Lattice CNNs for Matching Based Chinese Question Answering

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

Short text matching often faces the challenges that there are great word mismatch and expression diversity between the two texts, which would be further aggravated in languages like Chinese where there is no natural space to segment words explicitly. In this paper, we propose a novel lattice based CNN model (LCNs) to utilize multi-granularity information inherent in the word lattice while maintaining strong ability to deal with the introduced noisy information for matching based question answering in Chinese. We conduct extensive experiments on both document based question answering and knowledge based question answering tasks, and experimental results show that the LCNs models can significantly outperform the-state-of-the-art matching models and strong baselines by taking advantages of better ability to distill rich but discriminative information from the word lattice input.

Introduction

Short text matching plays a critical role in many natural language processing tasks, such as question answering, information retrieval, and so on. However, matching text sequences for Chinese or similar languages often suffers from word segmentation, where there are often no perfect Chinese word segmentation tools that suit every scenario. Text matching usually requires to capture the relatedness between two sequences in multiple granularities. For example, in Figure 1, the example phrase is generally tokenized as "中国(China) – 人民(citizen) – 生活(life) – 品质(quality) – 高(high)", but when we plan to match it with "中国人– 活– 得– 好(Chinese live well)", it would be more helpful to have the example segmented into "中国人(Chinese) – 民生(livelihood) – 活(live)" than its common segmentation.

Existing efforts use neural network models to improve the matching based on the fact that distributed representations can generalize discrete word features in traditional bag-of-words methods. And there are also works fusing word level and character level information, which, to some extent, could relief the mismatch between different segmentations, but these solutions still suffer from the original word sequential structures. They usually depend on an existing word tokenization, which has to make segmentation choices at one time e.g., "中国" and "中国人" when processing "中国人民". And the blending just conducts at one position in their frameworks.

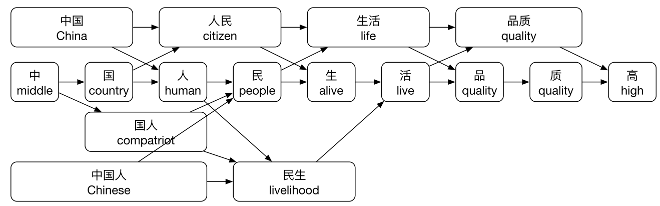

Figure 1: A word lattice for the phrase "Chinese people have high quality of life."

Specific tasks such as question answering (QA) could pose further challenges for short text matching. In document based question answering (DBQA), the matching degree is expected to reflect how likely a sentence can answer a given question, where questions and candidate answer sentences usually come from different sources, and may exhibit significantly different styles or syntactic structures, e.g., queries in web search and sentences in web pages. This could further aggravate the mismatch problems. In knowledge based question answering (KBQA), one of the key tasks is to match relational expressions in questions with knowledge base (KB) predicate phrases[1] , such as "注册地"(place of incorporation). Here the diversity between the two kinds of expressions are even more significant, where there may be dozens of different verbal expressions in natural language questions corresponding to only one KB predicate phrase. Those expression problems make KBQA a further tough task. Previous works (Yih, He, and Meek 2014; Yih et al. 2015) adopt letter-trigrams for the diverse expressions, which is similar to character level in Chinese. And the lattices are combinations of words and characters, so with lattices, we can utilize words information at the same time.

Recent advances have put efforts in modeling multi-granularity information for matching. (Seo et al. 2016; Wang, Hamza, and Florian 2017) blend words and characters to a simple sequence (in word level) , and (Dai et al.

[1]There are usually not enough training data to build a relation extractor for each predicate in a KB, thus the task of KB predicate identification is often formulated as a matching task, which is to select predicates that matches the given questions from candidates.

2018; Chen et al. 2018) utilize multiple convolutational kernel sizes to capture different n-grams. But most characters in Chinese can be seen as words by their own, so combining characters with corresponding words directly may loss the meanings that those characters can express alone. Because of the sequencial inputs, they will either loss word level information when conducting on character sequences or have to make segmentation choices.

In this paper, we propose a multi-granularity method for short text matching in Chinese question answering which utilizes lattice based CNNs to extract sentence level features over word lattice. Specifically, instead of relying on character or word level sequences, LCNs take word lattices as input, where every possible word and character will be treated equally and have their own context, so that they can interact at every layer. For each word in each layer, LCNs can capture different context words in different granularity via pooling methods. To the best of our knowledge, we are the first to introduce word lattice into the text matching tasks. Because of the similar IO structures to original CNNs and the high efficiency, LCNs can be easily adapted to more scenarios where flexible sentence representation modeling is required.

We evaluate our LCNs models on two question answering tasks, document based question answering and knowledge based question answering, both in Chinese. Experimental results show that LCNs significantly outperform the-state-of-the-art matching methods and other competitive CNNs baselines in both scenarios. We also find that LCNs can better capture the multi-granularity information from plain sentences, and, meanwhile, maintain better de-noising capability than vanilla graphic convolutional neural networks thanks to its dynamic convolutional kernels and gated pooling machnism.

