Finding Prototypes of Answers for Improving Answer Sentence Selection

Wai Lok Tam[1*], **Namgi Han**[1,2]
Juan Ignacio Navarro-Horniacek[1,2,3], **Yusuke Miyao**[1,2]
[1] National Institute of Informatics, Japan
[2] The Graduate University for Advanced Studies, Japan
[3] Livesense Inc., Japan
w.tam@i-globalsociety.com, {namgi, juan, yusuke}@nii.ac.jp

Abstract

Answer sentence selection has been widely adopted recently for benchmarking techniques in Question Answering. Previous proposals for the task are essentially general solutions taking the form of neural networks that measure semantic similarity. In contrast, the present paper describes a simple technique to take advantage of such general-purpose tools for dealing with questions and answer sentences without changing the base system. The technique involves replacing wh-words in input questions with a word denoting the prototype of all answers. These transformed questions are passed as input to an existing neural network built for measuring semantic similarity. This technique is evaluated on two different neural network architectures over two datasets: TrecQA and WikiQA. Results of our experiments show improvement in overall accuracy across most question types we are interested in: 'who', 'when' and 'where'-type questions.

1 Introduction

Answer sentence selection roughly corresponds to the passage retrieval stage of the standard Question Answering (QA) pipeline. The upper part of figure 1 illustrates a typical approach found in previous literature. At the center it has a neural network which accepts a question and a candidate answer sentence as input and assigns one of the two classes, correct ("✓") and incorrect ("x") based on their semantic similarity score. The neural network is often based on Convolutional Neural Network (CNN) [Krizhevsky *et al.*, 2012] or Long Short-Term Memory Network (LSTM) [Hochreiter and Schmidhuber, 1997]. For this task, it is not necessary to provide a long answer to the question, rather a sentence containing a short answer is sufficient to make it correct answer, as illustrated by the lower part of figure 1. Recent research on the task carried out by [Yang *et al.*, 2016] and [Rao *et al.*, 2016] focuses on revising network architectures for computing better semantic similarity.

[*]The current affiliation of the first author: Institution for a Global Society, Japan.

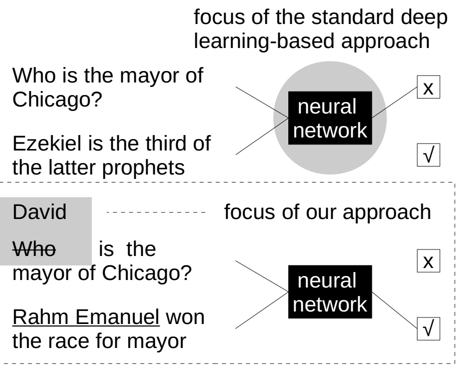
focus of the standard deep learning-based approach
Who is the mayor of Chicago?
Ezekiel is the third of the latter prophets
neural network
x
√
David
~~Who~~ is the mayor of Chicago?
focus of our approach
Rahm Emanuel won the race for mayor
neural network
x
√

Figure 1: The Standard Deep Learning-based Approach and Our Approach to Answer Sentence Selection

This paper presents a simple technique to boost the performance of the neural networks for answer sentence selection, without modifying the architecture of the base systems. In our proposed method, the wh-words in the input questions are replaced by a named entity (NE) that is found to be a prototype of the answers for the respective question type. Transformed questions and original candidate answer sentences are given as input to the neural networks proposed in previous literature for computing semantic similarity. As shown in the lower part of figure 1, our method replaces the wh-word in the input question ("*Who*") with another word ("*David*") selected from a set of named entities appropriate for who-type questions ("*Person class*"). In section 2 we introduce the motivation behind our method as well as the related works in answer sentence selection used in our experiments. In section 3, we explain the details of the two approaches to implement our method for selecting the prototype from a list of NEs provided by a named entity recognizer (NER).

