# Predicting and Integrating Expected Answer Types into a Simple Recurrent Neural Network Model for Answer Sentence Selection

Sanjay Kamath, Brigitte Grau, Yue Ma

# Predicting and Integrating Expected Answer Types into a Simple Recurrent Neural Network Model for Answer Sentence Selection

Sanjay Kamath[1,3], Brigitte Grau[1,2], and Yue Ma[3]

[1] LIMSI, CNRS, Université Paris-Saclay, Orsay, France
[2] ENSIIE, Université Paris-Saclay, Évry, France
[3] LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France
{sanjay, ma}@lri.fr, bg@limsi.fr

**Abstract.** Since end-to-end deep learning models have started to replace traditional pipeline architectures of question answering systems, features such as expected answer types which are based on the question semantics are seldom used explicitly in the models. In this paper, we propose a convolution neural network model to predict these answer types based on question words and a recurrent neural network model to find sentence similarity scores between question and answer sentences. The proposed model outperforms the current state of the art results on an answer sentence selection task in open domain question answering by 1.88% on MAP and 2.96% on MRR scores.

**Keywords:** Question answering · Deep learning · Answer sentence selection · Expected answer types · Sentence similarity

## 1 Introduction

Question answering systems in recent times have mainly been dominated by neural network approaches that fetch state of the art results across different NLP tasks. Open domain question answering tasks include answer sentence selection, reading comprehension, multi-hop reasoning and reading etc. An example of a question answer pair from a dataset:
*Q: How a water pump works?*
*A: pumps operate by some mechanism (typically reciprocating or rotary), and consume energy to perform mechanical work by moving the fluid.*
An answer sentence selection model would retrieve the entire sentence from a paragraph as an answer. A common goal of the neural network models is to build end to end approaches which do not rely on intermediate tools or data provided by other systems. Some recent works such as BERT [3] and ELMO [11] use pre-trained language models trained with large neural network architectures and use it to fine tune downstream NLP tasks. These methods outperform current state of the art systems for reading comprehension as well as many other tasks. However, training such models on large datasets and the requirement of large scale computation power is sometimes not a feasible solution.

Other state of the art models such as QANet [19] on SQUAD and other end to end approaches try to implicitly learn information such as entity types, part of the speech tags, named entities, syntactic dependencies etc. and perform downstream tasks. But the challenge still remains in understanding whether or not they utilize such information implicitly or just overfit over the datasets and their unintended bias. A feasible yet challenging approach would be to utilize both the power of neural networks approaches and explicit information such as entity types, dependencies, tags, together.

Expected Answer Types (referred to as EAT hereafter) is one such vital information which is important for question answering systems to detect which type of answers do the questions require. Some examples of EAT with questions are listed below:

Question: Which NFL team represented the AFC at Super Bowl 50?
Expected Answer Type: HUM.

Question: Where was franz kafka born ?
Expected Answer Type: LOC.

[15] refer to this information as *Question Classes* in their work and show a significant improvement over a previous state of the art DNN model on TrecQA dataset which uses only word level information.

Our contributions in this article are as follows. We introduce two different ways of using *Question Classes* which is further referred as *EAT or Expected Answer Types* and experiment with several datasets along with TrecQA to determine if this would work better for a wider range of large scale datasets by using a simple model of a recurrent neural network which uses a pre-attention mechanism. To annotate other datasets apart from TrecQA, with EAT information, we propose a multiclass classifier model which is trained on a dataset built by using an existing rule-based system which predicts EAT for questions.

We report our findings on WikiQA, SQUAD-Sent and TrecQA dataset performance and show that we outperform state of the art results on TrecQA dataset[4] by the two different ways of highlighting Expected Answer Types in the data.

## 2 Related Work

Answer sentence selection task has been extensively studied with different approaches ranging from n-gram models to neural network models. In former feature based QA systems, the Expected Answer Type (EAT) has been shown as a very important feature [7]. The EAT corresponds to an entity type organized in answer type taxonomies, as in [8] for the open domain or semantic types in biomedical domain as in [5].

