Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 1774–1782

Procedia Computer Science

www.elsevier.com/locate/procedia

21th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

Sentence selection with neural networks using string kernels

Mihai Masala[a,b], Stefan Ruseti[a,b*], Traian Rebedea[a,b]

[a] *University Politehnica of Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania*
[b] *Autonomous Systems, 22 Tudor Vladimirescu, 050883 Bucharest, Romania*

Abstract

In recent years, there have been several advancements in question answering systems. These were achieved both due to the availability of a greater number of datasets, some of them significantly larger in size than any of the existing corpora, and to the recent advancements in deep learning for text classification. In this paper, we explore the improvements achieved by employing neural networks using the features computed by a string kernel for sentence/answer selection. We have validated this approach using two different standard corpora used as benchmarks in question answering and we have found a significant improvement over string kernels and other unsupervised methods for sentence selection.

Keywords: Question answering; Sentence selection; String kernels; Neural Networks

1. Introduction

Question answering (QA) is one of the most popular applications at the border of natural language processing (NLP) and information retrieval. The final objective of question answering is to build a general system able to handle open-domain questions and to answer either with short phrases, in case of fact-based QA[1], or with longer phrases or even sentences or paragraphs containing the answer to more complex questions. The latter is called sentence or answer selection[2] and aims to identify the phrase (span of text), most often a sentence or a group of sentences, that contains the answer for a question.

For open-domain complex QA systems, the candidates for sentence (answer) selection are retrieved from large text databases. In the case of large-scale web QA, this database is composed of all available web pages. However, due to the difficulty of these problems, most datasets are either domain-specific and/or also offer a list of candidates for each question. In the current work, we propose a comparison between string kernels used either standalone - in an unsupervised manner - or together with a neural network classifier to discriminate the correct answer from the list of

* Stefan Ruseti
E-mail address: stefan.ruseti@cs.pub.ro

candidates. The results show that using neural networks over string kernels provide better results than string kernels alone or other unsupervised methods, but below the state of the art achieved with deep neural networks for answer selection.

The paper continues with a section on related work in sentence selection for question answering, including a short description of the two datasets used during evaluation. Section 3 describes the proposed method that uses a simple multi-layer perceptron (MLP) network using features from string kernels to discriminate between different possible candidates (sentences or large documents) as the most probable answer for a given question. In section 4 we are presenting the results achieved on two standard corpora, including a comparison with existing methods. Section 5 contains a short discussion on the improvements obtained by the proposed approach versus unsupervised string kernels for the same data sets. The paper ends with concluding remarks and future improvements.

2. Related Work

2.1. Datasets

SQuAD[3] is a reading comprehension dataset which contains 536 articles, usually a couple of paragraphs in size, and more than $100,000$ questions about the text in these articles. The answer for each question is a span of text from the article which completely encompasses an answer for the human reader. The span can range from several words to a couple of sentences. The dataset is very suitable for comparing deep learning models thanks to the large number of included questions. A list of state of the art models for this task is published on their website[1], some of these also being presented in the next subsection.

While SQuAD is open-domain, the second dataset used in our research is called InsuranceQA[4] and it is a domain specific QA dataset. The content of this dataset consists of real world questions from users and answers from experts with deep domain knowledge in insurance. In this paper we used both versions of the corpus. The first version consists of four parts: train set, development set and two test sets. The development and tests set feature a candidate pool of size 500. For this reason we also used for training the same candidate size by adding the ground truth answers to the pool and then random sampling answers from the complete answer pool until reaching 500 candidates. The second version of the dataset consists of three parts: train set, validation set and test set. For the second version the train file already has the pool size of 500 candidates for each question. For both versions of the dataset, a question can have more than a single correct answer.

Table 1 contains some simple statistics for the two datasets. InsuranceQA is smaller, however it also has a domain-specific vocabulary. For both datasets, the question is smaller than the average candidate answer, however, the average size of an answer is much larger for InsuranceQA than for SQuAD. This can be explained by the fact that the former contains longer answers formed by more than a sentence (usually a paragraph), while the latter has been used for sentence selection in our experiments (thus, a candidate contains a single question).