The proposed technique is evaluated on two datasets created for benchmarking answer sentence selection: WikiQA [Yi Yang, 2015] and TrecQA [Wang *et al.*, 2007]. In section 4 we describe our experiment setup and its results over the two neural network architectures [Rao *et al.*, 2016; Tan *et al.*, 2015] that we consider to be the most representative of recent work. The results we obtained, described in

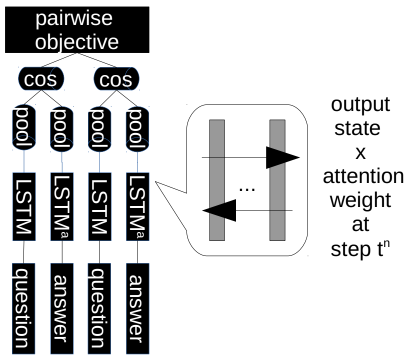

Figure 2: The Architecture of [Tan *et al.*, 2015]

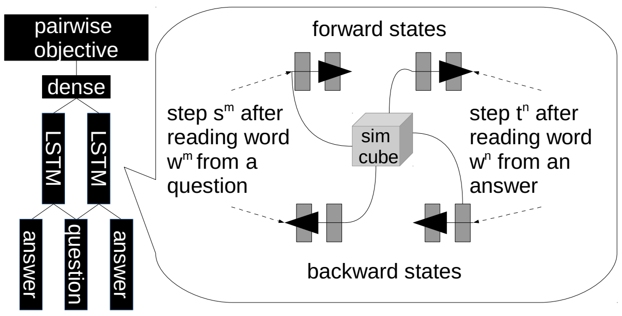
pairwise objective
dense
LSTM
LSTM
answer
question
answer
forward states
step s^m after reading word w^m from a question
sim cube
step t^n after reading word w^n from an answer
backward states

Figure 3: The Architecture of [Rao *et al.*, 2016]

section 4.2, reveal that our technique improves accuracy of both [Tan *et al.*, 2015] and [Rao *et al.*, 2016]. Possible ideas to extend this work are described in section 5.

2 Related Works and Motivation

Recent proposals for the answer sentence selection task include a wide range of combinations of similarity measures, different arrangement of layers of neural networks (LSTM vs CNN vs Fully Connected Layers) and various ways to pass the input to a neural network (passing a question and an answer sentence separately to two networks or organizing them into a matrix taken as input by a single network) [Rao *et al.*, 2016; Tan *et al.*, 2015; He and Lin, 2016; He *et al.*, 2015; Yang *et al.*, 2016; Feng *et al.*, 2015; Tan *et al.*, 2015; dos Santos *et al.*, 2016]. As it is described in section 4, we selected two of the most relevant proposals, as the base systems of our experimental setups.

The first one is [Tan *et al.*, 2015], which is illustrated in figure 2. The architecture proposed by them has two stacks, each made up by a bidirectional Long Short Term Memory network (bi-LSTM) and a pooling layer. The cosine similarity function is applied to the output of each stack, obtaining two similarity values. This is done by processing the input with these two stacks twice, once with the input question and a correct answer sentence and once again with the same question and an incorrect answer sentence. We illustrate this by duplicating the two stacks in figure 2, making it look like there are four stacks. The cosine similarity between the question and the correct answer sentence is compared with that between the question and the incorrect answer sentence by a pairwise objective function. The proposal is focused on a different way to compute the output state in the stack taking an answer sentence. Each output state $h_a(t)$ is applied a weight $s_{a,q}(t)$, whose computation, given in formula 1, takes into account the question embedding e_q.

$$s_{a,q}(t) \propto \exp(w_m^T \tanh(w_a h_a(t) + w_q e_q)))) \qquad (1)$$

The second proposal which we would look into is [Rao *et al.*, 2016]. The network architecture proposed in the paper comes in several variations. We illustrate one of these variations in figure 3. The architecture has two stacks. Each stack is a bi-LSTM with pairwise word interaction model described in [He and Lin, 2016] or a Multi-perspective CNN [He *et al.*, 2015]. The output of the two stacks are merged. The merged output is first passed to a fully connected layer ("dense" in figure 3) and then a pairwise objective function. The gist of the pairwise word interaction model is illustrated by the balloon. The output of the stack is a similarity cube ("sim cube" in figure 3). Values in this cube are similarity scores between the state in a pass (which can be either the forward pass or the backward pass) after accepting a word w^m from the question and the state in a pass after accepting a word w^n from the answer. The similarity cube compares the state in the forward pass after accepting a word in the question. Similarity is measured in terms of the cosine distance, the L2 Euclidean distance and the dot-product. The design delivers state-of-the-art results on TrecQA. When testing on WikiQA, some variations of the architecture also report results that come close to [He and Lin, 2016], which delivers state-of-the-art performance on WikiQA.