Lattice CNNs

Our Lattice CNNs framework is built upon the siamese architecture (Bromley et al. 1993), one of the most successful framework in text matching, which takes the word lattice format of a pair of sentences as input, and output the matching score.

Siamese Architecture

The siamese architecture and its variant have been widely adopted in sentence matching (Mitra, Diaz, and Craswell 2017; Wang, Hamza, and Florian 2017) and matching based question answering (Yu et al. 2014; Yih, He, and Meek 2014; Yu et al. 2017) ,which has a symmetrical component to extract high level features from different input channels, which share parameters and map inputs to the same vector space. Then, the sentence representations are merged and compared to output the similarities.

For our models, we use multi-layer CNNs for sentence representation. Residual connections (He et al. 2016) are used between convolutional layers to enrich features and make it easier to train. Then, max-pooling summarizes the global features to get the sentence level representations, which are merged via element-wise multiplication. The matching score is produced by a multi-layer perceptron (MLP) with 1 hidden layer based on the merged vector. The fusing and matching procedure is formulated as follows:

$$s = \sigma(\boldsymbol{W}_2 \, \texttt{ReLU}(\boldsymbol{W}_1(\boldsymbol{f}_{qu} \odot \boldsymbol{f}_{can}) + \boldsymbol{b}_1^T) + \boldsymbol{b}_2^T) \quad (1)$$

where $\boldsymbol{f}_{qu}$ and $\boldsymbol{f}_{can}$ are feature vectors of question and candidate (sentence or predicate) separately encoded by CNNs, σ is sigmoid function, $\boldsymbol{W_2}, \boldsymbol{W_1}, \boldsymbol{b_1^T}, \boldsymbol{b_2^T}$ are the parameters, and $\odot$ is element-wise multiplication . The training objective is to minimize the binary cross-entropy loss, defined as:

$$L = -\sum_{i=1}^{N} [y_i log(s_i) + (1 - y_i) log(1 - s_i)] \quad (2)$$

Note that the CNNs in the sentence representation component can be either original CNNs with sequence input or lattice based CNNs with lattice input. Intuitively, in an original CNN layer, several kernels scan every n-grams in a sequence and result in one feature vector, which can be seen as the representation for the center word and will be fed into following layers. However, each word may have different context words in different granularities in a lattice and may be treated as center in various kernel spans with same lengths. Therefore, different from the original CNNs, when the model takes a word lattice as input, there could be several feature vectors produced for a given word, which is the key challenge to apply directly the standard CNNs to a lattice input.

For the example shown in Figure 2, the word “人民” is the center word of four text spans with length 3: “中国-人民-生活”, “中国-人民-生”, “国-人民-生活”, “国-人民-生”, so four feature vectors will be produced for width-3 convolutional kernels for “人民”.

Word Lattice

As shown in Figure 1, a word lattice is a directed graph $G = \langle V, E \rangle$, where V represents node set and E represents edge set. For a sentence in Chinese, which is a sequence of Chinese characters $S = c_{1:n} = c_1, c_2, ..., c_n$, all its possible substrings that can be considered as words are treated as vertexes, i.e. $V = \{c_{i:j} | c_{i:j} \texttt{ is word}\}$. Then, all neighbor words are connected by directed edges according to their positions in original sentence, i.e. $E = \{e(c_{i:j}, c_{j:k}) | \forall i, j, k \;\; s.t. \; c_{i:j}, c_{j:k} \in V\}$.

Here, one of the key issues is how we decide a sequence of characters can be considered as a word. We approach this through an existing lookup vocabulary, which contains frequent words appearing in BaiduBaike[2] Note that most Chinese characters can be considered as words by their own, thus are included in this vocabulary when they have been used as words on their own in this corpus.

However, doing so will inevitably introduce noisy words (e.g., “中” in Figure 1) into the word lattices, which will be smoothed by the poolings procedures in our model. And the constructed graph could be disconnected because there may exist a few characters out of the vocabulary, so we append $\langle unk \rangle$ labels to replace those characters to connect the graph.

[2]https://baike.baidu.com

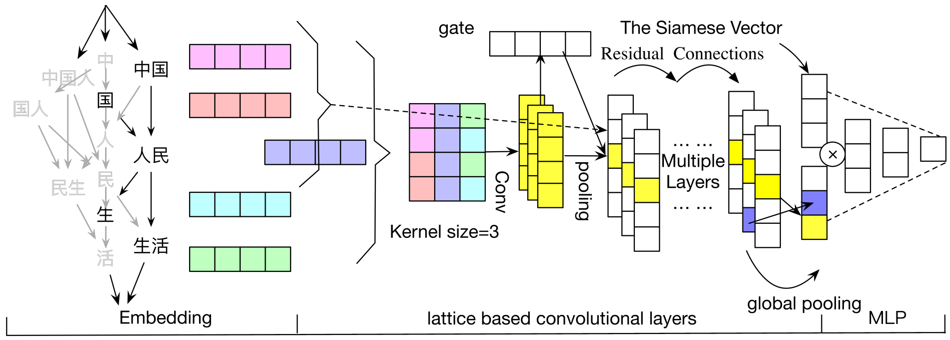

Figure 2: An illustration of our LCN-gated, when “人民” (people) is being considered as the center of convolutional spans.

