]

In rare cases that the LCCS on the word level has length 0, we treat both entity string and question as character sequence, then compute LCCS σ on character level. Finally, mention is formed by expanding σ on both sides up to a space or the text boundary.

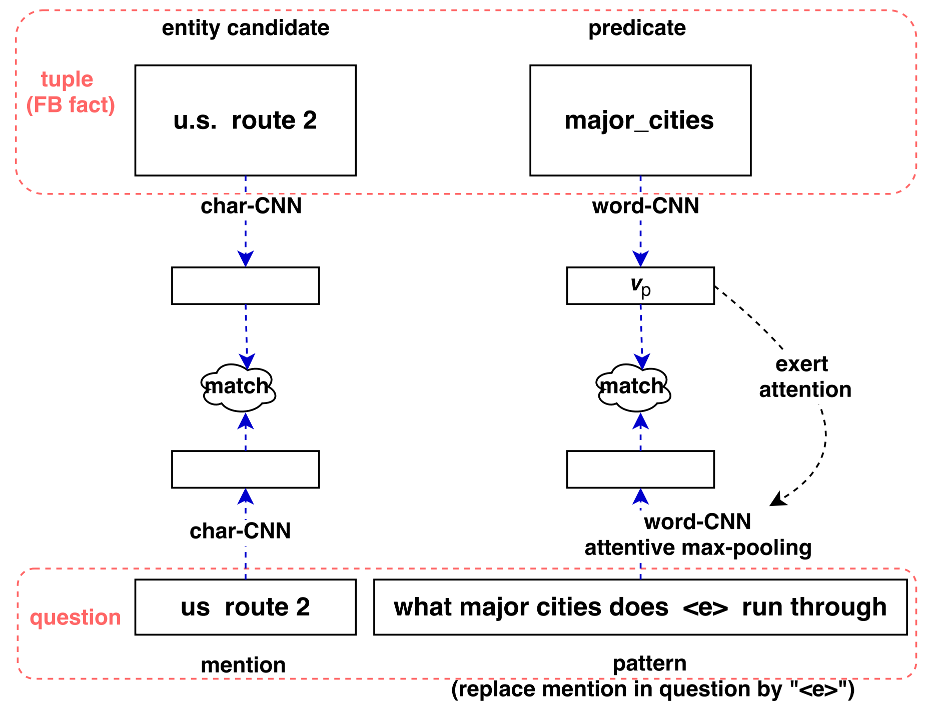
For each question, this approach to mention detection usually produces *more than one (mention, pattern) pair.*

Active Entity Linker. In the training set of SimpleQuestions, the topic entity of each question is labeled. Active entity linker is then achieved by detecting mention in a question by sequential labeling. The key idea is to train a model to predict the text span of the topic entity which can match the gold entity. This is inspired by some prior work. For example, Dai et al. (2016) map the gold entity back to the text to label the text span for each question and then train a BiGRU-CRF model to do the mention detection. Golub and He (2016) propose a generative model which generates the topic entity based on character-level text spans with soft attention scores. Similar to the work (Dai et al., 2016), we trained a BiLSTM-CRF model to detect the entity mentions.

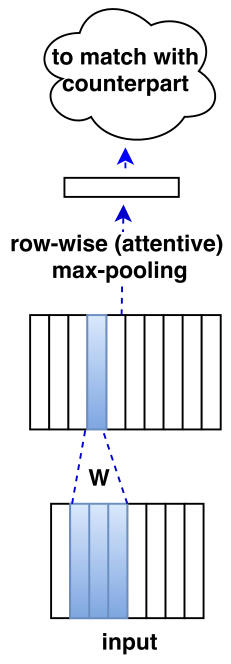
This approach to mention detection produces *only one (mention, pattern) pair* for each question. Then, based on this detected mention, *we use each word of it* to search for the entity candidates via the three steps in "Passive Entity Linker".

We presented two styles of mention detection in questions – passive or active. In passive mention detection, the mention of a question depends on the entity candidates returned by an entity linker. Due

[3]Only using LCCS as mention performed worse.



(a) The whole system. Question: what major cities does us route 2 run through; Tuple: ("u.s. route 2", "major_cities")



(b) Convolution for representation learning

Figure 1: CNN System for SimpleQA

to the different furface-forms of entity candidates, a question can be detected in different spans as mentions. Instead, active mention detection is conducted in a similar way with Name Entity Recognition. Hence, the mention does not depend on the returned entity candidates, a single-relation question has only one mention. Our experiments will show that active entity linker bring better coverage of ground truth entities, nevertheless this method requires the availability of entity-labeled questions as training data.

After mention detection, we then convert the question into the tuple (mention, pattern) where pattern is created by replacing the mention in the question with <e>.

5 Fact Selection

Entity linker provides top-N entity candidates for each question. All facts having those entities as subject form a fact pool, then we build the system to seek the best.

Our whole system is depicted in Figure 1(a). It consists of match from two aspects: (i) a CNN on character level (char-CNN) to detect the similarity of entity string and the mention string in surface-form (the left column); (ii) a CNN with attentive maxpooling (AMP) in word level (word-AMPCNN) to detect if the predicate is a paraphrase of the pattern.

Word-AMPCNN is motivated by the observation that the FB predicate name is short and fixed whereas the corresponding pattern in the question is highly variable in length and word choice. Our hypothesis is that the predicate-pattern match is best done based on keywords in the pattern (and perhaps humans also do something similar) and that the CNN therefore should identify helpful keywords. Traditional maxpooling treats *all n-grams equally*. In this work, we propose *attentive maxpooling* (AMP). AMP gives *higher weights to n-grams that better match the predicate*. As a result, the predicate-pattern match computed by the CNN is more likely to be correct.

Next, we introduce the CNN combined with maxpooling for both char-CNN and word-CNN, then present AMPCNN. Figure 1(b) shows the common framework of char-CNN and word-CNN; only input granularity and maxpooling are different.

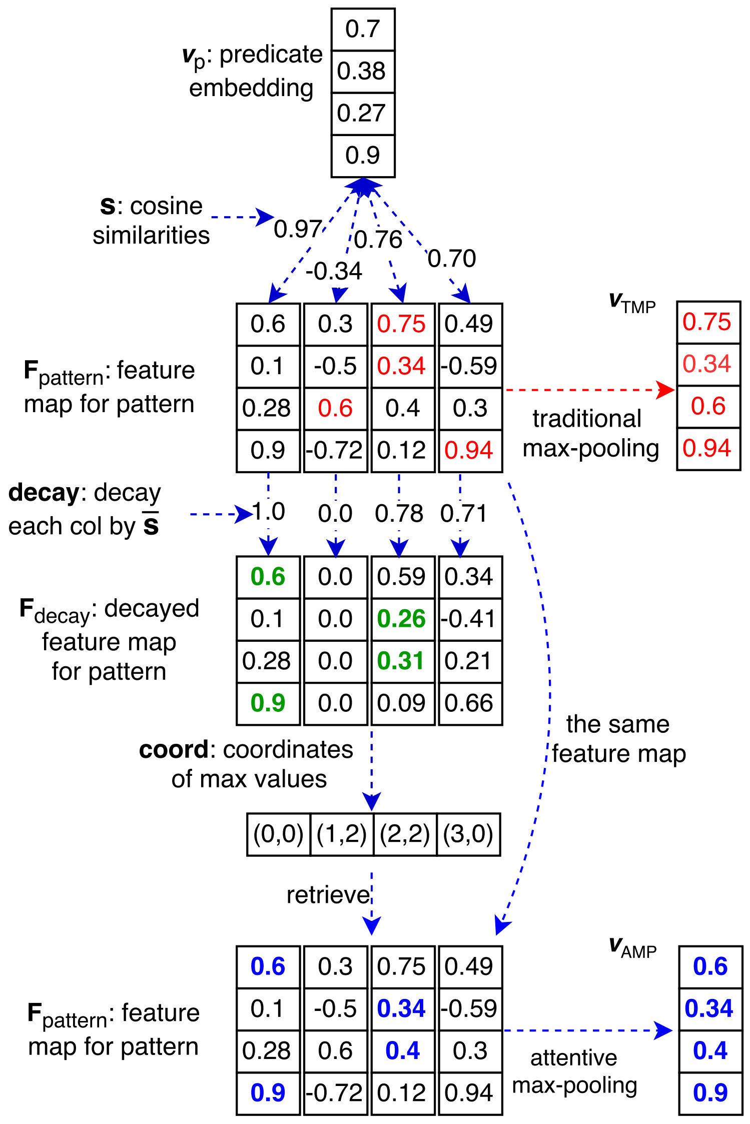


Figure 2: Traditional maxpooling vs. Attentive maxpooling

5.1 Framework of CNN-Maxpooling

Both char-CNN and word-CNN have two weight-sharing CNNs, as they model two pieces of text. In what follows, we use "entry" as a general term for both character and word.