Although the systems detailed in these proposals run on QA datasets, they are designed for a more general purpose, measuring semantic similarity between a pair of sentences. Take the QA Matching Matrix with Value-shared Weights proposed by [Yang *et al.*, 2016] as an example. Their design comes with two justifications: First, the design allows terms in a question to interact directly with terms in an answer sentence, which is not possible with LSTM. Second, the design can handle non-local similarities, which CNN cannot capture. These two points justify the design being suitable not only for handling question-answer sentence pairs but also any other pair of sentences. This is because not only a question-answer sentence pair, but any other sentence pair, like a hypothesis and its premise in textual entailment, can have non-local similarities and be decomposed into parts which can be compared with each other. It is evident that [Yang *et al.*, 2016] is a general solution for measuring sentence similarity, just like several earlier designs framed as QA solution [Feng *et al.*, 2015; Tan *et al.*, 2015; dos Santos *et al.*, 2016] and many other neural networks built for the Semantic Text Similarity (STS) task and Recognizing Textual Entailment (RTE) task.

We think a dedicated solution like the one proposed in this paper has potential for making the general solutions work better on the specific task of answer sentence selection. Our assumption is that the capture of semantic similarity does not work with wh-words as well as with other words found in the answer sentences. In a word vector space model, the representation of a word is determined by its surrounding words. However, we assume that the words surrounding wh-words in the kind of documents used for training such models, e.g. Wikipedia, have very different distributions from the wh-words observed in question sentences. Documents used for training word vectors consist of an overwhelming majority of declarative sentences, and wh-words found in declarative sentences are mostly relative pronouns, which we suppose have totally different distribution than those observed in questions. Therefore their vectors are not easily related to any specific surrounding words, and thus getting a less defined position in the embedding space.

We look for hints on a dedicated solution from techniques addressing specific needs of the three stages of the standard QA pipeline: question processing, passage retrieval and answer processing. We find the idea of answer type detection [Li and Roth, 2005] inspiring. It is a technique applied at the first stage of the pipeline for taking care of wh-words and turning them into hints on the semantic classes of answers. These hints are reused at the final stage, answer processing, for ranking NEs, patterns, bigrams and sentences from passages retrieved in the second stage. Along the same line, our technique focuses on treatment of wh-words and aims at providing hints for answer sentence selection in the form of prototypes, words selected for replacing wh-words in questions before passing them to a neural network for computing semantic similarity.

The term "prototype" is borrowed from studies of categorization: how objects are put into natural categories by linguists and cognitive psychologists [Lakoff, 1987; Rosch, 1973]. Prototypes are considered as a representative of categories. Following this idea, our proposal is aimed to find a representative word vector for each category of answer words.

3 Replacing Wh-words with Prototypes

Replacing wh-words with prototypes of answers is the core part of our technique. Given a question and a candidate answer sentence as shown in the lower part of figure 1, we replace the crossed-out wh-word "*Who*" with the prototype of answers to who-type questions "*David*". The transformed question "David is the mayor of Chicago" is passed together with an unchanged candidate answer sentence to a neural network. From this point onwards, the pairs are processed in the same way as in the upper part of figure 1, which illustrates what previous proposals do.

In this work, we focus on three question types: who-type, when-type and where-type questions. Therefore, we require a prototype for each of the three types of questions. A shortlist of candidates for the prototype of answers to a question type are created with the Stanford NER using the 7-class model [Finkel *et al.*, 2005]. Out of the seven classes, namely, Location, Person, Organization, Money, Percent, Date and Time, three lists of NEs are created, one for the Person class, one for the Location class and one for the Time and Date class combined. Entities assigned the Person class are candidates

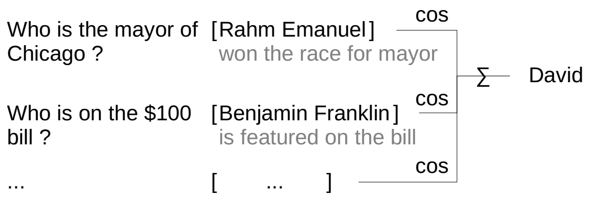

Figure 4: Summed Cosine Similarity between an NE and Short Answers

for the prototype of answers to who-type questions. Entities assigned the Location class are candidates for the prototype of answers to where-type questions. Entities assigned either the Time or Date class are candidates for the prototype of answers to when-type questions. It is obvious that entities in the Person class are possible answers to who-type questions, entities in the Location class are possible answers to where-type questions, and entities in the Time or Date class are possible answers to when-type questions. In contrast, we cannot find such clear connections between other NE classes and other type of questions. Therefore, we do not make any attempt to find prototypes of answers to the other question types. As a result, the question words in questions of these types are not replaced.