Table 1. QA datasets statistics

Dataset	Number of questions	Average #words in question	Average #words in candidate answer
SQuAD Train	87,636	10.05	23.47
SQuAD Dev	10,600	10.30	24.27
InsuranceQA V1 Train	12,887	8.14	96.53
InsuranceQA V1 Test set 1	1,800	8.16	96.49
InsuranceQA V1 Test set 2	1,800	8.16	96.50
InsuranceQA V2 Train	12,889	9.37	172.73
InsuranceQA V2 Test set	2,000	9.40	172.36

[1] https://rajpurkar.github.io/SQuAD-explorer/

2.2. Deep Learning Solutions

In general, any model based on neural networks computes separate representations for the question and for each candidate and then computes the similarity between them (in the case of answer selection), or generates start and end tokens of the span of text delimiting the answer in the text (if the answer is extracted from text). The representation is most often computed with a Bi-directional Long Short-Term Memory (LSTM) network[5], or a convolutional neural network (CNN) on words like the one used by Kim et al. (2014)[6]. Since the question and the answer have varying length, the dynamic number of outputs of the Bi-LSTM needs to be transformed into a fixed-length representation by applying a simple max or average pooling, concatenating the first and last outputs, or even by applying another CNN on top of them[7].

One of the improvements recently brought to neural QA is using attention-based models. The intuition for this approach is that looking at the question can help you understand what are you looking for in the candidate text by giving different weights for each word in the text. For example, Santos et al. (2016)[8] combined the outputs for the question and for the text containing the answer, obtained from convolution or LSTM, into a single matrix, from which the attention weights for each of them are extracted. These weights were then used to modify the output representation of both the question and of the candidate text.

Another solution is to combine a deep learning model with more traditional features. Tymoshenko et al. (2016)[9] successfully combined a CNN with tree kernels and improved the results on two QA datasets. The authors tested two different means of combining the two models: using the CNN extracted features in a polynomial kernel along with the tree kernels or using the predictions from both models to train a logistic regression classifier.

2.3. Character-Level Methods

String kernels[10] are a character comparison technique that uses different sized character n-grams. In order to be effective, large n-grams are usually used for comparison. However, this would require projecting the text into an extremely high dimensional vector space. Instead of computing this representation, string kernels use a kernel function that simulates the cross-product of two texts in that vector space.

Different types of string kernels have been proposed, but the common idea among them relies on counting the common substrings of a fixed length in the two texts which are compared. Usually, the kernels vary in the length of the n-grams, which can be between 2 and 10 characters, and in the function used for computing the overlap of the n-grams. Typical kernel functions are spectrum, presence and intersection, as they are described in Ionescu et al. (2014)[11]. Using ridge regression or Support Vector Machines (SVM) over the features computed by presence and intersection string kernels has showed to provide state of the art results for various NLP applications, such as native language identification[12].

The idea of using a supervised method for combining multiple kernels is not new. A significant amount of work has been done on this topic, different multiple kernels learning methods showing improvements over single kernel methods[13]. These methods can eliminate kernels that are not useful, while combining relevant kernels into a function more suitable for the task at hand.

In the last years, more and more solutions use character level information for a variety of NLP tasks. Instead of using a vector representation for each word, these methods use a vector for each character and then combine these vectors into a word representation. Using this method, Ling et al. (2015)[14] employ a Bi-directional LSTM[5] over the computed word representations and achieve state of the art results in part of speech tagging.

Other ways of combining the character embeddings into words were also proposed. Kim et al. (2016)[15] use a CNN and a highway network[16] to compute word representations, which are then passed through a LSTM layer in order to model a sentence. Another solution is to combine word representations with a character-based encoding, as in the model proposed by Seo et al. (2016)[17].

Although it might seem surprising, character-level methods proved to be effective in language representation, either alone or combined with word embeddings. Furthermore, it was recently shown that the written form of words alone can be used to predict their meaning[18].

Another advantage of character-based methods is the lack of need for any language dependent tools. Also, models using character representations use fewer parameters than the same ones based on word representations, making them easier to train and also proving useful for smaller corpora.

3. Method and Experiments

3.1. String Kernels

String kernels are kernel functions that operate on strings or other type of sequences (e.g. DNA). The kernel function works as a function for computing the similarity between two documents. The higher the value of the kernel, the more similar the two documents are. A big advantage of this method is that the documents can have different sizes.

