Injecting Relational Structural Representation in Neural Networks for Question Similarity

Antonio Uva[†] and **Daniele Bonadiman**[†] and **Alessandro Moschitti**
[†]DISI, University of Trento, 38123 Povo (TN), Italy
Amazon, Manhattan Beach, CA, USA, 90266
{antonio.uva,d.bonadiman}@unitn.it
amosch@amazon.com

Abstract

Effectively using full syntactic parsing information in Neural Networks (NNs) to solve relational tasks, e.g., question similarity, is still an open problem. In this paper, we propose to inject structural representations in NNs by (*i*) learning an SVM model using Tree Kernels (TKs) on relatively few pairs of questions (few thousands) as gold standard (GS) training data is typically scarce, (*ii*) predicting labels on a very large corpus of question pairs, and (*iii*) pre-training NNs on such large corpus. The results on Quora and SemEval question similarity datasets show that NNs trained with our approach can learn more accurate models, especially after fine tuning on GS.

1 Introduction

Recent years have seen an exponential growth and use of web forums, where users can exchange and find information just asking questions in natural language. Clearly, the possibility of reusing previously asked questions makes forums much more useful. Thus, many tasks have been proposed to build automatic systems for detecting duplicate questions. These were both organized in academia, e.g., SemEval (Nakov et al., 2016, 2017), or companies, e.g., Quora [1]. An interesting outcome of the SemEval challenge was that syntactic information is essential to achieve high accuracy in question reranking tasks. Indeed, the top-systems were built using Support Vector Machines (SVMs) trained with Tree Kernels (TKs), which were applied to a syntactic representation of question text (Filice et al., 2016, 2017; Barrón-Cedeño et al., 2016).

In contrast, NNs-based models struggled to get good accuracy as (*i*) large training sets are typically not available [2], and (*ii*) effectively exploiting full-syntactic parse information in NNs is still an open issue. Indeed, despite Das et al. (2016) showed that NNs are very effective to manage lexical variability, no neural model encoding syntactic information has shown a clear improvement. Indeed, also NNs directly exploiting syntactic information, such as the Recursive Neural Networks by Socher et al. (2013) or the Tree-LSTM by Tai et al. (2015), have been shown to be outperformed by well-trained sequential models (Li et al., 2015).

Finally, such tree-based approaches depend on sentence structure, thus are difficult to optimize and parallelize. This is a shame as NNs are very flexible in general and enable an easy system deployment in real applications, while TK models require syntactic parsing and longer testing time.

In this paper, we propose an approach that aims at injecting syntactic information in NNs, still keeping them simple. It consists of the following steps: (*i*) train a TK-based model on a few thousands training examples; (*ii*) apply such classifier to a much larger set of unlabeled training examples to generate automatic annotation; (*iii*) pre-train NNs on the automatic data; and (*iv*) fine-tune NNs on the smaller GS data.

Our experiments on two different datasets, i.e., Quora and Qatar Living (QL) from SemEval, show that (*i*) when NNs are pre-trained on the predicted data, they achieve accuracy higher than the one of TK models and (*ii*) NNs can be further boosted by fine-tuning them on the available GS data. This suggests that the TK properties are captured by NNs, which can exploit syntactic information even more effectively, thanks to their well-known generalization ability.

[1]https://www.kaggle.com/c/quora-question-pairs

[2]SQuAD by Rajpurkar et al. (2016) is an exception, also possible because dealing with a simpler factoid QA task

In contrast to other semi-supervised approaches, e.g., self-training, we show that the improvement of our approach is obtained only when a very different classifier, i.e., TK-based, is used to label a large portion of the data. Indeed, using the same NNs in a self-training fashion (or another NN in a co-training approach) to label the semi-supervised data does not provide any improvement. Similarly, when SVMs using standard similarity lexical features are applied to label data, no improvement is observed in NNs.

One evident consideration is the fact that TKs-based models mainly exploit syntactic information to classify data. Although, assessing that NNs specifically learn such syntax should require further investigation, our results show that only the transfer from TKs produces improvement: this is a significant evidence that makes it worth to further investigate the main claim of our paper. In any case, our approach increases the accuracy of NNs, when small datasets are available to learn high-level semantic task such as question similarity. It consists in (*i*) using heavier syntactic/semantic models, e.g., based on TKs, to produce training data; and (*ii*) exploit the latter to learn a neural model, which can then be fine-tuned on the small available GS data.