The **input layer** is a sequence of entries of length s where each entry is represented by a d-dimensional randomly initialized embedding; thus the sequence is represented as a feature map of dimensionality $d \times s$. Figure 1(b) shows the input layer as the lower rectangle with multiple columns.

Convolution Layer is used for representation learning from sliding n-grams. For an input sequence with s entries: $v_1, v_2, \ldots, v_s$, let vector $\mathbf{c}_i \in \mathbb{R}^{nd}$ be the concatenated embeddings of n entries $v_{i-n+1}, \ldots, v_i$ where n is the filter width and $0 < i < s + n$. Embeddings for v_i, $i < 1$ or $i > s$, are zero padded. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^d$ for the n-gram $v_{i-n+1}, \ldots, v_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d \times nd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b}) \tag{1}$$

where bias $\mathbf{b} \in \mathbb{R}^d$.

Maxpooling. All n-gram representations $\mathbf{p}_i$ $(i = 1 \cdots s{+}n{-}1)$ are used to generate the representation of input sequence s by maxpooling: $\mathbf{s}_j = \max(\mathbf{p}_{j1}, \mathbf{p}_{j2}, \cdots)$ $(j = 1, \cdots, d)$.

5.2 AMPCNN: CNN-Attentive-Maxpooling

Figure 2 shows TMP (Traditional MaxPooling) and AMP (Attentive MaxPooling) as we apply them to SimpleQA. Recall that we use standard CNNs to produce (i) the predicate representation $\mathbf{v}_p$ (see Figure 1(a)) and (ii) a feature map of the pattern, i.e., a matrix with columns denoting n-gram representations (shown in Figure 1(b), the matrix below "row-wise (attentive) maxpooling"). In Figure 2, we refer to the feature map as $\mathbf{F}_{\text{pattern}}$ and to the predicate representation as $\mathbf{v}_p$.

TMPCNN, i.e., traditional maxpooling, outputs the vector shown as $\mathbf{v}_{\text{TMP}}$; the same $\mathbf{v}_{\text{TMP}}$ is produced for different $\mathbf{v}_p$. The basic idea of AMPCNN is to let the predicate $\mathbf{v}_p$ *bias the selection and weighting of subsequences of the question to compute the representation of the pattern.* The first step in doing that is to compute similarity scores $\mathbf{s}$ between the predicate representation $\mathbf{v}_p$ and each column vector of $\mathbf{F}_{\text{pattern}}$:

$$\mathbf{s}_i = \mathbf{cos}(\mathbf{v}_p, \mathbf{F}_{\text{pattern}}[:, i]) \tag{2}$$

These cosines are then transformed into *decay* values by setting negative values to 0 (negatively correlated column vectors are likely to be unrelated to the predicate) and normalizing the positive values by dividing them by the largest cosine (.97 in this case), so that the largest decay value is 1.0. This is shown as "decay" and $\overline{\mathbf{s}}$ in the figure. Finally, we compute the reweighted feature map $\mathbf{F}_{\text{decay}}$ as follows:

$$\mathbf{F}_{\text{decay}}[:, i] = \mathbf{F}_{\text{pattern}}[:, i] * \overline{\mathbf{s}}_i \tag{3}$$

In $\mathbf{F}_{\text{decay}}$, the matrix with four green values, we can locate the maximal values in each dimension. Notice that they are not the true features by CNN any more, instead, *they convey the original feature values as well as their importance to be considered.* In $\mathbf{F}_{\text{decay}}$, we can see that the maximal values in each dimension mostly come from the first column and the third column which have relatively higher similarity scores 0.97 and 0.76 respectively to the predicate. *We use the coordinates of those maximal values to retrieve features from* $\mathbf{F}_{\textit{pattern}}$ as a final pattern representation $\mathbf{v}_{\text{AMP}}$, the blue column vector[4].

In summary, TMP has no notion of context. The novelty of AMP is that it is guided by attentions from the context, in this case attentions from the predicate. In contrast to TMP, we expect AMP to mainly extract features that come from n-grams that are related to the predicate.

6 Experiments

6.1 Training Setup

Our fact pool consists of all facts whose subject entity is in the top-N entity candidates. For train, we sample 99 negative facts for each ground truth fact; for dev and test, all fact candidates are kept.

Figure 1(a) shows two-way match between a tuple t and a question q: entity-mention match by char-CNN (score m_e), predicate-pattern match by word-AMPCNN (score m_r). The overall ranking score of the pair is $s_t(q, t) = m_e + m_r + s_e$ where s_e is the entity ranking score in entity linking phase.

Our objective is to minimize ranking loss:

$$l(q, t^+, t^-) = \max(0, \lambda + s_t(q, t^-) - s_t(q, t^+)) \tag{4}$$

where λ is a constant.

We build word and character vocabularies on train. OOV words and characters from dev and test are mapped to an OOV index. Then, words (resp. characters) are randomly initialized into d_{word}-dimensional (resp. d_{char}-dimensional) embeddings. The output dimensionality in convolution, i.e., Equation 1, is the same as input dimensionality. We employ Adagrad (Duchi et al., 2011), L_2 regularization and diversity regularization (Xie et al., 2015). Hyperparameters (Table 1) are tuned on dev. For active mention detection, we trained a two-layer BiLSTM followed by a CRF, the hidden layer sizes of both BiLSTM are 200.

[4]We tried max-pooling over $\mathbf{F}_{\text{decay}}$ as $\mathbf{v}_{\text{AMP}}$ directly, but much worse performance was observed.

d_{word}	d_{char}	lr	L_2	div	k	λ
500	100	0.1	.0003	.03	[3,3]	0.5

Table 1: Hyperparameters. $d_{\text{word}}/d_{\text{char}}$: embedding dimensionality; lr: learning rate; L_2: L_2 normalization; div: diversity regularizer; k: filter width in char/word-CNN. λ: see Eq. 4

	baseline		**Ours**				
N	**raw**	**rerank**	**passive-linker**	**-a**	**-b**	**-c**	**active-linker**
1	40.9	52.9	**56.6**	11.0	34.9	52.3	**73.6**
5	–	–	**71.1**	29.5	49.5	67.7	**85.0**
10	64.3	74.0	**75.2**	40.7	56.6	72.8	**87.4**
20	69.3	77.8	**81.0**	63.3	62.4	78.6	**88.8**
50	75.7	82.0	**85.7**	77.1	67.1	84.2	**90.4**
100	79.6	85.4	**87.9**	81.2	70.4	87.0	**91.6**

Table 2: Experimental results for entity linking

6.2 Entity Linking

In Table 2, we compare our (passive and active) entity linkers with the state-of-the-art entity linker (Golub and He, 2016) in this SimpleQA task. Golub and He (2016) report the coverage of ground truth by top-N cases ($N \in \{1, 10, 20, 50, 100\}$). In addition, they explore a reranking algorithm to refine the entity ranking list.