The most frequently used string kernel functions are spectrum, presence, and intersection[11]. Spectrum kernel (eq. 1) computes the product of the frequencies (number of occurrences) of shared n-grams. For the presence kernel (eq. 2), the number of occurrences is ignored and the product is computed between presence bits. The intersection kernel (eq. 3) returns the minimum of the frequencies for each n-gram in the two compared documents. In order to be able to compare documents (both questions and answers) of different sizes, the normalized versions of these kernels were used.

$$k_p(s,t) = \sum_{v \in \sum_p} num_v(s) \cdot num_v(t) \tag{1}$$

$$k_p^{0/1}(s,t) = \sum_{v \in \sum_p} in_v(s) \cdot in_v(t) \tag{2}$$

$$k_p^{\cap}(s,t) = \sum_{v \in \sum_p} min\{num_v(s), num_v(t)\} \tag{3}$$

where:

- $\sum_p$ = all p-grams of a given size p
- $num_v(s)$ = number of occurrences of string (n-gram) v as substring in document s
- $in_v(s)$ = 1 if string (n-gram) v occurs as substring in document s, 0 otherwise

3.2. Neural Network using String Kernels

Inspired by Ionescu et al. (2014)[11], we wanted to find a combination of these string kernels that better captures the measure of similarity between a question and its answers.

We used string kernels as a measure of similarity between the question and the candidate answer. From each type of kernel (e.g. intersection, spectrum, presence) we extracted a feature vector $v \in \mathcal{R}^5$ based on the following n-gram (inclusive) ranges: 1-2, 3-4, 5-6, 7-8, and 9-10. The feature vector used for training a supervised classifier over the string kernel features was obtained by concatenating these three vectors, resulting in a vector $V \in \mathcal{R}^{15}$.

We then trained a simple feed-forward MLP with one hidden layer, which computes a score for each candidate answer. The candidate answer with the largest score computed by the MLP was selected as the answer to a question provided by our system. For the hidden layer of the MLP we opted for a layer of size 8, and have used a batch size of 100 in combination with Adam optimizer for training. For the loss function we used hinge loss, as defined in eq. 4, similar to the one proposed by Hu and Lu (2016)[19] for finding similarities between two sentences (e.g. paraphrase identification and detecting the responses for a given tweet) with a CNN.

$$e(q, s^+, s^-) = max(0, M + sim(q, s^-) - sim(q, s^+)) \tag{4}$$

where:

- q is the question
- y^+ is (one of) the correct answer(s)
- y^- is an incorrect answer from the candidate list
- $sim(q, s)$ is the similarity score computed by the MLP between the representations of the question q and candidate answer s

- M is the desired margin between positive and negative examples

One of the main advantages of this method is that it does not require large datasets and can be easily trained. Moreover, being a character-based model, it is language independent. Of course, there are also disadvantages coming from the simplicity of this model. The most important one is that the answer will be selected using only a combination of morpholexical features (n-grams), thus ignoring more complex syntactical or semantic information which cannot be completely captured by n-grams and string kernels. In the next section, we show how the proposed supervised method compares with the unsupervised string kernels used for the same data and also compare our method with existing deep learning solutions for answer selection.

4. Results

We evaluated the proposed method using the two QA datasets described in section 2. For each corpus, we have made a comparison with several unsupervised methods as baselines, including using string kernels for computing the similarity between the question and each candidate. In addition, for the InsuranceQA dataset we have also compared our results with the state of the art deep learning models. To the best of our knowledge, these are the first reported results of using string kernel-based methods for the two datasets.

The Word2Vec embeddings [20] used in both experiments were the pretrained versions on the Google News dataset [2]. The string kernels were computed with an existing library [3].

4.1. InsuranceQA Dataset

For both versions of this dataset, we have used a value of 0.1 for the margin of the hinge loss function. The neural network was trained for 500 epochs with a learning rate of 0.1. These hyperparameters were selected using a grid search (e.g for the margin, the tested values were 0.1, 0.5, 0.8, and 1) performed on the dev set of InsuranceQA v1.

