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) – 质量(quality) – 高(high)”, but when we plan to match it with “中国人–生活–质量–高(Chinese live well)”, it would be more helpful to have the example segmented into “中国人(Chinese) – 生活(livelihood) – 质量(quality)” 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

Figure 1: A word lattice for the phrase “Chinese people have high quality of life.”
The matching score is produced by a multi-layer perceptron with 1 hidden layer based on the merged vector. The fusing and matching procedure is formulated as follows:

\[ s = \sigma(W_2 \text{ReLU}(W_1(f_{qu} \odot f_{can}) + b_1^T) + b_2^T) \]  

(1)

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

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

(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- 生活”，“国-人民-生活”，“国-人民- 生活”，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 \( G = (V, E) \), 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. \( V = \{c_{ij} | c_{ij} \text{ is word}\} \). Then, all neighbor words are connected by directed edges according to their positions in original sentence, i.e. \( E = \{e(c_{ij}, c_{jk}) | \forall i, j, k \text{ s.t. } c_{ij}, c_{jk} \in V\} \).

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. 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.

---

2https://baike.baidu.com
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 layer 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 \( G = (V, E) \) can be formulated as Eq 3:

\[
F_w = g\{f(W (v_w_1 : ... : v_w_n) + b^T)| \forall i, w_i \in V, (w_i, w_{i+1}) \in E, w[w_{i+1}] = w \}
\]

(3)

where \( f \) is the activation function, \( v_w_i \) is the input vector corresponding to word \( w_i \) in this layer, \( (v_w_1 : ... : v_w_n) \) means the concatenation of those vectors, and \( W, b \) are parameters with size \([m', n \times m], [m']\), 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:

\[
\alpha_1, ..., \alpha_t = \text{softmax}\{v_{\gamma}^T v_1 + b_{\gamma}, ..., v_{\gamma}^T v_t + b_{\gamma}\}
\]

(4)

\[
gated-pooling\{v_1, ..., v_t\} = \sum_{i=1}^{n} \alpha_i \times v_i
\]

(5)

where \( v_{\gamma}, b_{\gamma} \) are parameters, and \( \alpha_i \) 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 average 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 dataset based question answering task. 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 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 web-pages with Google’s word2vec, 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 with Tensorflow 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 (CNN-jieba), and Stanford Chinese word segmenter 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-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

---

3About 3% of the questions in the original dataset are removed since they can not link to correct entities/relations due to label errors.
4https://code.google.com/archive/p/word2vec/
5https://keras.io
6https://www.tensorflow.org
7https://pypi.python.org/pypi/jieba/
8https://nlp.stanford.edu/software/segmenter.shtml
incorporates human designed features including POS-tag in-
layer with fine tuned hyper-parameters. (Xie 2017) further
models (CNN-jieba/CNN/CTB), but outperforms our mod-
granularity fashion. They do not employ multiple kernel sizes and residual con-
structed in the siamese architecture with CNN layers,
perform the worst. Although Arc1 and CDSSM are also
distributed on the DBQA
Table 1: The performance of all models on the two datasets.
The best results in each group are bolded. * is the best pub-
lished DBQA result.

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
<th>P@1</th>
<th>KBRE P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MatchZoo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arc1</td>
<td>.4006</td>
<td>.4011</td>
<td>22.39%</td>
<td>32.18%</td>
<td>.5144</td>
</tr>
<tr>
<td>Arc2</td>
<td>.4780</td>
<td>.4785</td>
<td>30.47%</td>
<td>76.07%</td>
<td>.8518</td>
</tr>
<tr>
<td>CDSSM</td>
<td>.5344</td>
<td>.5349</td>
<td>36.45%</td>
<td>68.90%</td>
<td>.7974</td>
</tr>
<tr>
<td>MP</td>
<td>.7715</td>
<td>.7723</td>
<td>65.61%</td>
<td>86.21%</td>
<td>.9137</td>
</tr>
<tr>
<td>MV-LSTM</td>
<td>.8154</td>
<td>.8162</td>
<td>71.71%</td>
<td>86.87%</td>
<td>.9271</td>
</tr>
</tbody>
</table>

State-of-the-Art DBQA

| (Fu et al. 2016) | .8586 | .8592 | 79.06% |          |      |
| (Xie 2017)*     | .8763 | .8768 | —     |          | —    |