Table 2 first shows the *overall* performance of our passive entity linker and its performance without factor a, b or c (-a, -b, -c). Our passive entity linker outperforms the baseline's *raw* results by big margins and is 2–3 percent above their *reranked* scores. This shows the outstanding performance of our passive entity linker despite its simplicity. The table also shows that all three factors (a, b, c) matter. Observations: (i) Each factor matters more when N is smaller. This makes sense because when N reaches the entity vocabulary size, all methods will have coverage 100%. (ii) The position-related factor c has less influence. From top1 to top100, its contribution decreases from 4.3 to .9. Our linker still outperforms the reranked baseline for $N \geq 20$. (iii) Factor a is dominant for small N, presumably because it chooses the longer one when two candidates exist, which is critical for small N. (iv) Factor b plays a more consistent role across different N.

The last column of Table 2 shows the overall results of our active entity linker, which are significantly better than the results of baseline linker and our passive linker. *We release our entity linking results for follow-up work to make better comparison.*

6.3 SimpleQuestions

Table 3 compares AMPCNN with two baselines. (i) **MemNN** (Bordes et al., 2015), an implementation of memory network for SimpleQuestions task. (ii) **Encoder-Decoder** (Golub and He, 2016), a character-level, attention-based encoder-decoder LSTM (Hochreiter and Schmidhuber, 1997) model. (iii) **CFO** (Dai et al., 2016), the state-of-the-art system in this task with CNN or BiGRU subsystem.

We report results for both passive entity linker and active entity linker. Furthermore, we compare AMPCNN to TMPCNN, i.e., we remove attention and representations for the predicate-pattern match are computed without attention. We choose top-20 (i.e., $N = 20$) entities returned by entity linker. Table 3 shows that AMPCNN with active entity linker has optimal performance for FB2M and FB5M. Performance on FB5M is slightly lower than on FB2M, which should be mainly due to the lower coverage for entity linking on FB5M – about 2% below that on FB2M. In addition, our CNN can still get competitive performance even if the attention mechanism is removed (TMPCNN result). This hints that CNN is promising for SimpleQA.

Settings		Methods	FB2M	FB5M
passive entity linker	baseline	random guess	4.9	4.9
		MemNN	62.7	63.9
		CFO w/ CNN	-	56.0
		CFO w/ BiGRU	-	62.6
	CNN	TMPCNN	**67.5***	**66.6***
		AMPCNN	**68.3***	**67.2***
active entity linker	baseline	Encoder-Decoder	70.9	70.3
		CFO w/ CNN	-	71.1
		CFO w/ BiGRU	-	75.7
	CNN	TMPCNN	75.4	74.6
		AMPCNN	**76.4***	**75.9**

Table 3: Experimental results for SimpleQuestions. Significant improvements over top baseline are marked with * (test of equal proportions, $p < .05$).

	RC	Para
OWA-HABCNN (Yin et al., 2016a)	.847	0
OWA-ABCNN (Yin et al., 2016b)	.902	0
OWA-APCNN (Santos et al., 2016)	.905	$\mathbb{R}^{d\times d}$
AMPCNN	.913	0

Table 4: Comparing different attention schemes of CNN in terms of RC, *extra* parameters brought (Para).

6.4 Effect of Attentive Maxpooling (AMP)

We compare AMP (one main contribution of this work) with three CNN attention mechanisms that are representative of related work in modeling two pieces of text: (i) **HABCNN**: Hierarchical attention-based CNN (Yin et al., 2016a); (ii) **ABCNN**: Attention-based CNN (Yin et al., 2016b); (iii) **APCNN**: CNN with attentive pooling (Santos et al., 2016).

Since attentive matching of predicate-pattern is only one part of our jointly trained system, it is hard to judge whether or not an attentive CNN performs better than alternatives. We therefore create a relation classification (RC) subtask to compare AMP with baseline schemes directly. RC task is created based on SimpleQuestions: label each question (converted into a pattern first) with the ground truth predicate; all other predicates of the gold subject entity are labeled as negative. The resulting datasets have sizes 72,239 (train), 10,310 (dev) and 20,610 (test). It is worth mentioning that this relation classification task is not unspecific to the SimpleQA task, as RC is actually the predict-pattern match part. Hence, this RC subtask can be viewed to check how well the predict-pattern subsystem performs within the whole architecture, and the effectiveness of various attention mechanisms is more clear.

In the three baselines, two pieces of text apply attention to each other. We adapt them into *one-way attention* (OWA) as AMP does in this work: fix predicate representation, and use it to guide the learning of pattern representation. To be specific, ABCNN first gets predicate representation by mean pooling, then uses this representation to derive similarity scores of each n-gram in pattern as attention scores, finally averages all n-gram embeddings weighted by attentions as pattern representation. HABCNN first gets predicate representation by max pooling, then computes attention scores the same way as ABCNN, finally does maxpooling over representations of top-k similar n-grams. APCNN is similar to ABCNN except that the similarity scores are computed by a nonlinear bilinear form.

Table 4 shows that AMPCNN performs well on relation classification, outperforming the best baseline APCNN by 0.8%. AMPCNN also has fewer parameters and runs faster than APCNN.

7 Conclusion

This work explored CNNs for the SimpleQA task. We made two main contributions. (i) A simple and effective entity linker that brings higher coverage of ground truth entities. (ii) An attentive maxpooling stacked above convolution layer that models the relationship between predicate and question pattern more effectively. Our model shows outstanding performance on both simpleQA and relation classification.

Acknowledgments. Wenpeng Yin and Hinrich Schütze were partially supported by DFG (grant SCHU 2246/8-2).

References

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of CIKM*, pages 1431–1440.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of ACL*, pages 800–810.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of ACL-IJCNLP*, volume 1, pages 260–269.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of EMNLP*, pages 1535–1545.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of ACL*, pages 1608–1618.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. In *Proceedings of EMNLP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of CIKM*, pages 2333–2338.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of ACL*, pages 588–598.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. 2015. Generating quiz questions from knowledge graphs. In *Proceedings of WWW*, pages 113–114.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*, pages 2431–2439.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of ICLR*.

Pengtao Xie, Yuntian Deng, and Eric Xing. 2015. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*.

Kun Xu, Yansong Feng, Siva Reddy, Songfang Huang, and Dongyan Zhao. 2016. Enhancing freebase question answering using textual evidence. *arXiv preprint arXiv:1603.00957*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2013. Robust question answering over the web of linked data. In *Proceedings of CIKM*, pages 1107–1116.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of ACL*, pages 956–966.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information extraction or semantic parsing? In *Proceedings of ACL Workshop on Semantic Parsing*, pages 82–86.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of NAACL-HLT*, pages 66–70.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*, pages 643–648.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, pages 1321–1331.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016a. Attention-based convolutional neural network for machine comprehension. In *Proceedings of NAACL Human-Computer QA Workshop*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016b. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *TACL*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *Proceedings of ICLR Workshop*.

Chapter 8

Attention-Based Convolutional Neural Network for Machine Comprehension

Attention-Based Convolutional Neural Network for Machine Comprehension

Wenpeng Yin, Sebastian Ebert, Hinrich Schütze
LMU Munich, Germany
wenpeng, ebert@cis.lmu.de

Abstract

Understanding open-domain text is one of the primary challenges in NLP. Machine comprehension benchmarks evaluate a system's ability to understand text based on the text content only. In this work, we investigate machine comprehension on MCTest, a question answering (QA) benchmark. Prior work is mainly based on feature engineering approaches. We come up with a neural network framework, named hierarchical attention-based convolutional neural network (HABCNN), to address this task without any manually designed features. Specifically, we explore HABCNN for this task by two routes, one is through traditional joint modeling of document, question and answer, one is through textual entailment. HABCNN employs an attention mechanism to detect key phrases, key sentences and key snippets that are relevant to answering the question. Experiments show that HABCNN outperforms prior deep learning approaches by a big margin.

The road to Grandpa's house was long and winding. [...] Jimmy liked to collect insects on the way to his Grandpa's house, so had picked the longer path. As he went along, Jimmy found more and more insects to add to his jar. [...]. Finally, Jimmy arrived at Grandpa's house and knocked. Grandpa answered the door with a smile and welcomed Jimmy inside. They sat by the fire and talked about the insects. They watched the lightning bugs light up as night came.

