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

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 63–73,
Beijing, China, July 26-31, 2015.

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 non-leaf 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*.

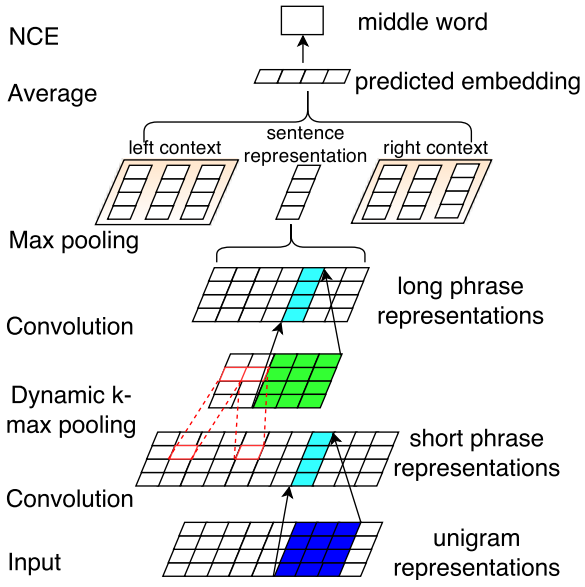

Figure 2: gpCNN: learning g-phrase representations. This figure only shows two convolution layers (i.e., $L = 2$) for saving space.

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.

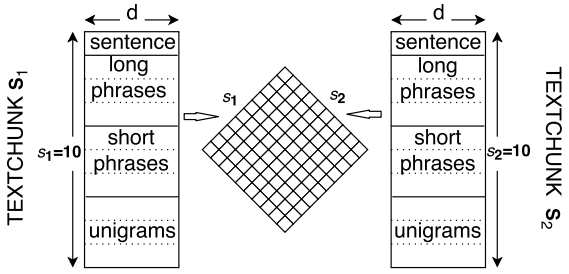

Figure 3: General illustration of match feature model. In this example, both S_1 and S_2 have 10 g-phrases, so the match feature matrix $\hat{\mathbf{F}} \in \mathbb{R}^{s_1 \times s_2}$ has size 10×10.

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.

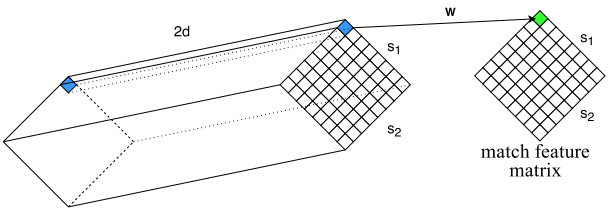

Figure 4: CONCAT match feature model

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

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

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 INDIRECT-SIM. 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:

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

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:

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

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 $\lfloor \frac{s_1}{s^*} \rfloor$ rows while the remaining b splits each have $\lfloor \frac{s_1}{s^*} \rfloor + 1$ 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

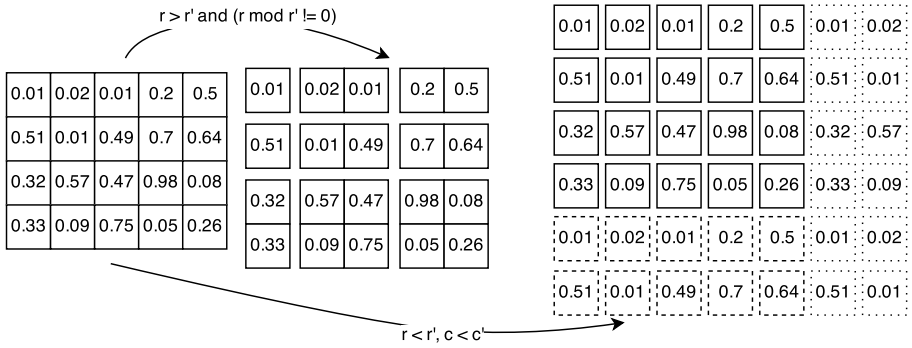

Figure 5: Partition methods in grid-based pooling. Original matrix with size 4×5 is mapped into matrix with size 3×3 and matrix with size 6×7, respectively. Each dynamic pool is distinguished by a border of empty white space around it.

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 $F_r = \{f_1^r, \ldots, f_{s_1 s_2}^r\}$ (resp. $F_c = \{f_1^c, \ldots, f_{s_1 s_2}^c\}$) 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 ($b = 1, 2 \ldots$). 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:

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

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

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 $b + 1$:

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

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

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.

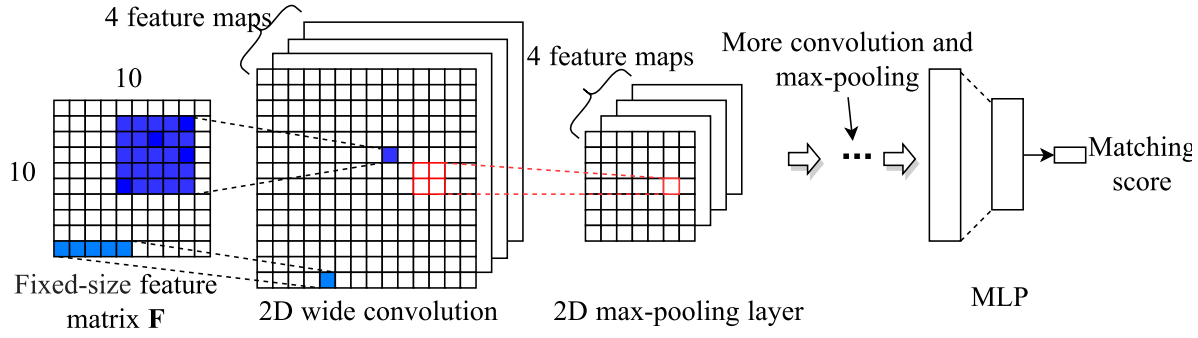

Figure 6: mfCNN & MLP for matching score learning. $s^* = 10$, $f_b = 5$, $k_b = 2$, $c_b = 4$ in this example.

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	1	2	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
	1	93.2	93.5	93.9	93.5
	2	**94.7**	94.2	93.7	93.3
	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*.