Obviously, word lattices are collections of characters and all possible words. Therefore, it is not necessary to make explicit decisions regarding specific word segmentations, but just embed all possible information into the lattice and take them to the next CNN layers. The inherent graph structure of a word lattice allows all possible words represented explicitly, no matter the overlapping and nesting cases, and all of them can contribute directly to the sentence representations.

Lattice based CNN Layer

As we mentioned in previous section, we can not directly apply standard CNNs to take word lattice as input, since there could be multiple feature vectors produced for a given word. Inspired by previous lattice LSTM models (Su et al. 2017; Zhang and Yang 2018), here we propose a lattice based CNN layers to allow standard CNNs to work over word lattice input. Specifically, we utilize pooling mechanisms to merge the feature vectors produced by multiple CNN kernels over different context compositions.

Formally, the output feature vector of a lattice CNN layer with kernel size n at word w in a word lattice $G = \langle V, E \rangle$ can be formulated as Eq 3 :

$$F_w = g\{f(\boldsymbol{W}_c(\boldsymbol{v}_{\boldsymbol{w}_1} : ... : \boldsymbol{v}_{\boldsymbol{w}_n}) + \boldsymbol{b}_c^T)| \\ \forall i, w_i \in V, (w_i, w_{i+1}) \in E, w_{\lceil \frac{n+1}{2} \rceil} = w\} \tag{3}$$

where f is the activation function, $\boldsymbol{v}_{\boldsymbol{w}_i}$ is the input vector corresponding to word w_i in this layer, $(\boldsymbol{v}_{\boldsymbol{w}_1} : ... : \boldsymbol{v}_{\boldsymbol{w}_n})$ means the concatenation of those vectors, and $\boldsymbol{W}_c, \boldsymbol{b}_c$ are parameters with size $[m', n \times m]$, and $[m']$, respectively. m is the input dim and m' is the output dim. g is one of the following pooling functions: max-pooling, ave-pooling, or gated-pooling, which execute the element-wise maximum, element-wise average, and the gated operation, respectively. The gated operation can be formulated as:

$$\alpha_1, ..., \alpha_t = \texttt{softmax}\{\boldsymbol{v}_g^T \boldsymbol{v}_1 + b_g, ..., \boldsymbol{v}_g^T \boldsymbol{v}_t + b_g\} \tag{4}$$

$$\texttt{gated-pooling}\{\boldsymbol{v}_1, ..., \boldsymbol{v}_t\} = \sum_{i=1}^{n} \alpha_i \times \boldsymbol{v}_i \tag{5}$$

where $\boldsymbol{v}_g, b_g$ are parameters, and α_i are gated weights normalized by a softmax function. Intuitively, the gates represent the importance of the n-grams contexts, and the weighted sum can control the transmission of noisy context words. We perform padding when necessary.

For example, in Fig 2, when we consider “人民” as the center word, and the kernel size is 3, there will be five words and four context compositions involved, as mentioned in previous section, each marked in different colors. Then, 3 kernels scan on all compositions and produce four 3-dim feature vectors. The gated weights are computed based on those vectors via a dense layer, which can reflect the importance of each context compositions. The output vector of the center word is their weighted sum, where noisy contexts are expected to have lower weights to be smoothed. This pooling over different contexts allow LCNs to work over word lattice input.

Word lattice can be seen as directed graphs and modeled by Directed Graph Convolutional networks (DGCs) (Marcheggiani and Titov 2017; Vashishth et al. 2018), which use poolings on neighboring vertexes that ignore the semantic structure of n-grams. But to some situations, their formulations can be very similar to ours (See **Appendix** for detailed derivation). For example, if we set the kernel size in LCNs to 3, ignore activations functions and suppose the pooling mode is average in both LCNs and DGCs, at each word in each layer, the DGCs computes the agerage of the first order neighbors together with the center word, while the LCNs compute the average of the pre and post words separately and add them to the center word. Empirical results are exhibited in Experiments section.

Finally, given a sentence that has been constructed into a word-lattice form, for each node in the lattice, an LCN layer will produce one feature vector similar to original CNNs, which makes it easier to stack multiple LCN layers to obtain more abstract feature representations.

Experiments

Our experiments are designed to answer: (1) whether multi-granularity information in word lattice helps in matching based QA tasks, (2) whether LCNs capture the multi-granularity information through lattice well, and (3) whether the noisy words introduced by word lattice influence LCNs.

Datasets

We conduct experiments on two Chinese question answering datasets from NLPCC-2016 evaluation task (Duan 2016).