2 Tasks and Baseline Models

We introduce our question similarity tasks along with two of the most competitive models for their solutions.

2.1 Question Matching and Ranking

Question similarity in forums can be set in different ways, e.g., detecting if two questions are semantically similar or ranking a set of retrieved questions in terms of their similarity with the original question. We describe the two methods below:

The Quora task regards detecting if two questions are duplicate or not, or, in other words, if they have the same intent. The associated dataset (Wang et al., 2017) contains over $404,348$ pairs of questions, posted by users on the Quora website, labelled as duplicate pair or not. For example, *How do you start a bakery?* and *How can one start a bakery business?* are duplicated while *What are natural numbers?* and *What is a least natural number?* are not. The ground-truth labels contain some amount of noise.

In the QL task at SemEval-2016 (Nakov et al., 2016) users were provided with a new (original) question q_o and a set of related questions $(q_1, q_2, ...q_n)$ from the QL forum[3] retrieved by a search engine, i.e., Google. The goal is to rank question candidates, q_i, by their similarity with respect to q_o. q_i were manually annotated as *PerfectMatch*, *Relevant* or *Irrelevant*, depending on their similarity with q_o. *PerfectMatch* and *Relevant* are considered as relevant. A question is composed of a subject, a body and a unique identifier.

2.2 Support Vector machines

A top-performing model in the SemEval challenge is built with SVMs, which learn a classification function, $f : Q \times Q \rightarrow \{0, 1\}$, on the relevant vs. irrelevant questions belonging to the question set, Q. The classifier score is used to rerank a set of candidate questions q_i provided in the dataset with respect to an original question q_o. Three main representations were proposed: (*i*) vectors of similarity features derived between two questions; (*ii*) a TK function applied to the syntactic structure of question pairs; or (*iii*) a combination of both.

Feature Vectors (FV) are built for question pairs, (q_1, q_2), using a set of *text similarity* features that capture the relations between two questions. More specifically, we compute 20 similarities $sim(q_1, q_2)$ using word n-grams ($n = [1, \ldots, 4]$), after stopword removal, greedy string tiling (Wise, 1996), longest common subsequences (Allison and Dix, 1986), Jaccard coefficient (Jaccard, 1901), word containment (Lyon et al., 2001), and cosine similarity.

Tree Kernels (TKs) measure the similarity between the syntactic structures of two questions. Following (Filice et al., 2016), we build two macro-trees, one for each question in the pair, containing the syntactic trees of the sentences composing a question. In addition, we link two macro-trees by connecting the phrases, e.g., NP, VP, PP, etc., when there is a lexical match between the phrases of two questions. We apply the following kernel to two pairs of question trees: $K(\langle q_1, q_2\rangle, \langle q_1', q_2'\rangle) = TK(t(q_1, q_2), t(q_1', q_2')) + TK(t(q_2, q_1), t(q_2', q_1'))$, where $t(x, y)$ extracts the syntactic tree from the text x, enriching it with relational tags (REL) derived by matching the lexical between x and y.

[3]http://www.qatarliving.com/forum

3 Injecting Structures in NNs

We inject TK knowledge in two well-known and state-of-the-art networks for question similarity, enriching them with relational information.

3.1 NNs for question similarity

We implemented the Convolutional NN (CNN) model proposed by (Severyn and Moschitti, 2016). This learns f, using two separate sentence encoders $f_{q_1} : Q \to \mathbb{R}^n$ and $f_{q_2} : Q \to \mathbb{R}^n$, which map each question into a fixed size dense vector of dimension n. The resulting vectors are concatenated and passed to a Multi Layer Perceptron that performs the final classification. Each question is encoded into a fixed size vector using an embedding layer, a convolution operation and a global max pooling function. The embedding layer transforms the input question, i.e., a sequence of token, $X_q = [x_{q_1}, ..., x_{q_i}, ..., x_{q_n}]$, into a sentence matrix, $S_q \in R^{m \times n}$, by concatenating the word embeddings w_i corresponding to the tokens x_{q_i} in the input sentence.