1: multiple: Why did Grandpa answer the door?
A) Because he saw the insects
B) Because Jimmy was walking
*C) Because Jimmy knocked
D) Because the trip took a long time

2: one: Where do Jimmy and his Grandpa sit?
A) On insects
B) Outside
*C) By the fire
D) On the path

Figure 1: One example with 2 out of 4 questions in the MCTest. "*" marks correct answer.

1 Introduction

Machine comprehension is an open-domain question-answering problem which contains factoid questions, but the answers can be derived by extraction or induction of key clues. Figure 1 shows one example in MCTest (Richardson et al., 2013). Each example consists of one document, four associated questions; each question is followed by four answer candidates of which only one is correct. Questions in MCTest have two categories; "one" questions can be answered based on a single sentence from document where "multiple" questions require several sentences. To correctly answer the first question in the example, the two blue sentences are required; for the second question instead, we only need the red sentence. The following observations hold for the whole MCTest. (i) Most of the sentences in the document are irrelevant for a given question. It hints that we need to pay attention to just some key regions. (ii) Answer candidates vary in length and abstraction level and usually do not appear in the document. For example, candidate B for the second question is "outside", which is one word and does not exist in the document, while the answer candidates for the first question are longer texts with some auxiliary words like "Because" in the text. This requires our system to handle flexible texts via *extraction* as well as *abstraction*. (iii) Some questions require multiple sentences to infer the answer, and those vital sentences mostly appear close to each other (we call them *snippet*). Hence, our system should be able to make a choice or compromise between potential single-sentence clue and snippet clue.

Prior work is mostly based on feature engineering. We take the lead in presenting a deep neural network without linguistic feature engineering.

Concretely, we propose HABCNN, a hierarchical attention-based convolutional neural network, to address this task in two roadmaps. In the first one, we project the document in two different ways, one based on question-attention, one based on answer-attention and then compare the two projected document representations to determine whether the answer matches the question. In the second one, every question-answer pair is reformatted into a statement, then the whole task reduces to textual entailment.

In both roadmaps, convolutional neural network (CNN) is explored to model all types of text. As human beings usually do for such a QA task, our model is expected to be able to detect the key snippets, key sentences, and key words or phrases in the document. In order to detect those informative parts required by questions, we explore an attention mechanism to model the document so that in its representation the required information is emphasized. In practice, instead of imitating human beings in QA task top-down, our system models the document bottom-up, through accumulating the most relevant information from word level to snippet level.

Our approach is novel in three aspects. (i) A document is modeled by a hierarchical CNN for different granularity, from word to sentence level, then from sentence to snippet level. (ii) In the example in Figure 1, apparently not all sentences are required given a question, and usually different snippets are required by different questions. Hence, the same document should have different representations based on what the question is. To this end, attention is incorporated into the hierarchical CNN to guide the learning of dynamic document representations which closely match the information requirements by questions. (iii) Document representations at sentence and snippet levels both are informative for the question. Therefore a highway network is developed to combine them, enabling our system to make a flexible tradeoff.

Overall, we make three contributions. (i) We present a hierarchical attention-based CNN system "HABCNN". It is, to our knowledge, the first deep learning (DL) based system for this MCTest task.

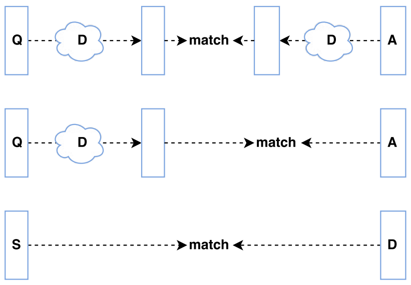


Figure 2: Illustrations of HABCNN-QAP (top), HABCHH-QP (middle) and HABCNN-TE (bottom). Q, A, S: question, answer, statement; D: document

(ii) Prior document modeling systems based on deep neural networks mostly generate generic representation, this work is the first to incorporate attention so that document representation is biased towards the question requirement. (iii) Our HABCNN systems outperform other DL competitors by big margins.

2 Related Work

Existing systems for MCTest are mostly based on manually engineered features, e.g., (Narasimhan and Barzilay, 2015; Sachan et al., 2015; Wang et al., 2015; Smith et al., 2015). In these works, a common route is first to define a loss function based on feature vectors, then the effort focuses on designing effective features based on various rules. Even though this research is groundbreaking for this task, its flexibility and capacity for generalization is limited.

DL approaches appeal to increasing interest in analogous tasks. Weston et al., (2014) introduce memory networks for factoid QA. Memory network framework is extended in (Weston et al., 2015; Kumar et al., 2015) for Facebook bAbI dataset. Peng et al. (2015)'s Neural Reasoner infers over multiple supporting facts to generate an entity answer for a given question and it is also tested on bAbI. All of this work deals with short texts with simple grammar, aiming to *generate* an answer that is restricted to be one word denoting a location, a person etc.

There is also work on other kinds of QA, e.g., (Iyyer et al., 2014; Hermann et al., 2015). Overall, for open-domain MCTest machine comprehension task, we are the first to use deep neural networks.

HABCNN shares similarities with the model published by Trischler et al. (2016) six weeks after our submission on arxiv. It considers multiple levels of granularity in a way that is similar to our approach.

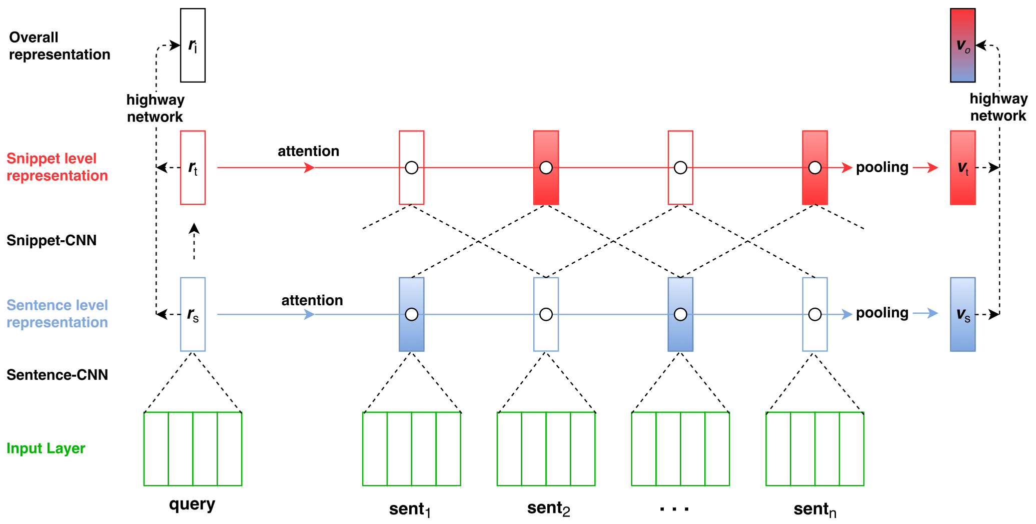


Figure 3: HABCNN. Feature maps for phrase representations $\mathbf{p}_i$ and the max pooling steps that create sentence representations out of phrase representations are omitted for simplification. Each snippet covers three sentences in snippet-CNN. Symbols ○ mean cosine similarity calculation.

Trischler et al. (2016) achieve better performance than HABCNN, but they still use linguistically engineered features like Stanford dependencies whereas our approach is more truly end-to-end.

3 Model

We investigate this task by three approaches, illustrated in Figure 2. (i) We can compute two different document (D) representations in a common space, one based on question (Q) attention, one based on answer (A) attention, and compare them. This architecture is named HABCNN-QAP. (ii) We compute a representation of D based on Q attention (as before), but now we compare it directly with a representation of A. We name this architecture HABCNN-QP. (iii) We treat this QA task as textual entailment (TE), first reformatting Q-A pair into a statement (S), then matching S and D directly. This architecture we name HABCNN-TE. All three approaches are implemented in the common framework HABCNN.