As baselines, we have used three different unsupervised methods for computing the similarity between the question and a candidate answer:

- Bag-of-words: The idf-weighted bag-of-words feature vector was computed for each question and all of its answer candidates. The answer was selected based on the maximum cosine similarity between a candidate and the question.
- Word2Vec bag-of-words: The feature vector was based on the idf-weighted sum of the Word2vec representation of the words for each question and for all its answer candidates. The answer was then selected based on the maximum cosine similarity.
- String kernels: We computed the similarity between the question and all of its answer candidates using the three (intersection, presence, and spectrum) kernels on n-grams of size between 3-7. The candidate answer with the highest similarity was selected.

Besides these unsupervised baselines, we also compared some of the best performing solutions on this task. For InsuranceQA v1, we added three models based on LSTMs, each using a different type of attention mechanism, together with the results reported by their authors [7,8]. For InsuranceQA v2, the only reported result that we could find was an unsupervised method that combined an IR system with a simple max- plus min-pooling applied on the word embeddings from each question and candidate pair [21]. For comparison, we have also used another supervised method, a logistic regression over string kernels. The results are reported in Table 2, where we have used precision for the first result returned by each method. A discussion of these results follows in the next section.

[2] https://code.google.com/archive/p/word2vec/
[3] http://string-kernels.herokuapp.com

Table 2. Precision@1 for the InsuranceQA dataset

Method	V1 Test set 1 (%)	V1 Test set 2 (%)	V2 Test set (%)
Bag-of-words	49.4	47.0	15.8
Word2Vec Bag-of-words	39.2	37.6	10.5
Intersection kernel	51.6	48.5	21.9
Presence kernel	51.7	48.6	21.9
Spectrum kernel	5.1	3.2	2.5
FastHybrid [21]	N\A	N\A	22.7
Conv-pooling LSTM [7]	67.5	64.4	N\A
Attentive LSTM [7]	69.0	64.8	N\A
AP-BILSTM [8]	71.7	66.4	N\A
Logistic regression using string kernels	42.3	41.2	15.2
Neural network using string kernels	54.1	50.2	25.0

4.2. SQuAD Dataset

Since SQuAD is a machine comprehension dataset, we needed to transform the task into a sentence selection one. In the original dataset, each question had an answer that was included in the given paragraph and the task was to select the smallest span of text from the paragraph that contained this answer. For sentence selection, we considered that the sentence that contained this span of text is the correct one, while the other sentences are candidates. The span of text containing the answer ranged from several words to a couple of sentences. Thus, if the answer spanned on more than one sentence we removed that entry from the sentence selection dataset. However, this rarely happened - only about 1.2% of the questions in the standard SQuAD dataset had an answer spanning over more than a single sentence.

Because the original corpus was modified for the sentence selection task, there are no other results to compare with besides the baselines detailed below.

- N-gram overlap: Compute unigram and bigram overlap between the question and each of its answer/sentence candidates, then select the one with the highest score.
- Neural network using Word2Vec: Average over the Word2Vec embeddings of the words in the question and in each candidate. Over these features, question and candidate average embedding, train a neural network with one hidden layer of size 100 using hinge loss similar to the MLP proposed for string kernels.
- String kernels: We computed the similarity between the question and each of its answer candidates using the three (intersection, presence, and spectrum) kernels on n-grams of size between 3-7. The candidate with the highest score was selected.
- Basic CNN: The model proposed by Kim (2014)[6] was transformed into a siamese network, so it can receive two sentences as input. The outputs from both networks were passed to a decision layer that computed the similarity. The model used word embeddings of size 50, which were learnt on the current corpus.

Table 3. Precision@1 for the SQuAD sentence selection dataset

Method	Dev set (%)
N-gram Overlap	74.8
Neural network using Word2vec	63.8
Basic CNN	48.0
Intersection kernel	79.5
Presence kernel	79.5
Spectrum kernel	45.0
Logistic regression using string kernels	79.9
Neural network using string kernels	81.0

For training the MLP using string kernels, we used a hinge loss margin of 0.1 with a batch size of 100. The network was trained for 20 epochs using a learning rate of 0.1. As for the previous experiment, we have also used logistic regression over string kernels. The results obtained using our proposed method and the baselines are reported in Table 3, while the results are commented in the next section.