DBQA is a document based question answering dataset. There are 8.8k questions with 182k question-sentence pairs for training and 6k questions with 123k question-sentence pairs in the test set. In average, each question has 20.6 candidate sentences and 1.04 golden answers. The average length for questions is 15.9 characters, and each candidate sentence has averagely 38.4 characters. Both questions and sentences are natural language sentences, possibly sharing more similar word choices and expressions compared to the KBQA case. But the candidate sentences are extracted from web pages, and are often much longer than the questions, with many irrelevant clauses.

KBRE is a knowledge based relation extraction dataset. We follow the same preprocess as (Lai et al. 2017) to clean the dataset[3] and replace entity mentions in questions to a special token. There are 14.3k questions with 273k question-predicate pairs in the training set and 9.4k questions with 156k question-predicate pairs for testing. Each question contains only one golden predicate. Each question averagely has 18.1 candidate predicates and 8.1 characters in length, while a KB predicate is only 3.4 characters long in average. Note that a KB predicate is usually a concise phrase, with quite different word choices compared to the natural language questions, which poses different challenges to solve.

The vocabulary we use to construct word lattices contains 156k words, including 9.1k single character words. In average, each DBQA question contains 22.3 tokens (words or characters) in its lattice, each DBQA candidate sentence has 55.8 tokens, each KBQA question has 10.7 tokens and each KBQA predicate contains 5.1 tokens.

Evaluation Metrics

For both datasets, we follow the evaluation metrics used in the original evaluation tasks (Duan 2016). For DBQA, P@1 (Precision@1), MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank) are adopted. For KBRE, since only one golden candidate is labeled for each question, only P@1 and MRR are used.

Implementation Details

The word embeddings are trained on the Baidu Baike webpages with Google's word2vector[4], which are 300-dim and fine tuned during training. In DBQA, we also followed previous works (Fu, Qiu, and Huang 2016; Xie 2017) to concatenate additional 1d-indicators with word vectors which denote whether the words are concurrent in both questions and answer sentences. In each CNN layer, there are 256, 512, and 256 kernels with width 1, 2, and 3, respectively. The size of hidden layer for MLP is 1024. All activation are ReLU, the dropout rate is 0.5, with batch size of 64. We optimize with adadelta (Zeiler 2012) with learning rate $= 1.0$ and decay factor $= 0.95$. We only tune the number of convolutional layers from [1, 2, 3] and fix other hyper-parameters. We sample at most 10 negative sentences per question in DBQA and 5 in KBRE. We implement our models in Keras[5] with Tensorflow[6] backend.

Baselines

Our first set of baselines use original CNNs with character (**CNN-char**) or word inputs. For each sentence, two Chinese word segmenters are used to obtain three different word sequences: jieba[7] (**CNN-jieba**), and Stanford Chinese word segmenter[8] in CTB (**CNN-CTB**) and PKU (**CNN-PKU**) mode.

Our second set of baselines combines different word segmentations. Specifically, we concatenate the sentence embeddings from different segment results, which gives four different word+word models: **jieba+PKU**, **PKU+CTB**, **CTB+jieba**, and **PKU+CTB+jieba**.

Inspired by previous works (Seo et al. 2016; Wang, Hamza, and Florian 2017), we also concatenate word and character embeddings at the input level. Specially, when the basic sequence is in word level, each word may be constructed by multiple characters through a pooling operation (**Word+Char**). Our pilot experiments show that average-pooling is the best for DBQA while max-pooling after a dense layer is the best for KBQA. When the basic sequence is in character level, we simply concatenate the character embedding with its corresponding word embedding (**Char+Word**), since each character can belong to one word only. Again, when the basic sequence is in character level, we can also concatenate the character embedding with a pooled representation of all words that are connected to this character in the word lattice (**Char+Lattice**), where we use max pooling as suggested by our pilot experiments.

DGCs (Marcheggiani and Titov 2017; Vashishth et al. 2018) are strong baselines that perform CNNs over directed graphs to produce high level representation for each vertex in the graph, which can be used to build a sentence representation via certain pooling operation. We therefore choose to compare with **DGC-max** (with maximum pooling), **DGC-ave** (with average pooling), and **DGC-gated** (with gated pooling), where the gate value is computed using the concatenation of the vertex vector and the center vertex vector through a dense layer.

We also implement several state-of-the-art matching models using the open-source project MatchZoo (Fan et al. 2017), where we tune the hyper-parameters using grid search, e.g., whether using word or character inputs. **Arc1, Arc2, CDSSM** are traditional CNNs based matching models proposed by (Hu et al. 2014; Shen et al. 2014). Arc1 and CDSSM compute the similarity via sentence representations and Arc2 uses the word pair similarities. **MV-LSTM** (Wan et al. 2016) compute the matching score by examining the interaction between the representations from two

[3]About 3% of the questions in the original dataset are removed since they can not link to correct entities/relations due to label errors.