The prototype of answers to a question type is selected from a list of NEs by two methods. The first method is to select the entity with the maximum summed cosine similarity to the short answer of each question in the training set. We would call this method the "short answer-based method". Formally, given a NE list E, a set of short answers A, the prototype $\hat{e}$ obtained by this method is defined as:

$$\hat{e} = \operatorname*{arg\,max}_{e \in E} \sum_{a \in A} \cos(e, a) \tag{2}$$

A concrete example showing how summed cosine similarity is calculated for the NE "David" with this method is given in figure 4. First, we obtain the cosine similarity (cos) between "David" and the short answer to the first question in brackets "Rahm Emanuel". Next, we compute the cos between "David" and the short answer to the second question "Benjamin Franklin". The same computation is repeated for short answers to all other questions (omitted in figure 4 and represented by ...). Finally, we sum up the results for all NE-short answer pairs.

The second method is to select the entity yielding transformed questions with the maximum summed cosine similarity to the correct answer sentence of each question in the training set. We would call this method the "wh-word substitution-based method". Formally, given a NE list E, a set of answer sentences S, a set of questions Q, each containing the wh-word w, the prototype $\hat{e}$ obtained by this method is defined as:

$$\hat{e} = \operatorname*{arg\,max}_{e \in E} \sum_{s \in S, q \in Q} \cos(e \oplus (q \ominus w), s) \tag{3}$$

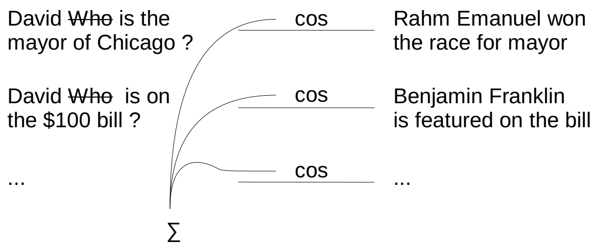
David ~~Who~~ is the mayor of Chicago ?
cos
Rahm Emanuel won the race for mayor
David ~~Who~~ is on the $100 bill ?
cos
Benjamin Franklin is featured on the bill
...
cos
...
Σ

Figure 5: Summed Cosine Similarity between Questions with Wh-words Replaced by an NE and Answer Sentences

where $\oplus$ is the concatenation operator and $\ominus$ is the operator for an operation that removes operand on the right from the beginning of the operand on the left.

A concrete example showing how summed cosine similarity is calculated for the NE ”David” with this method is given in figure 5. First, we replace the question word ”Who” in the first question with ”David” and obtain the cos between the transformed question ”David is the mayor of Chicago” and the correct answer sentence ”Rahm Emanuel won the race for mayor”. Next, we replace ”Who” in the second question with ”David” and compute the cos between the sentence resulted from the substitution ”David is on the $100 bill” and the correct answer sentence ”Benjamin Franklin is featured on the bill”. The substitution and computation are repeated for all other question-answer sentence pairs (omitted in figure 5 and represented by . . .). Finally, we sum up the results for all transformed question-answer sentence pairs.

The same method would deliver different results of the selection process if given a different list of named entities or if words denoting these entities are assigned different vectors by different word vector space models. For this paper, we use a single vector space model (Glove trained on Wikipedia [Pennington *et al.*, 2014]) but we would like to experiment with our technique on different datasets producing different lists of NEs.

Table 1 shows top-3 NEs selected as most prototypic of answers to each of the question types we are interested in by the two methods we have detailed so far. **R+S** stands for combining our technique with the short answer-based method. **R+W** stands for combining our technique with the wh-word substitution-based method. The former can only be used on WikiQA as TrecQA comes with no indication of short answers.

	WikiQA		TrecQA
Wh-word	**R+S**	**R+W**	**R+W**
who	David	William	Stenn
	George	Robert	Tonelli
	Michael	David	Friedan
when	1975	1923	1947-1956
	1949	1919	1929-1968
	1974	1946	1917-1963
where	Jamaica	Florida	Fussen
	Greece	Ohio	Mlada
	Europe	Virginia	94086

Table 1: Prototypes of Answers to Three Question Types

		WikiQA			TrecQA		
	type	train	dev	test	train	dev	test
split	who	119	15	34	190	11	8
	when	86	11	16	116	13	19
	where	71	17	22	96	9	11
full	target	276	43	72	402	33	38
	other	597	83	171	827	45	51

Table 2: Experimental Data by Question Type

4 Experiments

4.1 Experimental Setups

The purpose of our experiments is to evaluate the effectiveness of the proposed technique on the two neural networks introduced in section 2, over two standard datasets and in two ways of sampling: splitting the data by question type and processing all the data in a dataset as a whole.