Recent works on this task focus mainly on convolutional neural network approaches. [14] propose a CNN model using learning to rank approach, which

[4] https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)

computes a representation of both entries, candidate passage and question, and a similarity between these two representations using a pooling layer followed by similarity matrix computation. In [18], the similarity of the two entries is evaluated by computing interactions between words of the two texts by an attention layer. [4] propose a Multi-Perspective CNN for this task which was further used by [13] with a triplet ranking loss function to learn pairwise ranking from both positive and negative samples. [15] use the same model but use Question Classes to enhance the dataset with highlighting entities in it. Highlighting entities were done by mainly two ways called Bracketing (appending a special token before and after the entity occurrence) and Replacement (replacing the entity word with a special token) methods. Our work uses a similar technique by replacing the entity word with special tokens but allows to learn them according to the expected types. The leaderboard of TrecQA evaluation[4] reports the state of the art scores from different methods reported by several articles.

## 3 Answer Sentence Selection

Answer sentence selection is a question answering task which is also referred sometimes as sentence reranking task. The task involves reranking a set of sentences $S = \{S_1, ....., S_m\}$ for a question $Q$, so that the correct sentences are ranked first. Sentence set $S$ can contain the mixture of both negative and positive sentences relevant to the question, often more than one positive sentence.

We model this task as a pairwise similarity scoring task. For each sentence related to a question, we compute a similarity score against the question sentence and answer sentence. i.e., ($Q_i - S_{i,j}$, $Q_i - S_{i,j+1}$, $Q_i - S_{i,j+2}$, ....$Q_i - S_{i,j+n}$).

### 3.1 RNN-Similarity

Recurrent neural networks such as LSTMs and GRUs are widely used in several NLP tasks like machine translation, sequence tagging, and question answering tasks such as reading comprehension and answer sentence selection. We propose a simple model with recurrent neural networks and an attention mechanism to capture sequential semantic information of words in both questions and sentences and predict similarity scores between them. We refer to this model further in this article as *RNN-Similarity* model. Figure 1 shows the architecture of the model.

Question words $Q = \{q_1, ....., q_m\}$ and Sentence words $S = \{s_1, ....., s_n\}$ are sequences which are encoded using an embedding layer of dimension $D$.

$$E(Q) = \{E(q_1), .., E(q_m)\} \tag{1}$$

$$E(S) = \{E(s_1), .., E(s_n)\} \tag{2}$$

A pre-attention mechanism captures the similarity between sentence words and questions words in the same layer. For this purpose, a feature $\mathcal{F}align$ shown in Equation 3 is added as a feature to the LSTM layer.

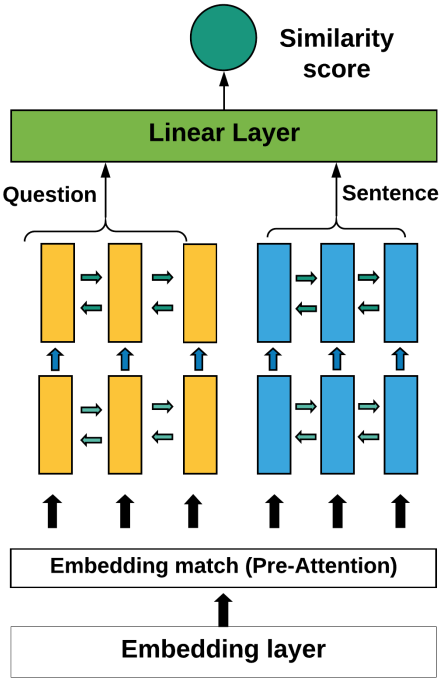


**Fig. 1.** Proposed RNN-Similarity model

$$\mathcal{F}align(p_i) = \Sigma_j a_{i,j} E(q_j) \tag{3}$$

Where $a_{i,j}$ is,

$$a_{i,j} = \frac{exp\left(\alpha(E(s_i)) \cdot \alpha(E(q_j)\right)}{\Sigma_{j'}\ exp(\alpha(E(s_i)) \cdot \alpha(E(q_{j'}))} \tag{4}$$

which computes the dot products between nonlinear mappings of word embeddings of question and sentence.

The above process is similar to [1] who use LSTMs to model Question and Paragraph to encode the words for reading comprehension task. We use 3-layer Bidirectional LSTM layers for both question and sentence encodings.

$$\{E(q_1), .., E(q_n)\} = \text{Bi-LSTM}(\{\tilde{E}(q_1), .., \tilde{E}(q_n\}) \tag{5}$$

$$\{E(s_1), .., E(s_n)\} = \text{Bi-LSTM}(\{\tilde{E}(s_1), .., \tilde{E}(s_n\}) \tag{6}$$

The LSTM output states are further connected to a linear layer and a sigmoid non-linear activation function is applied on the output of the linear layer which outputs the score ranging between 0-1, which signifies the similarity between the question and the answer sentence.

