KeLP at SemEval-2017 Task 3: Learning Pairwise Patterns in Community Question Answering

Simone Filice[1], **Giovanni Da San Martino**[2] and **Alessandro Moschitti**[2]
[1] DICII, University of Roma, Tor Vergata
[2] ALT, Qatar Computing Research Institute, HBKU
filice.simone@gmail.com
{gmartino,amoschitti}@hbku.edu.qa

Abstract

This paper describes the KeLP system participating in the SemEval-2017 community Question Answering (cQA) task. The system is a refinement of the kernel-based sentence pair modeling we proposed for the previous year challenge. It is implemented within the Kernel-based Learning Platform called KeLP, from which we inherit the team's name. Our primary submission ranked first in subtask A, and third in subtasks B and C, being the only systems appearing in the top-3 ranking for all the English subtasks. This shows that the proposed framework, which has minor variations among the three subtasks, is extremely flexible and effective in tackling learning tasks defined on sentence pairs.

1 Introduction

This paper describes the KeLP system participating in the SemEval-2017 cQA challenge (Nakov et al., 2017). The task setting for the English part is the same as the previous edition (Nakov et al., 2016): the corpus is extracted from *Qatar Living*[1], a web forum where people pose questions about multiple aspects of their daily life in Qatar, and three subtasks are defined:

Subtask A: Given a question q and its first 10 comments $c_1, \ldots, c_{10}$ in its question thread, re-rank these 10 comments according to their relevance with respect to the question, i.e., the *good* comments have to be ranked above *potential* or *bad* comments.

Subtask B: Given a new question o and the set of the first 10 related questions $q_1, \ldots, q_{10}$ (retrieved by a search engine), re-rank the related questions according to their similarity with respect to o, i.e., the *perfect match* and *relevant* questions should be ranked above the *irrelevant* ones.

Subtask C: Given a new question o, and the set of the first 10 related questions, $q_1, \ldots, q_{10}$, (retrieved by a search engine), each one associated with its first 10 comments, $c^q_1, \ldots, c^q_{10}$, appearing in its thread, re-rank the 100 comments according to their relevance with respect to o, i.e., the *good* comments are to be ranked above *potential* or *bad* comments.

We participated to the previous year edition, where our system (Filice et al., 2016) achieved very good results, i.e., first in subtask A, third in B and second in C. For the new year challenge, we therefore decided to reuse the same system applied to a new method for selecting tree structures, (Barrón-Cedeño et al., 2016; Romeo et al., 2016) summarized in Sec. 3.

We modeled the three subtasks as binary classification problems: kernel-based classifiers are trained and the classification score is used to sort the instances and produce the final ranking. We implemented models within the Kernel-based Learning Platform[2] (KeLP) (Filice et al., 2015a), which determined the team's name. Our tests provide two main contributions: (*i*) we asses the results obtained in (Filice et al., 2016), demonstrating that our kernel-based models for relational learning tasks between two texts (Filice et al., 2015b) are effective for community Question Answering. (*ii*) We studied the impact of text selection described in (Barrón-Cedeño et al., 2016).

Our primary submission ranked first in subtask A, and third in subtasks B and C, demonstrating that the proposed method is very accurate and adaptable to different learning problems. At the moment, we could not find out if text selection is always useful as our contrastive submission not

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

[2] http://www.kelp-ml.org/

using it turned out to be much more accurate for Task B.

In the reminder, Section 2 introduces the proposed kernel-based system, Section 3 describes the pruning technique to select the relevant parts from the input sentences, while Section 4 reports official results.

2 The KeLP system: kernel-based learning from text pairs

In the three subtasks, the underlying problem is to understand if two texts are related. Thus, in subtasks A and C, each pair, (question, comment), generates a training instance for a binary Support Vector Machine (SVM) (Chang and Lin, 2011), where the positive label is associated with a *good* comment and the negative label includes the *potential* and *bad* comments. In subtask B, we evaluated the similarity between two questions. Each pair generates a training instance for SVM, where the positive label is associated with the *perfect match* or *relevant* classes and the negative label is associated with the *irrelevant*; the resulting classification score is used to rank the question pairs.