The candidate answers in this dataset are closely linked together semantically and also have a high cohesion, as all candidates are individual sentences from the same paragraph. One of our assumptions was that there might not be enough information in just a single sentence to determine the correct answer. Thus, we ran another experiment where we added features from the previous and next sentence for each candidate. Looking at the results presented in Table 4, there is an insignificant improvement for the MLP using string kernel features when considering both the previous and subsequent sentence, in addition to the candidate.

Table 4. Precision@1 for the SQuAD sentence selection dataset using previous and next sentence

Considered features	Cross-validation on train set with five folds (%)
Current sentence (candidate)	80.05
Previous and current sentence	79.99
Previous, current, and next sentence	80.07

5. Discussion

For both InsuranceQA and SQuAD, string kernels proved to be the most effective unsupervised method for sentence selection. In these cases, the best kernel was statically selected and the chosen answer was the one with the highest similarity. With regards to this aspect, it can be easily observed from Tables 2 and 3 that the spectrum kernel has a very poor performance compared with the other two kernel functions. For spectrum kernel, the frequency of a p-gram has a large contribution to the kernel score. Because of this, the more frequent p-grams tend to have a much higher contribution than the less common ones making the contribution of a relevant but singular p-gram negligible in a long answer. This effect is even more visible for the InsuranceQA dataset, where candidates have an average length of almost 100 words, compared with the questions with an average of 8 words (also see Table 1).

On top of this unsupervised string kernel approach, we added a simple neural network with one hidden layer that learns how to combine all the kernels and features from each kernel (score per n-gram of a given size). Instead of statically choosing the best kernel for the whole dataset, the network can learn which are the more useful kernels and how to combine those scores into a more predictive function. The extra hidden layer of the network, compared with the logistic regression approach, allows more complex functions for combining the kernels, which proved to be necessary for these two datasets.

In the case of SQuAD, the network only improved the static result by 1.5%. This might be caused by the fact that the original score was already very high. On the other hand, although the number of candidate answers is much lower than in the case of InsuranceQA, the task is more difficult due to the similarity of different candidates as they are extracted from the same paragraph. Thus, the information needed to answer a question might not be included in a single sentence, so treating the sentences independently can lead to contradictory results. This could also be an explanation for the poor results of supervised methods on this task, especially when compared with the original SQuAD comprehension task where the best solutions are complex deep-learning models, that obtain over 80% F1 score. Although returning the sentence seems easier than directly returning the answer, these models have the advantage of using the whole paragraph as input, so no information is lost by splitting the text into sentences. Another explanation is that there were not enough training examples to learn how to discriminate between candidates. Although the dataset contains over $100,000$ questions, it uses only 536 different articles/paragraphs for searching the answer. Due to this, it may be difficult for supervised models to learn a representation for an article.

For the InsuranceQA dataset, the unsupervised methods have significant lower scores than the supervised neural networks. But they still perform well, considering the large number of candidate answers. However, since the false answers in the candidate list were sampled randomly, lexical features are good enough to discriminate between positive and negative examples most of the time. Only semantic information (in a bag of words fashion), as in the case of the

Word2Vec baseline pretrained on a general corpus, does not help compared to lexical methods. Since this is a specific domain dataset, training the word embeddings on texts related to insurances might help.

One possible explanation for the success of string kernels compared to bag-of-words approaches is that string kernels can be used to tackle information for more than a single word (the example of longer n-grams) and they also recognize words that have similar word functions (e.g. using suffixes - small n-grams) or that have the same base word (again using n-grams). Moreover, the neural network finds a more appropriate combination of the features (n-gram scores) computed by each string kernel function for the task at hand. Using a MLP over string kernels provides a relative improvement over string kernels alone for all datasets ranging from 2% for SQuAD to 11% for Insurance QA v2. Moreover, the result obtained for Insurance QA v2 are above results reported in other papers we have identified.

6. Conclusions

In this paper we compared string kernels with both supervised and unsupervised methods on the sentence selection task. Among the unsupervised methods, the standalone string kernels obtained the best result on the two QA datasets used in this paper, InsuranceQA and SQuAD. We have also shown that a simple neural network used on top of different kernel types and sizes can improve the performance of string kernels, but it is less effective than deep-learning models with attention for the same task. The main advantages of our proposed method is that it is language and domain-independent and that it requires much smaller datasets to train.