The purpose of running experiments and control experiments on two base systems is for generalizing findings from results of our experiments across architectures. The first neural network chosen for our experiments is found in [Tan *et al.*, 2015]. The neural network proposed in the paper is selected for being a general design. Our experiments are run on a third party open source implementation of the design.[1] In this implementation, settings unmentioned in the original paper are determined by the developer and we follow these settings.

The second neural network on which we run our experiments is taken from [Rao *et al.*, 2016]. This neural network is selected for the state-of-the-art performance it delivers.

The actual implementation of the design of [Rao *et al.*, 2016] on which we run our experiments is done by the first author of the paper.[2] All default settings except for the batch size provided with the implementation are kept when running our experiments. For batch size, we use the commented-out value of 25 instead of 1.

The two networks accept data as it is or transformed from two datasets. The first dataset is TrecQA [Wang *et al.*, 2007], which is well established for benchmarking answer sentence selection. It is large but it does not provide any indication of the short answer to a question. Therefore, we cannot use it with the short answer-based method. The second dataset we use is WikiQA [Yi Yang, 2015]. It is smaller but it comes with short answers, making it a good test bed for our short answer-based method. We break down the content of each dataset by question types in table 2.

Out of each dataset, we create a *split* set and a *full* set. The split set consists in the three subsets containing only the who, when and where questions each. Each of these subsets is divided into training set, development set and a test set depending on where they come from in the original dataset.

From the split set, we create a control version of it, whose content is left unchanged. We also create (an) experimental version(s) of the split set, which come with question words

[1]https://github.com/codekansas/keras-language-modeling
[2]https://github.com/castorini/NCE-CNN-Torch

	WikiQA			TrecQA	
	Baseline	**R+S**	**R+W**	Baseline	**R+W**
who	0.685	**0.783**	0.692	0.629	**0.734**
when	0.681	0.674	**0.687**	0.811	**0.829**
where	0.692	0.766	**0.769**	0.894	**0.922**
target	0.686	**0.755**	0.714	0.798	**0.836**

Table 3: Results from Processing the Split Set with [Tan *et al.*, 2015]

	WikiQA			TrecQA	
	Baseline	**R+S**	**R+W**	Baseline	**R+W**
who	**0.708**	0.701	0.702	0.775	**0.781**
when	0.616	0.589	**0.664**	0.895	**0.921**
where	0.613	0.600	**0.616**	**0.894**	0.864
target	0.659	0.646	**0.667**	0.869	**0.875**

Table 4: Results from Processing the Split Set with [Rao *et al.*, 2016]

replaced by prototypes selected by the wh-word substitution-based method and the short answer-based method, if applicable.

From the full set, we also create a control version and an experimental version. The control version is just the dataset itself, without being split or changed. The experimental version is a mix of transformed questions and untransformed questions. The mix is created by replacing wh-words in who-type questions, when-type questions and where-type questions with prototypes selected by the wh-word substitution-based method and leaving other types of questions unchanged.

Once we obtain results from training one model per type with transformed data (the experimental version of the split set), we compare them with the results obtained by training different models with untransformed data (the control version of the split set) on the same base system over each question type. Experimenting with split data allows us to see whether our technique is effective for handling question types we are interested in.

Next, we experiment with mixed data (experimental data of the full set) and compare the results with those obtained from passing untransformed data as a whole (the control version of the full set) to the same network. This would enable us to see how far any effect identified in target question types would be diluted when they are mixed with non-targets. It is any impact left on the mixed data, with the same number and variety of questions as the control version of the full set used for evaluating previous proposals, which matters in telling whether our technique has an advantage over these proposals.