3.1 HABCNN

Recall that we use the abbreviations A (answer), Q (question), S (statement), D (document). HABCNN performs representation learning for triple (Q, A, D) in HABCNN-QP and HABCNN-QAP, for tuple (S, D) in HABCNN-TE. For convenience, we use "query" to refer to Q, A, or S uniformly. HABCNN, depicted in Figure 3, has the following phases.

Input Layer. The input is (query, D). Query is two individual sentences (for Q, A) or one single sentence (for S), D is a sequence of sentences. Words are initialized by d-dimensional pre-trained word embeddings. As a result, each sentence is represented as a feature map with dimensionality of $d \times s$ (s is sentence length). In Figure 3, each sentence in the input layer is depicted by a rectangle with multiple columns.

Sentence-CNN is used for sentence representation learning from word level. Given a sentence of length s with a word sequence: $v_1, v_2, \ldots, v_s$, let vector $\mathbf{c}_i \in \mathbb{R}^{wd}$ be the concatenated embeddings of w words $v_{i-w+1}, \ldots, v_i$ where w is the filter width, d is the dimensionality of word representations and $0 < i < s+w$. Embeddings for words v_i, $i < 1$ and $i > s$, are zero padding. We then generate the representation $\mathbf{p}_i \in \mathbb{R}^{d_1}$ for the *phrase* $v_{i-w+1}, \ldots, v_i$ using the convolution weights $\mathbf{W} \in \mathbb{R}^{d_1 \times wd}$:

$$\mathbf{p}_i = \tanh(\mathbf{W} \cdot \mathbf{c}_i + \mathbf{b}) \qquad (1)$$

where bias $\mathbf{b} \in \mathbb{R}^{d_1}$. $d_1 = s + w - 1$ is the CNN's "kernel size". Sentence-CNNs for query and all document sentences share the same weights, so that the generated representations are comparable.

Sentence-Level Representation. The sentence-CNN generates a new feature map (omitted in Figure 3) for each input sentence, one column in the feature map denotes a phrase representation (i.e., $\mathbf{p}_i$ in Equation (1)).

For the query and each sentence of D, we do *element-wise 1-max-pooling* ("max-pooling" for short) (Collobert and Weston, 2008) over phrase representations to form their representations at this level.

We now treat D as a set of "vital" sentences and "noise" sentences. We propose *attention-pooling* to learn the sentence-level representation of D as follows: first identify vital sentences by computing attention for each of D's sentences as the cosine similarity between its representation and the query representation, then select the k highest-attention sentences to do max-pooling over them. Taking Figure 3 as an example, based on the output of the sentence-CNN layer, $k = 2$ important sentences with blue color are combined by max-pooling as the sentence-level representation $\mathbf{v}_s$ of D; the other – white-color – sentence representations are neglected as they have low attention. (If $k = all$, attention-pooling returns to the common max-pooling in (Collobert and Weston, 2008).) When the query is (Q, A), this step will be repeated, once for Q, once for A, to compute representations of D at the sentence level that are biased with respect to Q and A, respectively.

Snippet-CNN. As the example in Figure 1 shows, to answer the first question "why did Grandpa answer the door?", it does not suffice to compare this question only to the sentence "Grandpa answered the door with a smile and welcomed Jimmy inside"; instead, the snippet "Finally, Jimmy arrived at Grandpa's house and knocked. Grandpa answered the door with a smile and welcomed Jimmy inside" should be used to compare. To this end, it is necessary to stack another CNN layer, *snippet-CNN*, to learn representations of snippets, i.e., units of one or more sentences. Thus, input to snippet-CNN (resp. sentence-CNN) are sentences (resp. words) and the output is representations of snippets (resp. sentences).

Concretely, snippet-CNN puts all sentence representations in column sequence as a feature map and conducts another convolution operation over it. With filter width w, this step generates representation of snippet with w consecutive sentences. Similarly, we use the same CNN to learn higher-abstraction query representations, treating query as a document with one sentence, so that the higher-abstraction query representation is in the same space with corresponding snippet representations.

Snippet-Level Representation. For the output of snippet-CNN, each representation is more abstract and denotes bigger granularity. We apply the same attention-pooling process to snippet level representations: attention values are computed as cosine similarities between query and snippets and the snippets with the k largest attention are retained. Max-pooling over the k selected snippet representations then creates the snippet-level representation $\mathbf{v}_t$ of D. Two selected snippets are shown as red in Figure 3.

Overall Representation. Based on convolution layers at two different granularity, we have derived query-biased representations of D at sentence ($\mathbf{v}_s$) and snippet ($\mathbf{v}_t$) levels. In order to create a flexible choice for open QA, we develop a highway network (Srivastava et al., 2015) to combine the two levels of representations as an overall representation $\mathbf{v}_o$ of D:

$$\mathbf{v}_o = (1 - \mathbf{h}) \odot \mathbf{v}_s + \mathbf{h} \odot \mathbf{v}_t \qquad (2)$$

where highway network weights $\mathbf{h}$ are learned by

$$\mathbf{h} = \sigma(\mathbf{W}_h \mathbf{v}_s + \mathbf{b}) \qquad (3)$$

where $\mathbf{W}_h \in \mathbb{R}^{d_1 \times d_1}$. With the same highway network, we can generate the overall query representation, $\mathbf{r}_i$ in Figure 3, by combining query's representation at sentence level r_s and at snippet level r_t.

3.2 HABCNN-QP & HABCNN-QAP

HABCNN-QP/QAP computes the representation of D as a projection of D, either based on attention from Q or based on attention from A. We hope that these two projections of the document are close for a correct A and less close for an incorrect A. As we said in related work, machine comprehension can be viewed as an answer selection task *using the document D as critical background information*. Here, HABCNN-QP/QAP do not compare Q and A directly, but they use Q and A to filter the document differently, extracting what is critical for the Q/A match by attention-pooling. Then they match the two document representations in the new space.

For simplicity, we have used the symbol $\mathbf{v}_o$ so far, but in HABCNN-QP/QAP we compute two different document representations: $\mathbf{v}_{oq}$, for which attention is computed with respect to Q; and $\mathbf{v}_{oa}$ for which attention is computed with respect to A. $\mathbf{r}_i$ also has two versions, one for Q: $\mathbf{r}_{iq}$, one for A: $\mathbf{r}_{ia}$.

HABCNN-QP and HABCNN-QAP make different use of $\mathbf{v}_{oq}$. HABCNN-QAP projects D twice, once based on attention from Q, once based on attention from A and compares the two projected representations, $\mathbf{v}_{oq}$ and $\mathbf{v}_{oa}$, shown in Figure 2 (top). HABCNN-QP only utilizes the Q-based projection of D and then compares the projected document $\mathbf{v}_{oq}$ with the answer representation $\mathbf{r}_{ia}$, shown in Figure 2 (middle).

3.3 HABCNN-TE

HABCNN-TE treats machine comprehension as *textual entailment*. We use the statements that are provided in MCTest. Each statement corresponds to a question-answer pair; e.g., the Q/A pair "Why did Grandpa answer the door?" / "Because he saw the insects" (Figure 1) is reformatted into the statement "Grandpa answered the door because he saw the insects". The question answering task is then cast as: "does the document entail the statement?"

For HABCNN-TE, shown in Figure 2 (bottom), the input for Figure 3 is the pair (S, D). HABCNN-TE tries to match S's representation $\mathbf{r}_i$ with D's representation $\mathbf{v}_o$.

4 Experiments

4.1 Dataset

MCTest[1] has two subsets. MCTest-160 contains 160 items (70 train, 30 dev, 60 test), each consisting of a document, four questions followed by one correct anwer and three incorrect answers and MCTest-500 500 items (300 train, 50 dev, 150 test).

4.2 Training Setup

Our training objective is to minimize the following ranking loss function:

$$L(d, a^+, a^-) = \max(0, \alpha + S(d, a^-) - S(d, a^+)) \tag{4}$$

where $S(\cdot, \cdot)$ is a matching score between two representation vectors. Cosine similarity is used throughout. α is a constant.