In KeLP, the SVM learning algorithm operates on a kernel combination of tree kernels and a linear kernel. In particular the linear kernel is applied on feature vectors containing (*i*) linguistic similarities between the texts in a pair (Section 2.1); (*ii*) task-specific features (Section 2.3).

Tree kernels are applied to evaluate inter-pair similarities between sentence pairs, in order to automatically discover pairwise relational patterns.

2.1 Intra-pair similarities

In subtasks A and C, a *good* comment is likely to share similar terms with the question. In subtask B a question that is relevant to another probably shows common words. Following this intuition, given a text pair (either question/comment or question/question), we define a feature vector whose dimensions reflect the following similarity metrics:

- **Lexical Similarities:** *Cosine similarity*, *Jaccard coefficient* (Jaccard, 1901) and *containment measure* (Broder, 1997) of *n*-grams of word lemmas ($n = 1, 2, 3, 4$ was used in all experiments); *Longest common substring measure* (Gusfield, 1997), *Longest common subsequence measure* (Allison and Dix, 1986), and *Greedy String Tiling* (Wise, 1996).

- **Syntactic Similarities:** *Cosine similarity* of *n*-grams of part-of-speech tags. It considers a shallow syntactic similarity ($n = 1, 2, 3, 4$ was used in all experiments); *Partial tree kernel* (Moschitti, 2006) between the parse tree of the sentences.

- **Semantic Similarities:** *Cosine similarity* between additive representations of word embeddings generated by applying word2vec (Mikolov et al., 2013) to the entire Qatar Living corpus from SemEval 2015[3]. Five features are derived considering (*i*) only nouns, (*ii*) only adjectives, (*iii*) only verbs, (*iv*) only adverbs and (*v*) all the above words.

These metrics are computed in all the subtasks between the two elements within a pair, i.e., q and c_i for subtask A, q and o for subtask B, o and c_i for subtask C. In addition, in subtasks B and C, the similarity metrics (except the Partial Tree Kernel similarity) are computed between o and the entire thread of q, concatenating q with its answers. Similarities between q and o are also employed in subtask C.

2.2 Inter-pair kernel methods

In tasks A and C, some question types may have an expected answering form. Similarly, in Task B, related questions may be characterized by the application of some latent paraphrasing rules. Such pairwise patterns cannot be captured by any intra-pair similarity feature, and require an alternative approach. Specific features may be manually defined, but this would require a complex feature engineering.

To automatize relational learning between pairs of texts, one of the early works is (Moschitti et al., 2007; Moschitti, 2008). This approach was improved in several subsequent researches (Severyn and Moschitti, 2012; Severyn et al., 2013a,b; Severyn and Moschitti, 2013; Tymoshenko et al., 2014; Tymoshenko and Moschitti, 2015), exploiting relational tags and linked open data. In particular, in (Filice et al., 2015b), we defined new inter-pair methods to directly employ text pairs into a kernel-based learning framework.

The kernels we proposed can be directly applied to subtask B and to subtasks A and C for learn-

[3] http://alt.qcri.org/semeval2015/task3

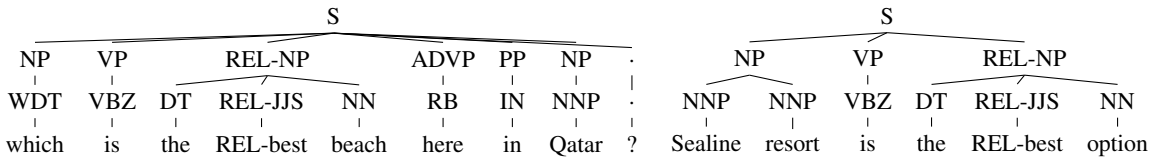

Figure 1: Structural Representation of a question-answer pair.