CNNs

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
<th>P@1</th>
<th>KBRE P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-jieba</td>
<td>.8281</td>
<td>.8289</td>
<td>75.10%</td>
<td>86.85%</td>
<td>.9152</td>
</tr>
<tr>
<td>CNN-PKU</td>
<td>.8339</td>
<td>.8343</td>
<td>76.00%</td>
<td>89.87%</td>
<td>.9370</td>
</tr>
<tr>
<td>CNN-CTB</td>
<td>.8341</td>
<td>.8347</td>
<td>76.04%</td>
<td>88.92%</td>
<td>.9302</td>
</tr>
<tr>
<td>CNN-char</td>
<td>.8803</td>
<td>.8809</td>
<td>82.09%</td>
<td>93.06%</td>
<td>.9570</td>
</tr>
</tbody>
</table>

CNNs Combined Different Word Segmentations

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
<th>P@1</th>
<th>KBRE P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>jieba+PKU</td>
<td>.8486</td>
<td>.8490</td>
<td>77.62%</td>
<td>90.57%</td>
<td>.9417</td>
</tr>
<tr>
<td>PKU+CTB</td>
<td>.8435</td>
<td>.8440</td>
<td>77.09%</td>
<td>90.48%</td>
<td>.9410</td>
</tr>
<tr>
<td>CTB+jieba</td>
<td>.8499</td>
<td>.8504</td>
<td>78.06%</td>
<td>90.29%</td>
<td>.9399</td>
</tr>
<tr>
<td>PKU+CTB+jieba</td>
<td>.8494</td>
<td>.8498</td>
<td>78.04%</td>
<td>91.16%</td>
<td>.9450</td>
</tr>
</tbody>
</table>

CNNs Combined Words and Characters

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
<th>P@1</th>
<th>KBRE P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word+Char</td>
<td>.8566</td>
<td>.8570</td>
<td>78.94%</td>
<td>91.64%</td>
<td>.9489</td>
</tr>
<tr>
<td>Char+Word</td>
<td>.8728</td>
<td>.8735</td>
<td>80.76%</td>
<td>92.78%</td>
<td>.9561</td>
</tr>
<tr>
<td>Char+Lattice</td>
<td>.8810</td>
<td>.8815</td>
<td>81.97%</td>
<td>93.12%</td>
<td>.9582</td>
</tr>
</tbody>
</table>

DGCs

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
<th>P@1</th>
<th>KBRE P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGC-ave</td>
<td>.8868</td>
<td>.8873</td>
<td>83.02%</td>
<td>93.49%</td>
<td>.9602</td>
</tr>
<tr>
<td>DGC-max</td>
<td>.8811</td>
<td>.8818</td>
<td>82.01%</td>
<td>92.79%</td>
<td>.9553</td>
</tr>
<tr>
<td>DGC-gated</td>
<td>.8790</td>
<td>.8795</td>
<td>81.69%</td>
<td>92.88%</td>
<td>.9562</td>
</tr>
</tbody>
</table>

LCNs

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
<th>P@1</th>
<th>KBRE P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCN-ave</td>
<td>.8864</td>
<td>.8869</td>
<td>83.14%</td>
<td>93.60%</td>
<td>.9609</td>
</tr>
<tr>
<td>LCN-max</td>
<td>.8870</td>
<td>.8875</td>
<td>83.06%</td>
<td>93.54%</td>
<td>.9604</td>
</tr>
<tr>
<td>LCN-gated</td>
<td>.8895</td>
<td>.8902</td>
<td>83.24%</td>
<td>93.32%</td>
<td>.9592</td>
</tr>
</tbody>
</table>

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

Analysis and Discussions

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. Match-
Pyramid(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 con-
nections, 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 mod-
els 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 in-
teration and TF-IDF scores, achieving state-of-the-art per-
fomance 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 mis-
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 re-
sult. And the models that utilize different word segmenta-
tions 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 in-
put, outperform all previous models that have sequential in-
puts, indicating that the word lattice is a more promising
form than a single word sequence, and should be better cap-
tured 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 struc-
tures and the gated pooling strategy.

To fairly compare with previous KBQA works, we com-
bine 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.
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 DGC-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 may 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 in-appreciable 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. The efficiency may be further improved if the net-
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

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”(station) can match the “网址”(Internet addresses), “网”(web), “址”(addresses) and also related to “导航”(navigate), from which it may infer 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