For the Expected Answer Types (EAT) version of question and sentences, we create special tokens for the entity type that are used for encoding the question $Q$ and each sentence $S$.

### 3.2 Highlighting Single Entity and Multiple Entity Types

| - | Method | Question | Sentence |
|---|---|---|---|
| 1 | Original text | Who is the author of the book, The Iron Lady: a biography of Margaret Thatcher | in The Iron Lady, *Young* traces ..... the greatest woman political leader since *Catherine the Great*. |
| 2 | Replacement - [15] (EAT Single type) | Who is the author of the book, The Iron Lady: a biography of Margaret Thatcher *max_entity_left* *entity_left* | in The Iron Lady, *max_entity_left* traces ..... the greatest woman political leader since *entity_left*. |
| 3 | EAT (Different types) | Who is the author of the book, The Iron Lady: a biography of Margaret Thatcher *max_entity_left* *entity_hum* | in The Iron Lady, *max_entity_left* traces ..... the greatest woman political leader since *entity_hum*. |
| 4 | EAT (MAX + Different types) | Who is the author of the book, The Iron Lady: a biography of Margaret Thatcher *max_entity_hum* *entity_hum* | in The Iron Lady, *max_entity_hum* traces ..... the greatest woman political leader since *entity_hum*. |

**Table 1.** Three methods for replacing entities along with an example from TrecQA dataset

The authors of [15] propose a method of replacing words by special token embeddings for highlighting entities that catch the EAT entity in sentences. In our work, this method is referred to as "EAT (single type)" in the following experiments. The entities belong to (HUM, LOC, ABBR, DESC, NUM or ENTY). HUM refers to a description, group, individual, title. LOC refers to city, country, mountain, state. ABBR refers to abbreviation, expansion. DESC refers to a definition, description, manner, reason. NUM refers to numerical values such as code, count, date, distance, money, order etc. ENTY refers to a numerous entity types such as animal, body, color, creation, currency, disease etc. More details regarding the taxonomy can be found in [9].

The entities, irrespective of which class they belong to, are treated similarly by replacing them by two special tokens *entity_left* for entity occurrences and *max_entity_left* for maximum occurring entity that corresponds to an entity that is at least twice the number of occurrences when compared to the second

maximum occurring entity. Entity types are recognized using the named entity recognition tool. When an entity type in a sentence matches the EAT from the question, *entity_left* token is used to replace the entity mentions in the sentences; same applies for the maximum occurring entity token *max_entity_left* as well.

Our proposition is to replace an entity according to the type it belongs to instead of replacing all kinds of entity by just one word i.e. *entity_left*. We do it based on the different types of EAT it belongs to based on the taxonomy used in the original work. The intuition behind this method is that the model would learn to better map the relations between question words and specific entity type tokens when used in a model with attention mechanisms, rather than learning the relation between question words and the same generic entity type token for all entities. This way, we can learn a different behaviour with an entity about location and with an entity about a person for example.

The example in Table 1 line 3 refers to an example that has an EAT as "HUM" from the taxonomy, so we replace it as *entity_hum*. We do the same for other expected answer types such as *entity_loc* for " LOC" type, *entity_enty* for " ENTY" type, *entity_num* for "NUM" type, *entity_desc* for "DESC" type, *entity_abbr* for "ABBR" type. We replace the entity mentions in the text whose types are matching the EAT from questions.

We also experiment with a variant where the *max_entity_left* is replaced with the entity type along with other entities. If the maximum entity is of type "HUM", we replace it as *max_entity_hum*. This method is referred to as "EAT (MAX + different types)" in the following experiments. We create a random word embedding ranging between (-0.5 - 0.5) with dimension $D$ for each of the EAT words and encode the word with this embedding when it appears in all our experiments.

## 4 Experiments

We perform experiments on three datasets namely 1) TrecQA, 2) WikiQA, 3) SQUAD-Sent with and without EAT annotations. Thus we had to develop our own annotation tools.

### 4.1 Annotation of the EAT

Since SQUAD-EAT (see section 4.3) is the result of a rule-based method with a high accuracy score (97.2% as reported in [9]), we use it to train a multiclass classifier based on a CNN model for text classification[5] by [6], by modifying the outputs into a multi-class setting. We further refer to this as EAT Classifier. We use 300 dimensions GloVe embeddings by [10].