Multitask learning. Based on work showing that question typing is helpful for QA (Sachan et al., 2015), we stack a logistic regression layer over question representation $\mathbf{r}_{iq}$, with the purpose that this subtask can favor the parameter tuning of the whole system, and finally the question is better recognized and answer identification is more accurate.

[1]http://requery.microsoft.com/mct

k	lr	d_1	bs	w	L_2	α
[1,3]	0.05	[90, 90]	1	[2,2]	0.0065	0.2

Table 1: Hyperparameters. k: top-k in attention-pooling for both CNN layers; lr: learning rate; d_1: kernel size in CNN layers; bs: mini-batch size; w: filter width; L_2: L_2 regularization; α: constant in loss function.

To be specific, we classify questions into 12 classes: "how", "how much", "how many", "what", "who", "where", "which", "when", "whose", "why", "will" and "*other*". The question label is created by querying for the label keyword in the question. If more than one keyword appears in a question, we adopt the one appearing earlier and the more specific one (e.g., "how much", not "how"). In case there is no match, the class "*other*" is assigned.

We train with AdaGrad (Duchi et al., 2011) and use 50-dimensional GloVe embeddings (Pennington et al., 2014) to initialize word representations,[2] kept fixed during training. Table 1 gives hyperparameter values, tuned on dev.

We consider two evaluation metrics: *accuracy* (proportion of questions correctly answered) and *NDCG*$_4$ (Järvelin and Kekäläinen, 2002). Unlike accuracy which evaluates if the question is correctly answered or not, NDCG$_4$, being a measure of ranking quality, evaluates the position of the correct answer in our predicted ranking.

4.3 Baseline Systems

(i) **Addition.** Directly compare question and answers without considering the document. Sentence representations are computed by element-wise addition over word representations. (ii) **Addition-proj.** First compute sentence representations for Q, A and all D sentences as in *Addition*. Then identify two sentences from D, taking $\mathbf{x}_q$ and $\mathbf{x}_a$ as example, satisfying that they are most similar to Q and A, respectively. The match score between Q and A is then the cosine similarity of $\mathbf{x}_q$ and $\mathbf{x}_a$. (iii) **NR.** The Neural Reasoner (Peng et al., 2015) has an en-

[2]http://nlp.stanford.edu/projects/glove/

	method	MCTest-150						MCTest-500					
		acc			$NDCG_4$			acc			$NDCG_4$		
		one	mul	all	one	mul	all	one	mul	all	one	mul	all
Baselines	Addition	39.3	32.4	35.7	60.4	50.3	54.6	35.7	30.2	32.9	56.6	55.2	55.8
	Addition-proj	42.1	38.7	40.3	65.3	61.3	63.2	39.4	36.7	38.0	63.3	60.1	61.7
	AR	48.1	44.7	46.3	70.5	68.9	69.6	44.4	39.5	41.9	66.7	64.2	65.4
	NR	48.4	46.8	47.6	70.7	68.2	69.7	45.7	45.6	45.6	71.9	69.5	70.6
	HABCNN-QP	57.9	53.7	55.7	80.4	80.0	80.2	53.7	46.7	50.1	75.4	72.7	74.0
	HABCNN-QAP	59.0	57.9	58.4	81.5	79.9	80.6	54.0	47.2	50.6	75.7	72.6	74.1
	HABCNN-TE	**63.3**	**62.9**	**63.1**	**86.6**	**85.9**	**86.2**	**54.2**	**51.7**	**52.9**	**76.1**	**74.4**	**75.2**
	Sachan et al. (2015)	–	–	–	–	–	–	67.6	67.9	67.8	86.7	86.9	86.8
	Wang et al. (2015)	84.2	67.8	75.2	–	–	–	72.0	67.9	69.9	–	–	–

Table 2: Experimental results for one-sentence (one), multiple-sentence (mul) and all cases.

coding layer, multiple reasoning layers and a final answer layer. The input for the encoding layer is a question and the sentences of the document (called facts); each sentence is encoded by a GRU into a vector. In each reasoning layer, NR lets the question representation interact with each fact representation as reasoning process. Finally, all temporary reasoning clues are pooled as answer representation. (iv) **AR.** The Attentive Reader (Hermann et al., 2015) is implemented by modeling the whole D as a word sequence – without specific sentence / snippet representations – using an LSTM. Attention mechanism is implemented at word representation level.

Baselines Addition(-proj) do not involve complex composition and inference. NR and AR are the top-performing deep neural networks for QA.

4.4 Results

Table 2 lists the performance of baselines, HABCNN systems, and two top-performing non-DL systems (Sachan et al. (2015), Wang et al. (2015)) in the first, second, and last block, respectively (we only report variants for top-performing HABCNN-TE). Consistently, our HABCNN systems outperform all baselines, especially surpass the two competitive deep learning based systems AR and NR. The margin between our best-performing ABHCNN-TE and NR is 15.5/16.5 (accuracy/NDCG) on MCTest-150 and 7.3/4.6 on MCTest-500. This demonstrates the promise of our architecture in this task.

As said before, both AR and NR systems aim to *generate* answers in entity form. Their designs might not suit this machine comprehension task, in which the answers are openly-formed based on summarizing or abstracting the clues. To be more specific, AR models D always at word level, attention is also paid to corresponding word representations, which is applicable for entity-style answers, but is less suitable for comprehension at sentence level or even snippet level. NR contrarily models D in sentence level always, neglecting the discovering of key phrases which however compose most of the answers. In addition, the attention of AR system and the question-fact interaction in NR system both bring large numbers of parameters, this potentially constrains their power in a dataset of limited size.

The size of MCTest is quite small. This is the most likely reason for the inferior performance of all deep learning approaches compared to non-deep-learning approaches. If the amount of training data is limited, then it may not be possible to get top performance without a large feature engineering effort.

5 Conclusion

This work takes the lead in presenting a CNN based neural network system for open-domain machine comprehension task. Our systems try to solve this task in a document projection way as well as a textual entailment way. The latter demonstrates slightly better performance. Overall, our architecture, modeling dynamic document representation by attention scheme from sentence level to snippet level, shows promising results in this task. In the future, more fine-grained representation learning approaches are expected to model complex answer types and question types.

Acknowledgments. We thank DFG for supporting this work (grant SCHU 2246/4-2).

References

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

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *53rd Annual Meeting of the Association for Computational Linguistics*.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. 2015. Towards neural network-based reasoning. *CoRR*, abs/1508.05508.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2.

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answerentailing structures for machine comprehension. In *Proceedings of ACL*.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1693–1698.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2368–2376.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of ACL-IJCNLP*, pages 700–706.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Bibliography

David W. Aha, Dennis F. Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *Proceedings of International Conference on Learning Representations*, 2015.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. EDU-based similarity for paraphrase identification. *Natural Language Processing and Information Systems*, pages 65–76, 2013.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. Exploiting discourse information to identify paraphrases. *Expert Systems with Applications*, 41(6): 2832–2841, 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations*, 2015.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 349–359, 2015.

Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010.

Alberto Barrón-Cedeño, Marta Vila, Maria Antònia Martí, and Paolo Rosso. Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistics*, 39(4):917–947, 2013.

Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 16–23, 2003.

Regina Barzilay, Kathleen McKeown, and Michael Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 550–557, 1999.

Hannah Bast and Elmar Haussmann. More accurate question answering on Freebase. In *Proceedings of ACM International Conference on Information and Knowledge Management*, pages 1431–1440, 2015.

Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. *Large-scale kernel machines*, 34(5):1–41, 2007.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013.

Matthew W. Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. Structured retrieval for question answering. In *Proceedings of Annual International ACM Conference on Research and Development in Information Retrieval*, pages 351–358, 2007.

William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556, 2012.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Bernd Bohnet. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of International Conference on Computational Linguistics*, pages 89–97, 2010.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of International Conference on Management of Data*, pages 1247–1250, 2008.

Danushka Bollegala, Kohei Hayashi, and Ken-ichi Kawarabayashi. Think globally, embed locally - locally linear meta-embedding of words. *CoRR*, abs/1709.06671, 2017.

Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 615–620, 2014a.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014b.