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

[5]https://keras.io

[6]https://www.tensorflow.org

[7]https://pypi.python.org/pypi/jieba/

[8]https://nlp.stanford.edu/software/segmenter.shtml

	DBQA			KBRE	
	MAP	MRR	P@1	P@1	MRR
MatchZoo					
Arc1	.4006	.4011	22.39%	32.18%	.5144
Arc2	.4780	.4785	30.47%	76.07%	.8518
CDSSM	.5344	.5349	36.45%	68.90%	.7974
MP	.7715	.7723	65.61%	86.21%	.9137
MV-LSTM	**.8154**	**.8162**	**71.71%**	**86.87%**	**.9271**
State-of-the-Art DBQA					
(Fu et al. 2016)	.8586	.8592	79.06%	—	—
(Xie 2017)*	**.8763**	**.8768**	—	—	—
CNNs					
CNN-jieba	.8281	.8289	75.10%	86.85%	.9152
CNN-PKU	.8339	.8343	76.00%	89.87%	.9370
CNN-CTB	.8341	.8347	76.04%	88.92%	.9302
CNN-char	**.8803**	**.8809**	**82.09%**	**93.06%**	**.9570**
CNNs Combined Different Word Segmentations					
jieba+PKU	.8486	.8490	77.62%	90.57%	.9417
PKU+CTB	.8435	.8440	77.09%	90.48%	.9410
CTB+jieba	**.8499**	**.8504**	**78.06%**	90.29%	.9399
PKU+CTB+jieba	.8494	.8498	78.04%	**91.16%**	**.9450**
CNNs Combined Words and Characters					
Word+Char	.8566	.8570	78.94%	91.64%	.9489
Char+Word	.8728	.8735	80.76%	92.78%	.9561
Char+Lattice	**.8810**	**.8815**	**81.97%**	**93.12%**	**.9582**
DGCs					
DGC-ave	**.8868**	**.8873**	**83.02%**	**93.49%**	**.9602**
DGC-max	.8811	.8818	82.01%	92.79%	.9553
DGC-gated	.8790	.8795	81.69%	92.88%	.9562
LCNs					
LCN-ave	.8864	.8869	83.14%	**93.60%**	**.9609**
LCN-max	.8870	.8875	83.06%	93.54%	.9604
LCN-gated	**.8895**	**.8902**	**83.24%**	93.32%	.9592

Table 1: The performance of all models on the two datasets. The best results in each group are bolded. * is the best published DBQA result.

sentences obtained by a shared BiLSTM encoder. **MatchPyramid(MP)** (Pang et al. 2016) utilizes 2D convolutions and pooling strategies over word pair similarity matrices to compute the matching scores.

We also compare with state-of-the-art models in DBQA (Fu, Qiu, and Huang 2016; Xie 2017).

Results

Here, we mainly describe the main results on the DBQA dataset, while we find very similar trends on the KBRE dataset. Table 1 summarizes the main results on the two datasets. We can see that the simple MatchZoo models perform the worst. Although Arc1 and CDSSM are also constructed in the siamese architecture with CNN layers, they do not employ multiple kernel sizes and residual connections, and fail to capture the relatedness in a multi-granularity fashion.

(Fu, Qiu, and Huang 2016) is similar to our word level models (CNN-jieba/CNN/CTB), but outperforms our models by around 3%, since it benefits from an extra interaction layer with fine tuned hyper-parameters. (Xie 2017) further incorporates human designed features including POS-tag interaction and TF-IDF scores, achieving state-of-the-art performance in the literature of this DBQA dataset. However, both of them perform worse than our simple **CNN-char** model, which is a strong baseline because characters, that describe the text in a fine granularity, can relief word mismatch problem to some extent. And our best LCNs model further outperforms (Xie 2017) by 0.0134 in MRR.

For single granularity CNNs, **CNN-char** performs better than all word level models, because they heavily suffer from word mismatching given one fixed word segmentation result. And the models that utilize different word segmentations can relief this problem and gain better performance, which can be further improved by the combination of words and characters.

The DGCs and LCNs, being able to work on lattice input, outperform all previous models that have sequential inputs, indicating that the word lattice is a more promising form than a single word sequence, and should be better captured by taking the inherent graph structure into account. Although they take the same input, LCNs still performs better than the best DGCs by a margin, showing the advantages of the CNN kernels over multiple n-grams in the lattice structures and the gated pooling strategy.

To fairly compare with previous KBQA works, we combine our LCN-ave settings with the entity linking results of the state-of-the-art KBQA model(Lai et al. 2017). The P@1 for question answering of single LCN-ave is 86.31%, which outperforms both the best single model (84.55%) and the best ensembled model (85.40%) in literature.

Analysis and Discussions

Effectiveness of Multi-Granularity information As shown in Table 1, the combined word level models (e.g. **CTB+jieba** or **PKU+CTB**) perform better than any word level CNNs with single word segmentation result (e.g. **CNN-CTB** or **CNN-PKU**). The main reason is that there are often no perfect Chinese word segmenters and a single improper segmentation decision may harm the matching performance, since that could further make the word mismatching issue worse, while the combination of different word segmentation results can somehow relief this situation.