The output classes of the classifier refer to a type based on the taxonomy such as *ABBR, DESC, ENTY, HUM, LOC, NUM* and a "NO_EAT" class to signify an EAT which is not in the above list of classes. We do not use the fine

[5] https://github.com/cmasch/cnn-text-classification

level taxonomy in this work because of a resulting large number of classes with sparse distribution of samples in the dataset.

Below is an example from SQUAD-EAT with HUM:

**Question**: Which NFL team represented the AFC at Super Bowl 50?

**Expected Answer Type**: HUM.

We train the multi-class classifier model using the SQUAD-EAT dataset which gets an accuracy score of 95.17% on the SQUAD-EAT dev in our experiment, according to the annotation done by [15] as reference.

### 4.2 Annotation of the entities

| EAT | Spacy annotated tag |
|---|---|
| HUM | PERSON, ORG, NORP |
| LOC | LOC, GPE |
| ENTY | PRODUCT, EVENT, LANGUAGE, WORK_OF_ART, LAW, FAC |
| NUM | DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL |

**Table 2.** Spacy named entity annotation scheme following OntoNotes 5 corpus mapped with EAT types

We detect the entities in the sentences using Dbpedia Spotlight tool by [2]. The detected entities by spotlight are verified for their entity type match using the Spacy NER tool which is mapped to EAT using the mapping shown in table 2. Only the matching entities are highlighted and others are discarded. We replace the special token by adding one for the maximum occurring entity, which is described in Section 3.2.

### 4.3 The Data

TrecQA dataset is a standard dataset used to benchmark state of the art systems for answer sentence selection task. The authors of [9, 15] provide the EAT annotations for the TrecQA dataset based on their rule-based approach.

We modify the QA dataset SQUAD by [12] designed for machine comprehension, into an answer sentence selection dataset to provide the answers in their original context. We name it as *SQUAD-Sent*. We do this by processing the dataset where each example is usually a triple of Question, Paragraph and Answer span (Text and the answer start offset in the paragraph) into a dataset where each triple is a Question, Sentence and Sentence label. The sentence label is 1 if the answer is present inside the sentence, else it is 0. We perform sentence tokenization using spacy toolkit[6] on paragraphs of SQUAD and perform a check for an exact match of answer strings in them. *SQUAD-Sent* is a special case

[6] https://spacy.io

dataset where there is just one positive sentence per question and the other sentences are negative examples. The motivation to do this is because of the large scale property of this dataset, compared to the other datasets, with human-generated questions. For the expected answer types of SQUAD questions, we use SQUAD-EAT which is a dataset with EAT annotated questions on SQUAD v1 dataset questions which is annotated by the authors of [9, 15] on our request.

WikiQA dataset by [17] is another dataset used for answer sentence selection task which was built using Bing search engine query logs. We use a preprocessed version as used by [13] which has removed certain examples without any positive answers and questions with more than 40 tokens to compare the scores. The questions and answer sentences are annotated with EAT information as described in section 4.1.

Table 3 shows the statistics of the datasets with EAT annotated questions and plain word level questions (regular datasets) and the number of entities annotated in each set. EAT version of TrecQA dataset is as reported in [15] and available through this link[7].

| Dataset | Split | #Plain Q | #EAT Q | #Entities |
|---|---|---|---|---|
| Trec QA | Train | 1229 | 649 | 9064 |
| | Dev | 82 | 76 | 382 |
| | Test | 100 | 82 | 597 |
| SQUAD-Sent | Train | 87,599 | 78,740 | 35087 |
| | Dev | 10,570 | 9,606 | 4757 |
| | Test | - | - | - |
| Wiki QA | Train | 873 | 859 | 132 |
| | Dev | 126 | 124 | 4 |
| | Test | 243 | 236 | 38 |

**Table 3.** Statistics of datasets with plain and EAT annotated questions. '#' refers to "Number of."

### 4.4 Implementation

We implement the RNN-Similarity model in Pytorch, and we use MSELoss (Mean Squared Error loss) to minimize the error of predictions for relevance scores. We use adamax optimizer and keep the missing word embeddings as zero embeddings. We implement the EAT Classifier using the CNN model available online[8] and used Keras to implement the multiclass classifier which uses GloVe embeddings as input. The code for both the models along with default hyperparameters is publicly available on Github [9].