Additionally, we implemented a Bidirectional (BiLSTM), using the standard LSTM by Hochreiter and Schmidhuber (1997). An LSTM iterates over the sentence one word at the time by creating a new word representation h_i by composing the representation of the previews word and the current word vector $h_i = LSTM(w_i, h_{i-1})$. A BiLSTM iterates over the sentence in both directions and the final representation is a concatenation of the hidden representations, h_N, obtained after processing the whole sentence. We apply two sentence models (with different weights), one for each question, then we concatenate the two fixed-size representations and fed them to a Multi-Layer Perceptron.

3.2 Relational Information

Severyn and Moschitti (2016) showed that relational information encoded in terms of overlapping words between two pairs of text can highly improve accuracy. Thus, for both networks above, we mark each word with a binary feature indicating if a word from a question appears in the other pair question. This feature is encoded with a fixed size vector (in the same way it is done for words).

3.3 Learning NNs with structure

To inject structured information in the network, we use a weak supervision technique: (*i*) an SVM with TK is trained on the GS data; (*ii*) this model classifies an additional unlabelled dataset, creating automatic data; and (*iii*) a neural network is trained on the latter data.

The pre-trained network can be fine-tuned on the GS data, using a smaller learning rate γ. This prevents catastrophic forgetting (Goodfellow et al., 2013), which may occur with a larger learning rate.

4 Experiments

We experiment with two datasets comparing models trained on gold and automatic data and their combination, before and after fine tuning.

4.1 Data

Quora dataset contains $384,358$ pairs in the training set and $10,000$ pairs both in the dev. and test sets. The latter two contain the same number of positive and negative examples.

QL dataset contains $3,869$ question pairs divided in $2,669$, 500 and 700 pairs in the training, dev. and test sets. We created 93k[4] unlabelled pairs from the QL dump, retrieving 10 candidates with Lucene for $9,300$ query questions.

4.2 NN setup

We pre-initialize our word embeddings with skip-gram embeddings of dimensionality 50 jointly trained on the English Wikipedia dump (Mikolov et al., 2013) and the jacana corpus[5]. The input sentences are encoded with fixed-sized vectors using a CNN with the following parameters: a window of size 5, an output of 100 dimensions, followed by a global max pooling. We use a single non-linear hidden layer, whose size is equal to the size of the sentence embeddings, i.e., 100. The word overlap embeddings is set to 5 dimensions. The activation function for both convolution and hidden layers is ReLU. During training the model optimizes the binary cross-entropy loss. We used SGD with Adam update rule, setting the learning rate to γ to 10^{-4} and 10^{-5} for the pre-training and fine tuning phases, respectively.

4.3 Results on Quora

Table 1 reports our different models, FV, TK, CNN and LSTM described in the previous section, where the suffix, -10k or -5k, indicates the amount of GS data used to train them, and the name in

[4]Note that we will release the 400k automatically labelled pairs from Quora as well as the new 93k pairs of QL along with their automatic labels for research purposes.

[5]Embeddings are available in the repository: `https://github.com/aseveryn/deep-qa`

Model	Automatic data	GS data	DEV	TEST
FV-10k	–	10k	0.7046	0.7023
TK-10k	–	10k	0.7405	0.7337
CNN-10k	–	10k	0.7646	0.7569
LSTM-10k	–	10k	0.7521	0.7450
CNN(CNN-10k)	50k	–	0.7666	0.7619
CNN(CNN-10k)*	50k	10k	0.7601	0.7598
CNN(FV-10k)	50k	–	0.6960	0.6931
CNN(FV-10k)*	50k	10k	0.7681	0.7565
CNN(TK-10k)	50k	–	0.7446	0.7370
CNN(TK-10k)*	50k	10k	0.7748	0.7652
LSTM(TK-10k)	50k	–	0.7478	0.7371
LSTM(TK-10k)*	50k	10k	0.7706	0.7505
TK-5k	–	5k	0.6859	0.6774
CNN-5k	–	5k	0.7532	0.7450
CNN(TK-5k)	50k	–	0.7239	0.7208
CNN(TK-5k)*	50k	5k	0.7574	0.7493
CNN(TK-10k)	375k	–	0.7524	0.7471
CNN(TK-10k)*	375k	10k	**0.7796**	**0.7728**
Voting(TK+CNN)	–	10k	0.7838	0.7792

Table 1: Accuracy on the Quora dataset.

parenthesis indicates the model used for generating automatic data, e.g., CNN(TK-10k) means that a CNN has been pre-trained with the data labelled by a TK model trained on 10k GS data. The amount of automatic data for pre-training is in the second column, while the amount of GS data for training or fine tuning (indicated by *) is in the third column. Finally, the results on the dev. and test sets are in the fourth and fifth columns.