The most important improvement to our approach is to also consider semantic information when deciding the most suitable candidate. For example, the network might also take as input the cosine similarity between the Word2Vec average embedding scores of the question and candidate. This way, the network should learn when to use the semantic similarity score and when the kernel scores are more appropriate.

Acknowledgements

This work was partially supported by the IAVPLN project (ID: POC-A1.2.1D-2015 P_39_287) funded by the Romanian Ministry of European Funds, as well as by University Politehnica of Bucharest through the Excellence Research Grants UPB-GEX Nr. 13/30.09.2016.

References

1. E. M. Voorhees, The TREC-8 Question Answering Track Report (1999). doi:10.1017/S1351324901002789.
2. J. Otterbacher, G. Erkan, D. R. Radev, Using random walks for question-focused sentence retrieval, in: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2005, pp. 915–922.
3. P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, SQuAD: 100,000+ Questions for Machine Comprehension of Text, Emnlp (ii) (2016) 2383–2392. arXiv:1606.05250.
4. M. Feng, B. Xiang, M. R. Glass, L. Wang, B. Zhou, Applying deep learning to answer selection: A study and an open task, in: 2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015 - Proceedings, IEEE, 2015, pp. 813–820. arXiv:1508.01585, doi:10.1109/ASRU.2015.7404872.
5. A. Graves, J. Schmidhuber, Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures, Neural Networks 18 (5) (2005) 602–610. doi:10.1016/j.neunet.2005.06.042.
6. Y. Kim, Convolutional Neural Networks for Sentence Classification, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014) (2014) 1746–1751arXiv:arXiv:1408.5882v1, doi:10.1109/LSP.2014.2325781.
7. M. Tan, C. dos Santos, B. Xiang, B. Zhou, Improved Representation Learning for Question Answer Matching, Acl (2016) 464–473.
8. C. dos Santos, M. Tan, B. Xiang, B. Zhou, Attentive Pooling Networks (2016). arXiv:1602.03609.
9. K. Tymoshenko, D. Bonadiman, A. Moschitti, Convolutional Neural Networks vs . Convolution Kernels : Feature Engineering for Answer Sentence Reranking, North American Association for Computational Linguistics (2016) 1268–1278.
10. H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, Text Classification using String Kernels, Journal of Machine Learning Research 2 (Feb) (2002) 419–444. doi:10.1162/153244302760200687.
11. R. T. Ionescu, M. Popescu, A. Cahill, Can characters reveal your native language? A language-independent approach to native language identification, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014) 1363–1373.
12. R. T. Ionescu, M. Popescu, A. Cahill, String kernels for native language identification: insights from behind the curtains, Computational Linguistics.
13. M. Gönen, E. Alpaydn, Multiple Kernel Learning Algorithms, Journal of Machine Learning Research 12 (2011) 2211–2268.

14. W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, I. Trancoso, Finding Function in Form: Compositional Character Models for Open Vocabulary Word RepresentationarXiv:1508.02096.
15. Y. Kim, Y. Jernite, D. Sontag, A. M. Rush, Character-Aware Neural Language Models, Aaai (2016) 2741–2749arXiv:1508.06615.
16. R. K. Srivastava, K. Greff, J. Schmidhuber, Highway networks, arXiv preprint arXiv:1505.00387.
17. M. Seo, A. Kembhavi, A. Farhadi, H. Hajishirzi, Bidirectional Attention Flow for Machine ComprehensionarXiv:1611.01603.
18. E. Darío Gutiérrez, R. Levy, B. K. Bergen, Finding Non-Arbitrary Form-Meaning Systematicity Using String-Metric Learning for Kernel Regression, in: ACL, 2016, pp. 2379–2388.
19. B. Hu, Z. Lu, Convolutional Neural Network Architectures for Matching Natural Language Sentences, Advances in Neural Information Processing Systems (2014) 1–9arXiv:arXiv:1503.03244v1.
20. T. Mikolov, K. Chen, G. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality, Nips (2013) 1–9arXiv:1310.4546.
21. L. Wang, M. Tan, FastHybrid : A Hybrid Model for Efficient Answer Selection, in: Proceedings of the 26th International Conference on Computational Linguistics (COLING-16), 2016, pp. 2378–2388.