[7] www.harishmadabushi.com/research/answer-selection/
[8] https://github.com/yoonkim/CNN_sentence
[9] https://github.com/rsanjaykamath

### 4.5 Results

| Datasets | Method | Acc.@1 | MAP | MRR |
|---|---|---|---|---|
| TrecQA | Plain words - [13] | - | 78 | 83.4 |
| | EAT words - [15] | - | 83.6 | 86.2 |
| | Plain words - RNN-S | 78.95 | 80.24 | 84.81 |
| | EAT words (single type) - RNN-S | 85.26 | 85.28 | **89.16** |
| | EAT words (different types) - RNN-S | 85.26 | **85.48** | 88.11 |
| | EAT words (MAX+different types) - RNN-S | **86.32** | 85.42 | 88.86 |
| WikiQA | Plain words - [13] | - | 70.9 | 72.3 |
| | Plain words - [16] | - | **75.59** | **77.00** |
| | Plain words - RNN-S | 56.79 | 69.07 | 70.55 |
| | EAT words (single type) - RNN-S | 56.38 | 68.63 | 70.59 |
| | EAT words (different types) - RNN-S | 58.4 | **70.04** | **71.56** |
| | EAT words (MAX+different types) - RNN-S | 57.20 | 69.17 | 70.89 |
| SQUAD-Sent | Plain words - Implementation[10] of model by [13] | - | - | 58.08 |
| | Plain words - RNN-S | 83.94 | - | 90.5 |
| | EAT words (single type) - RNN-S | 84.21 | - | 90.65 |
| | EAT words (different types) - RNN-S | **84.26** | - | **90.70** |
| | EAT words (MAX+different types) - RNN-S | 84.24 | - | 90.69 |

**Table 4.** Results reported on TrecQA, WikiQA, and SQUAD-Sent datasets. SQUAD-Sent dataset is a modified version for answer sentence selection task. RNN-S is RNN-Similarity model.

Table 4 shows various results on different versions of datasets. Note that the questions in the following experiments of Table 4 contain all the questions from the datasets, which includes questions which are highlighted with EAT and questions which are not highlighted with EAT as well. Note that we test our systems on the Raw version of TrecQA test dataset.

**TrecQA** : The current state of the art system is by [15] that uses EAT on word level model of [13]. Henceforth both results are presented. Our model RNN-Similarity on plain word level data fetches better result than the model of [13] by 2.24 % on MAP and 1.41 % on MRR. Our EAT words (single type), EAT words (different types) and EAT words (MAX + different types) models outperforms the state of the art performance for both MAP (1.68%) and MRR (2.96%) scores of the previous state of the art model by [15] where the MAP and MRR scores are higher for correct sentences being ranked as top 1.

**WikiQA** : Although a recent model by [16] which uses kernel methods outperforms all the scores of our model, we note that the performance on our EAT level models is higher than the ones on plain words. Only a few number of entities are

annotated by spotlight compared to other datasets which is shown in the table 3. To annotate entities better we experimented using Spacy NER types directly which resulted in more annotated entities but reduced the performance lower than the word level scores.

**SQUAD-Sent** : SQUAD official test set is hidden to the public users. Although the difference between word level and EAT word level is little, the difference highlights the fact that the entity words replaced in the sentence would not worsen the performance of the systems; instead it improves it subtly. We would like to note that the MAP and MRR values were the same because of the existence of just 1 positive sentence amongst other negative per question. Hence we only report MRR on this dataset. Plain words - [13] performance is obtained using the implementation available online[11], which we experimented on SQUAD-Sent dataset.

One aspect to be highlighted is that the implementation[11] of word level model by [13] originally made for TrecQA dataset performs poorly (58.05%) SQUAD-Sent dataset (maybe because SQUAD-Sent has only one positive answer sentence per question whereas other datasets have several ones) which motivated us to build a model (RNN-Similarity) which works robustly for all the three datasets we have experimented with, without changing any specific hyperparameters of these models. Table 5 shows various results on TrecQA and SQUAD-Sent datasets with only the questions which are annotated with EAT information in the train and test sets.