4.2 Results

The results we obtained on the experiment run over the split set on the [Tan *et al.*, 2015]-based system and the [Rao *et al.*, 2016]-based system are shown respectively in table 3 and 4. Columns labeled ”Baseline” give results in terms of Mean Reciprocal Rank (MRR) yielded from passing untransformed data type by type to the base systems. Columns labeled ”R+S” give results yielded from prototypes selected by the short answer-based technique. Columns labeled ”R+W”

	WikiQA		TrecQA	
	Baseline	**R+W**	Baseline	**R+W**
MRR	0.638	**0.660**	0.759	**0.829**
MAP	0.628	**0.652**	0.712	**0.756**

Table 5: Results from Processing the Full Set with [Tan *et al.*, 2015]

	WikiQA		TrecQA	
	Baseline	**R+W**	Baseline	**R+W**
MRR	0.665	**0.685**	0.849	**0.870**
MAP	0.650	**0.675**	0.751	**0.811**

Table 6: Results from Processing the Full Set with [Rao *et al.*, 2016]

give results yielded from prototypes selected by the wh-word substitution-based method. The last row gives the micro-averages of the MRR of the above 3 rows.

We can see that combining our technique with prototypes selected by the wh-word substitution-based method generally works better than the baseline (the bolded numbers in table 3 and table 4 indicates the best result for each base system on each dataset). The combination is also more stable than the combination of our technique with the short answer-based method. Row three to row five shows that it improves MRR in handling at least two of the three question types. The last row confirms that the combination of our technique with the wh-word substitution-based method generally works in showing that the micro-averaged MRRs of the combination exceed those of the baseline in all cases.

Results produced by experimenting with the full set on the [Tan *et al.*, 2015]-based system and the [Rao *et al.*, 2016]-based system are given in table 5 and 6 respectively. Columns labeled ”Baseline” show the results in terms of MRR and Mean Average Precision (MAP) from running the systems on the original data.

The results show that we still beat the baseline by a margin with the full dataset where only the ‘who’, ‘where’ and ‘when’-questions had their wh-words replaced by prototypes (the rest of the questions remain unchanged as in the original dataset).

5 Conclusions and Future Directions

In this paper, we proposed a pre-processing technique that improves the performance of the state-of-the-art neural networks for answer sentence selection task. The proposed technique leverages typological information from a NER and a word vector space model.

The technique can be summarized in the following three steps: First, we find a named entity with the maximum cosine similarity on all answer sentences for a question type. Next, we substitute the wh-words (we focused on only ‘where’, ‘when’ and ‘who’) with the named entity selected in the previous step, which we call a prototype. Finally, we provide the most probable answer based on the semantic similarity measure we found between the questions with its wh-word replaced and the answer sentences using the state-of-the-art neural networks. We have proven the effectiveness of our technique in improving performance to the answer sentence selection task over two datasets, on two state-of-the-art system architectures and in two ways of sampling.

For future works we propose two possible research directions: One direction can be, to use a better method to determine the question type more precisely (e.g. to support cases like "in what year" as when-type questions). The other direction can be to generalize the technique using tags as the prototype of the answer (like <PERSON>, <PLACE>, <YEAR>, etc), but this would require to train the tags embeddings (e.g.: duplicating the dataset to train the word embedding model but replacing the words in each answer category by its tag (e.g. 'David' $\rightarrow$ <PERSON>, 'London' $\rightarrow$ <PLACE>)) and thus it would lose the simplicity of the approach proposed in this paper, in which we are able to use the current general word vector model.

References

[dos Santos *et al.*, 2016] Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *CoRR*, abs/1602.03609, 2016.

[Feng *et al.*, 2015] Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. *CoRR*, abs/1508.01585, 2015.

[Finkel *et al.*, 2005] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[He and Lin, 2016] Hua He and Jimmy J. Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 937–948, 2016.

[He *et al.*, 2015] Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[Lakoff, 1987] George Lakoff. *Women, Fire and Dangerous Things*. University of Chicago Press, 1987.

[Li and Roth, 2005] X. Li and D. Roth. The role of semantic information. *Journal of Natural Language Engineering*, 2005.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[Rao *et al.*, 2016] Jinfeng Rao, Hua He, and Jimmy Lin. 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, pages 1913–1916, New York, NY, USA, 2016. ACM.

[Rosch, 1973] Eleanor Rosch. Natural Categories. In *Cognitive Psychology*, volume 4, pages 328–350. Kluwer, 1973.

[Tan *et al.*, 2015] Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108, 2015.

[Wang *et al.*, 2007] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Yang *et al.*, 2016] Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 287–296, New York, NY, USA, 2016. ACM.

[Yi Yang, 2015] Chris Meek Yi Yang, Scott Wen-tau Yih. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. ACL Association for Computational Linguistics, September 2015.