Furthermore, the models combining words and characters all perform better than **PKU+CTB+jieba**, because they could be complementary in different granularities. Specifically, **Word+Char** is still worse than CNN-char, because Chinese characters have rich meanings and compressing several characters to a single word vector will inevitably loss information. Furthermore, the combined sequence of **Word+Char** still exploits in a word level, which still suffers from the single segmentation decision. On the other hand, the **Char+Word** model is also slightly worse than CNN-char. We think one reason is that reduplicated word embeddings concatenated with each character vector confuse the CNNs, and perhaps lead to overfitting. But, we can still see that **Char+Word** performs better than **Word+Char**, because the former exploits in a character level and the fine-granularity information actually helps to relief word mismatch. Note that **Char+Lattice** outperforms **Char+Word**, and even slightly better than . This illustrates that multiple word segmentations are still helpful to further improve the character level strong baseline **CNN-char**, which may still benefits from word level information in a multi-granularity fashion.

In conclusion, the combination between different sequences and information of different granularities can help to improve text matching, showing that it is necessary to consider the fashion which considers both characters and more possible words, which perhaps the word lattice can provide.

Poolings in DGCs and LCNs For DGCs with different kinds of pooling operations, average pooling (DGC-ave) performs the best, which delivers similar performance with LCN-ave. While DGC-max performs a little worse, because it ignores the importance of different edges and the maximum operation is more sensitive to noise than the average operation. The DGC-gated performs the worst. Compared with LCN-gated that learns the gate value adaptively from multiple n-grams context, it is harder for DGC to learn the importance of each edge via the node and the center node in the word lattice. It is not surprising that LCN-gated performs much better than GDC-gated, indicating again that n-grams in word lattice play an important role in context modeling, while DGCs are designed for general directed graphs which may not be perfect to work with word lattice.

For LCNs with different pooling operations, LCN-max and LCN-ave lead to similar performances, and perform better on KBRE, while the LCN-gated is better on DBQA. This may be due to the fact that the sentences in DBQA are relatively longer with more irrelevant information which require LCN-gated to filter noisy context, while on KBRE with much shorter predicate phrases, LCN-gated may slightly overfit due to its more complex model structure. Overall, we can see that LCNs performs better than DGCs, thanks to its advantage of better capturing multiple n-grams context in word lattice.

How LCNs utilizes Multi-Granularity To investigate how LCNs utilize multi-granularity more intuitively, we analyze the MRR score against granularities of overlaps between questions and answers in DBQA dataset, which is shown in Figure 3. It is demonstrated that **CNN-char** performs better than **CNN-CTB** impressively in first few groups where most of the overlaps are single characters which will cause serious word mismatch. With growing of the length of overlaps, **CNN-CTB** is catching up and finally overtakes **CNN-char** even though its overall performance is much lower. This results show that word information is complementary to characters to some extent. The LCN-gated is approaching to the **CNN-char** in first few groups, and outperforms both character and word level models in next groups, where word level information become more powerful. This demonstrates that LCNs can effectively take advantages from different granularities, and the combination will not be harmful even when the matching clues presents in extreme cases.

Handling Noisy Words In pervious experiments, we construct the word lattice by an existing lookup vocabulary, which will introduce some noisy words inevitably. Here we construct from various word segmentations with different strategies to investigate whether LCNs can handle the noisy words. Here we only use the DBQA dataset because word lattices here are more complex. And the more complex the word lattices are, the more word information they can have, and meanwhile, the more noisy words they may carry.

From Table 2, it is shown that all kind of lattice are better than CNN-char, which also evidence the usage of word information. And among all LCN models, more complex lattice produces better performances in principle, which indicates that LCNs can handle the noisy words well and the influence of noisy words can not cancelling the positive information brought in complex lattices. It is also noticeable that LCN-Gate is better than LCN-C+20 by a considerable margin, which shown that the words not in general tokenization (e.g. “民生” in Fig 1) are potential useful.

Parameters and Efficiency The LCNs only introduce inappreciable parameters in gated pooling besides the increasing vocabulary, which will not bring a heavy burden. The training speed is about 2.8 batches per second, which is 5 times slower than original CNNs. The whole training of a 2-layer LCN on DBQA dataset only takes about 37.5 minutes[9]. The efficiency may be further improved if the net-

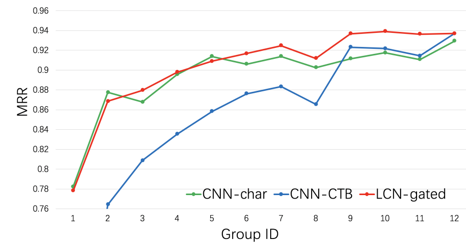

Figure 3: MRR score against granularities of overlaps between questions and answers, which is the average length of longest common substrings. About 2.3% questions are ignored for they have no overlaps and the rests are separated in 12 groups orderly and equally. Group 1 has the least average overlap length while group 12 has the largest.