We note that: first, NNs trained on 10k of GS data obtain higher accuracy than FV and TK on both dev. and test sets (see the first four lines);

Second, CNNs pre-trained with the data generated by FV or in a self-training setting, i.e., CNN(CNN-10k), and also fine-tuned do not improve[6] on the baseline model, i.e., CNN-10K, (see the second part of the table).

Third, when CNNs and LSTMs are trained on the data labelled by the TK model, match the TK model accuracy (third part of the table). Most importantly, when they are fine-tuned on GS data, they obtain better results than the original models trained on the same amount of data, e.g., 1% accuracy over CNN-10k.

Next, the fourth part of the table shows that the improvement given by our method is still present when training TK (and fine tuning the NNs) on less GS data, i.e., only 5k.

Additionally, the fifth section of the table shows a high improvement by training NNs on all available Quora data annotated by TK-10k (trained on just 10k). This suggests that NNs require more data to learn complex relational syntactic patterns expressed by TKs. However, the plot in Figure 1 shows that the improvement reaches a plateau around 100k examples.

Finally, in the last row of the table, we report the result of a voting approach using a combination of the normalized scores of TK-10k and CNN-10k. The accuracy is almost the same than CNN(TK-10k)*. This shows that NNs completely learn the combination of a TK model, mainly exploiting syntax, and a CNN, only using lexical information. Note that the voting model is heavy to deploy as it uses syntactic parsing and the kernel algorithm, which has a time complexity quadratic in the number of support vectors.

4.4 Results on Qatar Living

Table 2 reports the results when applying our technique to a smaller and different dataset such as QL. Here, CNNs have lower performance than TK models as 2,669 pairs are not enough to train their parameters, and the text is also noisy, i.e., there are a lot of spelling errors. Despite this problem, the results show that CNNs can approximate the

[6]The improvement of 0.5 is not statistically significant.

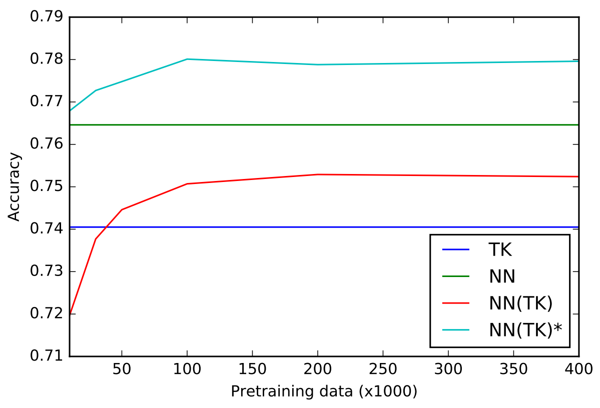

Figure 1: Impact of the pre-training data.

TK models well, when using a large set of automatic data. For example, the CNN trained on 93k automatically annotated examples and then fine tuned exhibits 0.4% accuracy improvement on the dev. set and almost 3% on the test set over TK models. On the other hand, using too much automatically labeled data may hurt the performance on the test set. This may be due to the fact the quality of information contained in the gold labeled data deteriorates. In other words, using the right amount of weekly-supervision is an important hyper-parameter that needs to be carefully chosen.

5 Related Work

Determining question similarity is one of the main challenges in building systems that answer real user questions (Agichtein et al., 2015, 2016) in community QA, thus different approaches have been proposed. Jeon et al. (2005) used a language model based on word translation table to compute the probability of generating a query question, given a target/related question. Zhou et al. (2011) showed the effectiveness of phrase-based translation models on Yahoo! Answers. Cao et al. (2009); Duan et al. (2008) proposed a similarity between two questions based on a language model that exploits the category structure of Yahoo! Answers. Wang et al. (2009) proposed a model to find semantically related questions by computing similarity between syntactic trees representing questions. Ji et al. (2012) and Zhang et al. (2014) used latent semantic topics that generate question/answer pairs.