Antoine Bordes, Jason Weston, and Nicolas Usunier. Open question answering with weakly supervised embedding models. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 165–180, 2014c.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 632–642, 2015a.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. Recursive neural networks can learn logical semantics. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 12–21, 2015b.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. Improved statistical machine translation using paraphrases. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 17–24, 2006.

Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, Deva Ramanan, and Thomas S. Huang. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2956–2964, 2015.

Yee Seng Chan and Hwee Tou Ng. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 55–62, 2008.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Vivek Srikumar. Discriminative learning over constrained latent representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 429–437, 2010.

Yung-Chun Chang, Cheng-Wei Shih, and Wen-Lian Hsu. Entailment-based intelligent system for software project monitoring and control. *IEEE Systems Journal*, pages 1–12, 2017.

Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. ABC-CNN: An attention based convolutional neural network for visual question answering. *CoRR*, abs/1511.05960, 2015.

Kyunghyun Cho. *Foundations and Advances in Deep Learning*. PhD thesis, Aalto University, Helsinki, Finland, 2014.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014a.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014b.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition, volume 1, pages 539–546, 2005.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent NN: First results. In *Proceedings of Deep Learning and Representation Learning Workshop*, 2014.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 577–585, 2015.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of International Conference on Machine Learning*, pages 160–167, 2008.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1651–1660, 2016.

Silviu Cucerzan and Eugene Agichtein. Factoid question answering over unstructured and structured web content. In *Proceedings of the Fourteenth Text REtrieval Conference*, volume 72, pages 90–95, 2005.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Zihang Dai, Lei Li, and Wei Xu. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 800–810, 2016.

Dipanjan Das and Noah A. Smith. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 468–476, 2009.

Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, 2013.

Michael Denkowski and Alon Lavie. Extending the meteor machine translation evaluation metric to the phrase level. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–253, 2010.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145, 2002.

Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of International Conference on Computational Linguistics*, pages 350–356, 2004.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over Freebase with multi-column convolutional neural networks. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 260–269, 2015.

Cícero Nogueira dos Santos and Victor Guimarães. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33, 2015.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 626–634, 2015.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016.

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

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1535–1545, 2011.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1608–1618, 2013.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. In *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining*, pages 1156–1165, 2014.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 813–820, 2015.

Andrew Finch, Young-Sook Hwang, and Eiichiro Sumita. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the Third International Workshop on Paraphrasing*, pages 17–24, 2005.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1168–1179, 2011.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. Modeling interestingness with deep neural networks. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2–13, 2014.

Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 2003.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of International Conference on Machine Learning*, pages 1462–1471, 2015.

Weiwei Guo and Mona Diab. Modeling sentences in the latent space. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 864–872, 2012.

Nizar Habash and Ahmed Elkholy. SEPIA: Surface span extension to syntactic dependency precision-based MT evaluation. In *Proceedings of the NIST Metrics for Machine Translation Workshop at the Association for Machine Translation in the Americas Conference*, 2008.

Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

Sanda Harabagiu and Finley Lacatusu. Topic themes for multi-document summarization. In *Proceedings of Annual International ACM Conference on Research and Development in Information Retrieval*, pages 202–209, 2005.

Hua He, Kevin Gimpel, and Jimmy J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1576–1586, 2015.

Xiaodong He and David Golub. Character-level question answering with attention. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1598–1607, 2016.

Michael Heilman and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1011–1019, 2010.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 1684–1692, 2015.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, 2016.

Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Seunghoon Hong, Junhyuk Oh, Honglak Lee, and Bohyung Han. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3204–3212, 2016.

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Ehsan Hosseini-Asl and Angshuman Guha. Similarity-based text recognition by deeply supervised siamese network. *CoRR*, abs/1511.04397, 2015.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2042–2050, 2014.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of ACM International Conference on Information and Knowledge Management*, pages 2333–2338, 2013.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 633–644, 2014.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

Yangfeng Ji and Jacob Eisenstein. Discriminative improvements to distributional sentence similarity. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 891–896, 2013.

Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of International Workshop on Semantic Evaluation*, pages 732–742, 2014.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of International Conference on Machine Learning*, pages 2342–2350, 2015.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 655–665, 2014.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 961–967, 2016.

S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13(3):637–649, 2001.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2741–2749, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 3294–3302, 2015.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

Zornitsa Kozareva and Andrés Montoyo. Paraphrase identification on the basis of supervised machine learning techniques. In *Proceedings of 5th International Conference on Advances in Natural Language Processing*, pages 524–533, 2006.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of International Conference on Machine Learning*, pages 1378–1387, 2016.

Alice Lai and Julia Hockenmaier. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 329–334, 2014.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of International Conference on Machine Learning*, pages 1188–1196, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Hang Li and Jun Xu. Beyond bag-of-words: Machine learning for query-document matching in web search. In *Proceedings of Annual International ACM Conference on Research and Development in Information Retrieval*, pages 1177–1177, 2012.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1106–1115, 2015.

Xutao Li, Michael K. Ng, and Yunming Ye. HAR: Hub, authority and relevance scores in multi-relational data for query search. In *Proceedings of the Twelfth SIAM International Conference on Data Mining*, pages 141–152, 2012.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the ACL Text Summarization Workshop*, pages 74–81, 2004.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1520–1530, 2015.

Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's Thesis, University of Helsinki, 1970.

Chen Liu. *Probabilistic Siamese Network for Learning Representations*. PhD thesis, University of Toronto, 2013.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *Proceedings of International Conference on Machine Learning*, pages 507–516, 2016.

Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 1367–1375, 2013.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

Nitin Madnani, Joel R. Tetreault, and Martin Chodorow. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190, 2012.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of International Workshop on Semantic Evaluation*, pages 1–8, 2014a.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 216–223, 2014b.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. Improved statistical machine translation using monolingually-derived paraphrases. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 381–390, 2009.

Yuval Marton, Ahmed El Kholy, and Nizar Habash. Filtering antonymous, trend-contrasting, and polarity-dissimilar distributional paraphrases for improving statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 237–249, 2011.

Dennis N Mehay and Michael White. Shallow and deep paraphrasing for improved machine translation parameter optimization. In *The AMTA 2012 Workshop on Monolingual Machine Translation, MONOMT*, 2012.

Rada Mihalcea. *Measuring semantic relatedness using salient encyclopedic concepts*. PhD thesis, University of North Texas, 2011.

Tomáš Mikolov. *Statistical language models based on neural networks*. PhD thesis, Brno University of Technology, 2012.

Tomáš Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proceedings of Annual Conference of the International Speech Communication Association*, pages 1045–1048, 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations Workshop*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 3111–3119, 2013b.

George A. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

Margaret Mitchell, Dan Bohus, and Ece Kamar. Crowdsourcing language generation templates for dialogue systems. In *Proceedings of the Eighth International Natural Language Generation Conference*, pages 16–24, 2014.

Andriy Mnih and Geoffrey E. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 641–648, 2007.

Andriy Mnih and Geoffrey E. Hinton. A scalable hierarchical distributed language model. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 1081–1088, 2008.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2265–2273, 2013.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of International Conference on Machine Learning*, pages 1751–1758, 2012.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2204–2212, 2014.

Dan I. Moldovan, Christine Clark, Sanda M. Harabagiu, and Daniel Hodges. Cogex: A semantically and contextually enriched logic prover for question answering. *Journal of Applied Logic*, 5(1):49–69, 2007.

Brent Mombourquette, Christian J. Muise, and Sheila A. McIlraith. Logical filtering and smoothing: State estimation in partially observable domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3613–3621, 2017.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. Tree-based convolution: A new neural architecture for sentence modeling. *CoRR*, abs/1504.01106, 2015.

Avo Muromägi, Kairit Sirts, and Sven Laur. Linear ensembles of word embedding models. *CoRR*, abs/1704.01419, 2017.

Karthik Narasimhan and Regina Barzilay. Machine comprehension with discourse relations. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1253–1262, 2015.