	MRR	P@1	l.qu	l.sen
CNN-char	.8809	82.09%	15.9	38.4
LCN-C+2&	.8851	82.41%	19.9	48.0
LCN-C+2	.8874	82.89%	20.4	49.5
LCN-C+20	.8869	82.81%	21.4	51.0
LCN-Gate	.8902	83.24%	22.3	55.8

Table 2: Comparisons of various ways to construct word lattice. l.qu and l.sen are the average token number in questions and sentences respectively. The 4 models in the middle construct lattices by adding words to CNN-char. +2& considers the intersection of words of CTB and PKU mode while +2 considers the union. +20 uses the top 10 results of the two segmentors.

[9]Environment: CPU:, 2*XEON E5-2640 v4. GPU, 1*NVIDIA GeForce 1080Ti

work structure builds dynamically with supported frameworks. The fast speed and little parameter increment gives LCNs a promising future in more NLP tasks.

Case Study

Question: 到2013年12月有多少人是由tao606卖家网址导航 到, 2013, 年, 12, 月, 有, 多少, 人, 是, 由, tao, 606, 卖家, 网址, 导航 By Dec. 2013, how many people were navigated by the tao606 seller website
Word: 2013年9月黄诚顺创建tao606卖家网址导航 **2013, 年**, 9, **月**, 黄诚, 顺, 创建, **tao, 606, 卖家, 网址, 导航** In Sep. 2013, Huang Chengshun created tao606 seller website navigation
Character: tao606是国内首家淘宝卖家专用的网址导航，由创始人黄诚顺于2013年9月创办... **tao, 606**, **是**, 国, 内, 首, **家**, 淘, 宝, **卖, 家**, 专, 用, 的, **网, 址, 导, 航**，由, 创, 始, 人, 黄, 诚, 顺, 于, **2013, 年**, 9, **月**, 创, 办... Tao606 is the first website navigation for Taobao sellers in China, which is founded by Huang Chengshun in Sep. 2013...
Lattice: 2013年12月，网站总用户达到近2万 **2013, 年, 12, 月**, **网**,站, 总, 用, 户, 达, 到, 近, 2, 万, **2013年, 12月, 2月**, 网站, 用户, 达到, 2万 In Sep. 2013, the total number of users of the website reached nearly 20,000.

Table 3: Example, questions (in word) and 3 sentences selected by 3 systems. **Bold** mean sequence exactly match between question and answer.

Table 3 shows a case study comparing models in different input levels. The word level model is relatively coarse in utilizing informations, and finds a sentence with the longest overlap (5 words, 12 characters). However, it does not realize the question is about numbers of people, and the "导航"(navigate) in question is a verb, but noun in the sentence. The character level model finds a long sentence which covers most of the characters in question, which shows the power of fine-granularity matching. But without the help of word, it is hard to distinguish the "人"(people) in "多少人"(how many people) and "创始人"(founder). So, it loses the most important information. While in lattice, although overlaps are limited, "网站"(website, "网" web, "站" statation) can match the "网址"(Internet addresses, "网" web, "址" addresses) and also related to "导航"(navigate), from which it may inference the "网站"(website) refers to "tao606卖家网址导航(a website name)". Moreover, "用户"(user) can match the "人"(people). With the cooperation between characters and words, it catches the key points of the question and eliminates the other two candidates, as a result, it finds the correct answer.

Related Work

Deep learning models have been widely adopted in natural language sentence matching. Representation based models (Hu et al. 2014; Shen et al. 2014; Yu et al. 2014; Yih, He, and Meek 2014; Yu et al. 2017) encode matching branches into hidden vectors and compare them for similarity scores. Interaction based models (Hu et al. 2014; Pang et al. 2016; Wan et al. 2016; Wang, Hamza, and Florian 2017) incorporates interactions features between all word pairs and adopt 2D-convolution to extract matching features. Our models are built upon the representation based architecture, which is better for short text matching.

In recent years, many researchers have become interested in utilizing all sorts of external or multi-granularity information in matching tasks. (Yin and Schütze 2015) exploit hidden units in different depths to realize interaction between substrings with different lengths. (Wang, Hamza, and Florian 2017) join multiple pooling methods in merging sentence level features, (Chen et al. 2018; Dai et al. 2018) exploit interactions between different lengths of text spans. For those more similar to this work, (Wang, Hamza, and Florian 2017) also incorporate characters, which is fed into LSTMs and concatenate the outcomes with word embeddings, and (Yu et al. 2017) utilize words together with predicates level tokens in KBRE task. However, none of them exploit the multi-granularity information in word lattice in languages like Chinese that do not have space to segment words naturally. Furthermore, our model has no conflicts with most of them except (Wang, Hamza, and Florian 2017) and could gain further improvement.