Training datasets with questions which contain EAT information only; if the question does not have a EAT value, it is discarded from the dataset below are the set of experiments and results:

- TrecQA (EAT): Apart from EAT words (MAX + different types) version of the dataset, the other two methods outperform word level models and EAT word level by [15] where the dataset statistics of this method can also be found.
- SQUAD-Sent(EAT): There is a difference of 8,800 questions from SQUAD-Sent dataset, which is considerably a huge number of missing questions. Yet the results from these experiments, do not decrease a lot, but rather perform better than SQUAD-Sent's plain word level model compare to EAT (different types) data.
- WikiQA (EAT): We remove the questions with 'NO-EAT' class which were 23 questions overall. The results are better with EAT (single type) which shows that the method works well in certain cases better than different types of EAT.

The results reported in table 5 shows that there is not a significant improvement over different methods when trained only on questions with EAT information. Henceforth it is better to train models with the entire dataset and highlight EAT information only when the question contains the EAT information.

[11] https://github.com/castorini/Castor

| Datasets | Method | Acc.@1 | MAP | MRR |
|---|---|---|---|---|
| TrecQA (EAT) | EAT words (single type) | 84.15 | 84.81 | 87.17 |
| | EAT words (different types) | 85.37 | 85.45 | 88.18 |
| | EAT words (MAX+different types) | **85.37** | **85.06** | **89.20** |
| WikiQA (EAT) | EAT words (single type) | **58.02** | **68.91** | **70.99** |
| | EAT words (different types) | 55.14 | 67.70 | 69.52 |
| | EAT words (MAX+different types) | 56.38 | 68.16 | 69.83 |
| SQUAD-Sent (EAT) | EAT words (single type) | 83.81 | - | 90.53 |
| | EAT words (different types) | 84.04 | - | 90.61 |
| | EAT words (MAX+different types) | **84.16** | - | **90.73** |

**Table 5.** Results reported on TrecQA and SQUAD-Sent datasets using RNN-Similarity model trained only on EAT annotated questions

## 5 Conclusion and Future work

The Expected Answer Types are a useful piece of information that used to be extensively exploited in the traditional QA systems. Using them with the current state of the art DNN systems improves the system performance. We propose a simple model using recurrent neural networks which works robustly on three different datasets without any hyperparameter tuning and annotate entities belonging to the expected answer type of the question. Our model outperforms the previous state of the art systems in answer sentence task. We also propose a model to predict the expected answer type based on the question words using a multiclass classifier trained on a rule based system's output on a large scale QA dataset.

Future work involves using the expected answer types information in other downstream tasks such as in reading comprehension or multihop reading systems for extracting a short answer span.

### Acknowledgements

We would like to thank Harish Tayyar Madabushi from the University of Birmingham for providing us with the annotated questions of SQUAD dataset. This work is funded by the ANR project GoAsQ (ANR-15-CE23-0022).

## References

1. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading wikipedia to answer open-domain questions. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2017)
2. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems. ACM (2013)

3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015)
5. Kamath, S., Grau, B., Ma, Y.: Verification of the expected answer type for biomedical question answering. In: Companion Proceedings of the The Web Conference 2018. WWW '18 (2018)
6. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
7. Kolomiyets, O., Moens, M.F.: A survey on question answering technology from an information retrieval perspective. Information Sciences **181**(24) (2011)
8. Li, X., Roth, D.: Learning question classifiers: the role of semantic information. Natural Language Engineering **12**(3) (2006)
9. Madabushi, H.T., Lee, M.: High accuracy rule-based question classification using question syntax and semantics. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (2016)
10. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (2014)
11. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). vol. 1 (2018)
12. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (2016)
13. Rao, J., He, H., Lin, J.: Noise-contrastive estimation for answer selection with deep neural networks. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. CIKM '16 (2016)
14. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. ACM (2015)
15. Tayyar Madabushi, H., Lee, M., Barnden, J.: Integrating question classification and deep learning for improved answer selection. In: Proceedings of the 27th International Conference on Computational Linguistics. Association for Computational Linguistics (2018)
16. Tymoshenko, K., Moschitti, A.: Cross-pair text representations for answer sentence selection. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2018)
17. Yang, Y., Yih, W.t., Meek, C.: Wikiqa: A challenge dataset for open-domain question answering. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (2015)
18. Yin, W., Schütze, H., Xiang, B., Zhou, B.: Abcnn: Attention-based convolutional neural network for modeling sentence pairs. Transactions of the Association for Computational Linguistics **4** (2016)
19. Yu, A.W., Dohan, D., Le, Q., Luong, T., Zhao, R., Chen, K.: Fast and accurate reading comprehension by combining self-attention and convolution. In: International Conference on Learning Representations (2018)