Natalia Neverova, Christian Wolf, Graham W. Taylor, and Florian Nebout. Multiscale deep learning for gesture detection and localization. In *Proceedings of Computer Vision - ECCV Workshops*, pages 474–490, 2014.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

Sebastian Padó. *User's guide to `sigf`: Significance testing by approximate randomisation*, 2006.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(4): 694–707, 2016.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of Annual*

Meeting of the Association for Computational Linguistics, pages 311–318, 2002.

Steven Parker. BADGER: A new machine translation metric. *Metrics for Machine Translation Challenge*, pages 21–25, 2008.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. Towards neural network-based reasoning. In *Proceedings of NIPS Workshop on Reasoning, Attention, Memory*, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pages 185–208, 1999.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. Mapping dependencies trees: An application to question answering. In *Proceedings of AI&Math 2004 (Special session: Intelligent Text Processing)*, 2004.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 18–26, 2006.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Dragomir R. Radev, Hongyan Jing, Magorzata Sty, and Daniel Tam. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6): 919–938, 2004.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 193–203, 2013.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. In *Proceedings of International Conference on Learning Representations*, 2016.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, pages 533–536, 1986.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 379–389, 2015.

Mrinmaya Sachan, Kumar Dubey, Eric P. Xing, and Matthew Richardson. Learning answer-entailing structures for machine comprehension. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 239–249, 2015.

Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep boltzmann machines. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Tobias Schnabel and Hinrich Schütze. FLORS: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26, 2014.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 588–598, 2016.

Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of Annual International ACM Conference on Research and Development in Information Retrieval*, pages 373–382, 2015.

Dominic Seyler, Mohamed Yahya, and Klaus Berberich. Generating quiz questions from knowledge graphs. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 113–114, 2015.

Dan Shen and Mirella Lapata. Using semantic roles to improve question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 12–21, 2007.

Yikang Shen, Wenge Rong, Zhiwei Sun, Yuanxin Ouyang, and Zhang Xiong. Question/answer matching for CQA system via combining lexical and sequential information. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 275–281, 2015.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. A strong lexical matching method for the machine comprehension test. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1693–1698, 2015.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006.

Matthew G. Snover, Nitin Madnani, Bonnie J. Dorr, and Richard M. Schwartz. Ter-plus: Paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127, 2009.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 801–809, 2011.

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

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2377–2385, 2015.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 2440–2448, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 3104–3112, 2014.

Ming Tan, Bing Xiang, and Bowen Zhou. LSTM-based deep learning models for non-factoid answer selection. In *Proceedings of International Conference on Learning Representations Workshop*, 2016.

Noriko Tomuro. Interrogative reformulation patterns and acquisition of question paraphrases. In *Proceedings of the second international workshop on Paraphrasing*, pages 33–40, 2003.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 173–180, 2003.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, and Philip Bachman. A parallel-hierarchical model for machine comprehension on sparse data. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 432–441, 2016.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 384–394, 2010.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1268–1278, 2016.

Zia Ul-Qayyum and Wasif Altaf. Paraphrase identification using semantic heuristic features. *Research Journal of Applied Sciences, Engineering and Technology*, 4(22):4894–4904, 2012.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, 2016.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2835–2841, 2016.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. Using dependency-based features to take the “para-farce” out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop*, volume 2006, pages 131–138, 2006.

Baoxun Wang, Xiaolong Wang, Chengjie Sun, Bingquan Liu, and Lin Sun. Modeling semantic relevance for question-answer pairs in web social communities. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1230–1238, 2010.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. Machine comprehension with syntax, frames, and semantics. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 700–706, 2015.

Mengqiu Wang and Christopher D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of 23rd International Conference on Computational Linguistics*, pages 1164–1172, 2010.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 22–32, 2007.

Shuohang Wang and Jing Jiang. Learning natural language inference with LSTM. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, 2016.

Paul J. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015a.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *Proceedings of International Conference on Learning Representations*, 2015b.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. *1960 IRE WESCON Convention Record*, pages 96–104, 1960.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1504–1515, 2016.

Pengcheng Wu, Steven CH Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of ACM Multimedia Conference*, pages 153–162, 2013.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of International Conference on Management of Data*, pages 481–492, 2012.

Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 842–850, 2015.

Pengtao Xie, Yuntian Deng, and Eric Xing. On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization. *arXiv preprint arXiv:1511.07110*, 2015.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of International Conference on Machine Learning*, pages 2048–2057, 2015.

Kun Xu, Yansong Feng, Siva Reddy, Songfang Huang, and Dongyan Zhao. Enhancing Freebase question answering using textual evidence. *arXiv preprint arXiv:1603.00957*, 2016.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. Robust question answering over the web of linked data. In *Proceedings of ACM International Conference on Information and Knowledge Management*, pages 1107–1116, 2013.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. Joint relational embeddings for knowledge-based question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 645–650, 2014.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2013–2018, 2015.

Xuchen Yao. Lean question answering over Freebase from scratch. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 66–70, 2015.

Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with Freebase. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 956–966, 2014.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Semi-markov phrase-based monolingual alignment. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 590–600, 2013a.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. Automatic coupling of answer extraction and information retrieval. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 159–165, 2013b.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. Freebase QA: Information extraction or semantic parsing? In *Proceedings of Annual Meeting of the Association for Computational Linguistics Workshop on Semantic Parsing*, pages 82–86, 2014.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1744–1753, 2013.

Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 643–648, 2014.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1321–1331, 2015.

Wenpeng Yin and Yulong Pei. Optimizing sentence modeling and selection for document summarization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1383–1389, 2015.

Wenpeng Yin and Hinrich Schütze. An exploration of embeddings for generalized phrases. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 41–47, 2014.

Wenpeng Yin and Hinrich Schütze. Multichannel variable-size convolution for sentence classification. In *Proceedings of the 19th Conference on Computational Natural Language Learning*, pages 204–214, 2015.

Wenpeng Yin and Hinrich Schütze. Learning word meta-embeddings. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1351–1360, 2016.

Wenpeng Yin and Hinrich Schütze. Attentive convolution. *arXiv preprint arXiv:1710.00519*, 2017.

Wenpeng Yin, Tobias Schnabel, and Hinrich Schütze. Online updating of word representations for part-of-speech tagging. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1329–1334, 2015.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep learning for answer sentence selection. *Proceedings of NIPS Deep Learning and Representation Learning Workshop*, 2014.

Yang Yu, Wei Zhang, Kazi Saidul Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end reading comprehension with dynamic answer chunk ranking. *CoRR*, abs/1610.09996, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations*, 2017.

Weinan Zhang, Zhaoyan Ming, Yu Zhang, Ting Liu, and Tat-Seng Chua. Exploring key concept paraphrasing based on pivot language translation for question retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 410–416, 2015a.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 649–657, 2015b.

Ye Zhang, Stephen Roller, and Byron C. Wallace. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527, 2016.

Jiang Zhao, Tiantian Zhu, and Man Lan. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of International Workshop on Semantic Evaluation*, pages 271–277, 2014.

Lei Zou, Ruizhe Huang, Haixun Wang, Jeffer Xu Yu, Wenqiang He, and Dongyan Zhao. Natural language question answering over RDF: A graph data driven approach. In *Proceedings of International Conference on Management of Data*, pages 313–324, 2014.

Curriculum Vitae

Education	
09/2013 – 06/2017	**Ph.D. Student** (CIS, LMU Munich, Germany) Research on Representation Learning
09/2010 – 06/2013	**Master Student** (Peking University, China) Specializations: Computer Science
09/2006 – 07/2010	**Bachelor Student** (Northwestern Polytechnical University, China) Specializations: Computer Science

Practical Experience	
03/2016 - 05/2016	**Intern Scientist** (IBM Watson Research Center, U.S.) Research on question answering via deep learning

Professional Activities	
Program Committee Member	NAACL’2016, ACL’2016, EMNLP’2016, COLING’2016 EACL’2017, IJCAI’2017, ACL’2017, EMNLP’2017 CCL’2017, IJCNLP’2017

Awards	
2014	Baidu Fellowship
2014	Google Travel Grant for ACL