GCNs (Bruna et al. 2014; Defferrard, Bresson, and Vandergheynst 2016) are the first to model graph information using convolutional networks, and DGCs generalize GCNs on directed graphs in the fields of semantic-role labeling (Marcheggiani and Titov 2017) and document dating (Vashishth et al. 2018). However, DGCs controls information flowing from neighbor vertexes via edge types, while we focus on capturing different contexts for each word in word lattice via convolutional kernels and poolings. There are previous works that involved Chinese lattice into neural networks, (Su et al. 2017) used lattice GRU on Chinese-English translation to alleviate error propagations of 1-best tokenizations, (Zhang and Yang 2018) used lattice LSTM in Chinese named entity recognition to utilize word information but avoiding segmented entity boundary errors. To the best of our knowledge, we are the first to conduct CNNs on word lattice, and the first to involve word lattice in matching tasks. And we motivate to utilize multi-granularity information in word lattices including characters and different segmented words to relief word mismatch and diverse expressions in Chinese question answering. Moreover, in LCNs, each word and character represented explicitly at each layer, so they can interact with contexts in multi-granularity, which is more intuitive than those lattice RNN models which just combines multiple pre-words at each sentence position.

Conclusions

In this paper, we propose a novel neural network matching method (LCNs) for matching based question answering in Chinese. Rather than relying on a word sequence only, our model takes word lattice as input, and performs CNNs over multiple n-grams context to exploit multi-granularity information to overcome the word mismatch challenges. Thorough experiments show that our model can better explore the word lattice via CNNs operations and rich context-aware pooling, thus outperform the-state-of-the-art models and competitive baselines by a large margin.

References

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1993. Signature verification using a "siamese" time delay neural network. In *NIPS 1993*, 737–744.

Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2014. Spectral networks and locally connected networks on graphs. In *ICLR 2014*.

Chen, H.; Han, F. X.; Niu, D.; Liu, D.; Lai, K.; Wu, C.; and Xu, Y. 2018. Mix: Multi-channel information crossing for text matching. In *SIGKDD 2018*, 110–119. ACM.

Dai, Z.; Xiong, C.; Callan, J.; and Liu, Z. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM 2018*, 126–134.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, 3844–3852.

Duan, N. 2016. Overview of the nlpcc-iccpol 2016 shared task: open domain chinese question answering. In *Natural Language Understanding and Intelligent Applications*. Springer. 942–948.

Fan, Y.; Pang, L.; Hou, J.; Guo, J.; Lan, Y.; and Cheng, X. 2017. Matchzoo: A toolkit for deep text matching. *arXiv preprint arXiv:1707.07270*.

Fu, J.; Qiu, X.; and Huang, X. 2016. Convolutional deep neural networks for document-based question answering. In *Natural Language Understanding and Intelligent Applications*, 790–797. Springer.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. 770–778.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS 2014*, 2042–2050.

Lai, Y.; Jia, Y.; Lin, Y.; Feng, Y.; and Zhao, D. 2017. A chinese question answering system for single-relation factoid questions. In *National CCF Conference on Natural Language Processing and Chinese Computing*, 124–135. Springer.

Marcheggiani, D., and Titov, I. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP 2017*, 1506–1515.

Mitra, B.; Diaz, F.; and Craswell, N. 2017. Learning to match using local and distributed representations of text for web search. 1291–1299.

Pang, L.; Lan, Y.; Guo, J.; Xu, J.; Wan, S.; and Cheng, X. 2016. Text matching as image recognition. In *AAAI*, 2793–2799.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW 2014*, 373–374. ACM.

Su, J.; Tan, Z.; Xiong, D.; Ji, R.; Shi, X.; and Liu, Y. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI 2017*, 3302–3308.

Vashishth, S.; Dasgupta, S. S.; Ray, S. N.; and Talukdar, P. 2018. Dating documents using graph convolution networks. In *ACL 2018*, 1605–1615.

Wan, S.; Lan, Y.; Guo, J.; Xu, J.; Pang, L.; and Cheng, X. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI 2016*, 2835–2841.

Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral multi-perspective matching for natural language sentences. In *AAAI 2017*, 4144–4150.

Xie, Z. 2017. Enhancing document-based question answering via interaction between question words and pos tags. In *National CCF Conference on Natural Language Processing and Chinese Computing*, 136–147. Springer.

Yih, W. T.; Chang, M. W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL 2015*, 1321–1331.

Yih, W. T.; He, X.; and Meek, C. 2014. Semantic parsing for single-relation question answering. In *ACL 2014*, 643–648.

Yin, W., and Schütze, H. 2015. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *ACL 2015*, 63–73.

Yu, L.; Hermann, K. M.; Blunsom, P.; and Pulman, S. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Yu, M.; Yin, W.; Hasan, K. S.; dos Santos, C.; Xiang, B.; and Zhou, B. 2017. Improved neural relation detection for knowledge base question answering. In *ACL 2017*, 571–581.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, Y., and Yang, J. 2018. Chinese ner using lattice lstm. *arXiv preprint arXiv:1805.02023*.