Regarding the use of automatically labelled data, Blum and Mitchell (1998) applied semi-supervised approaches, such as self-training and co-training to non-neural models. The main point

Model	Automatic Data	Dev	Test
CNN		0.7000	0.7514
TK		0.7340	0.7686
CNN(TK)	50k	0.5580	0.5428
CNN(TK)*	50k	0.7160	**0.7814**
CNN(TK)	93k	0.7000	0.6957
CNN(TK)*	93k	**0.7380**	0.7614

Table 2: Accuracy on QL using all available GS data.

of our paper is the use standard weakly-supervised methods to inject syntactic information in NNs.

Hu et al. (2016) tried to combine symbolic representations with NNs by transferring structured information of logic rules into the weights of NNs. Our work is rather different as we inject syntactic, and not logic, information in NNs.

The work most similar to our is the one by Croce et al. (2017), who use Nystrom methods to compact the TK representation in embedding vectors and use the latter to train a feed forward NNs. In contrast, we present a simpler approach, where NNs learn syntactic properties directly from data.

To our knowledge, ours is the first work trying to use NNs to learn structural information from data labelled by TK-based models. Finally, no systems of the SemEval challenges used NNs trained on syntactic information.

6 Conclusion

In this work, we have trained TK-based models, which make use of structural information, on relatively small data and applied them to new data to produce a much larger automatically labeled dataset. Our experiments show that NNs trained on the automatic data improve their accuracy. We may speculate that NNs learn relational structural information as (*i*) TK models mainly use syntactic structures to label data and (*ii*) other advanced models based on similarity feature vectors do not produce any improvement. Indeed, the latter only exploit lexical similarity measures, which are typically also generated by NNs. However, even if our conjecture were wrong, the bottom line would be that, thanks to our approach, we can have NN models comparable to TK-based approaches, by also avoiding to use syntactic parsing and expensive TK processing at deployment time.

References

Eugene Agichtein, David Carmel, Dan Pelleg, Yuval Pinter, and Donna Harman. 2015. Overview of the TREC 2015 LiveQA Track. In *TREC*.

Eugene Agichtein, David Carmel, Dan Pelleg, Yuval Pinter, and Donna K. Harman. 2016. Overview of the TREC 2016 LiveQA Track. In *TREC*.

Lloyd Allison and Trevor Dix. 1986. A bit-string longest-common-subsequence algorithm. *Information Processing Letters*, 23(6):305–310.

Alberto Barrón-Cedeño, Daniele Bonadiman, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad A Al Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and question selection for question answering on Arabic and English fora. *Proceedings of SemEval*, pages 896–903.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Xin Cao, Gao Cong, Bin Cui, Christian Søndergaard Jensen, and Ce Zhang. 2009. The use of categorization information in language models for question retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 265–274. ACM.

Danilo Croce, Simone Filice, Giuseppe Castellucci, and Roberto Basili. 2017. Deep learning in semantic kernel spaces. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 345–354.

Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 378–387, Berlin, Germany. Association for Computational Linguistics.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *Proceedings of ACL-08: HLT*, pages 156–164, Columbus, Ohio. Association for Computational Linguistics.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123. Association for Computational Linguistics.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2017. KeLP at SemEval-2017 Task 3: Learning Pairwise Patterns in Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 326–333. Association for Computational Linguistics.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420. Association for Computational Linguistics.

Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90. ACM.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2471–2474. ACM.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

Caroline Lyon, James Malcolm, and Bob Dickerson. 2001. Detecting short passages of similar text in large document collections. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 118–125, Pittsburgh, PA, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48. Association for Computational Linguistics.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *arXiv preprint arXiv:1604.01178*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based QA services. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 187–194. ACM.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Michael J. Wise. 1996. YAP3: Improved detection of similarities in computer program and other texts. In *ACM SIGCSE Bulletin*, volume 28, pages 130–134. ACM.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 371–380. ACM.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 653–662. Association for Computational Linguistics.