ing question-question and question-answer pairwise patterns (see also (Tymoshenko et al., 2016; Da San Martino et al., 2016). As shown in Figure 1, a pair of sentences is represented as pair of their corresponding shallow parse trees, where common or semantically similar lexical nodes are linked using a tagging strategy (which is propagated to their upper constituents). This method discriminates aligned sub-fragments from non-aligned ones, allowing the learning algorithm to capture relational patterns, e.g., *the* REL-*best beach* and *the* REL-*best option*. Thus, given two pairs of sentences $p_a = \langle a_1, a_2 \rangle$ and $p_b = \langle b_1, b_2 \rangle$, some tree kernel combinations can be defined:

$$\text{PTK}^{+}(p_a, p_b) = \text{PTK}(a_1, b_1) + \text{PTK}(a_2, b_2)$$
$$\text{PTK}^{\times}(p_a, p_b) = \text{PTK}(a_1, b_1) \times \text{PTK}(a_2, b_2)$$

where PTK is the Partial Tree Kernel (PTK) (Moschitti, 2006). Tree kernels, computing the shared substructures between parse trees, are effective in evaluating the syntactic similarity between two texts. The proposed tree kernel combinations extend such reasoning to text pairs.

2.3 Task Specific Features

While the features described so far can be effectively applied to any sentence pair modeling task, in this section, we describe features specifically developed for the cQA domain.

- **Ranking Features:** The ten questions related to an original question are retrieved using a search engine. We use their absolute and relative ranks[4] as features for subtasks B and C (for the latter the question rank is given to all the comments within the related question thread).

- **Heuristics:** We adopt the heuristic features described in (Barrón-Cedeño et al., 2015), which can be applied to subtasks A and C. In particular, forty-five features capture some comment characteristics such as its length, its category (*Socializing*, *Life in Qatar*, etc.), whether it includes URLs, emails, or other particular words, etc.

- **Thread-based features:** As discussed in (Barrón-Cedeño et al., 2015), comments in a common thread are strongly interconnected: users replicate to each others and start a concrete discussion. We used some specific features for subtasks A and C that aim at capturing some thread-level dependencies, such as whether a comment is part of a dialogue or whether a comment is followed by an acknowledgment from the user who asked the question

- **Stacking features:** A *good* comment for a question q should be also *good* for an original question o if q and o are strongly related, i.e., q is *relevant* or a *perfect match* to o. We thus developed a stacking strategy for Subtask C that uses the following scores in the classification step, w.r.t. an original question o and the comment c_i from the thread of q:
  - p_{q,c_i}, which is the score of the pair $\langle q, c_i \rangle$ provided by the model trained on Subtask A;
  - p_{o,c_i}, which is the score of the pair $\langle o, c_i \rangle$ provided by the model trained on Subtask A;
  - $p_{o,q}$, which is the score of the pair $\langle o, q \rangle$ provided by the model trained on Subtask B.

  Starting from these scores, we built the following features: (*i*) values and signs of p_{q,c_i}, p_{o,c_i} and $p_{o,q}$ (6 feats); (*ii*) a boolean feature indicating whether both p_{q,c_i} and $p_{o,q}$ are positive; (*iii*) *min value* = $min(p_{q,c_i}, p_{o,q})$; (*iv*) *max value* = $max(p_{q,c_i}, p_{o,q})$; (*v*) *average value* = $\frac{1}{2}(p_{q,c_i} + p_{o,q})$.

3 Tree Pruning Techniques

We propose to reduce the size of the input trees by removing all nodes and branches that are less

[4]Some of the results retrieved by the search engine were filtered out, because they were threads with less than 10 comments, or documents out of Qatar Living. Therefore, the threads in the dataset may have an associated rank higher than 10. The relative rank maps such absolute values into [1;10].

discriminative for the task. To determine such fragments, we use the supervised approach described in (Barrón-Cedeño et al., 2016). After training a tree kernel, $K()$, on pairs of trees, the solution of the dual optimization problem is expressed as a linear combination of a subset of the training examples, i.e., the support vectors: $M = \{(\alpha_j, (a_j, b_j))\}$, where the $(a_j, b_j) \in A \times B$ is a pair of parse trees (a_j could be the one of an original or related question and b_j the one of a related question or a comment, depending on the subtask) and α_j are the coefficients of the combination. The classification of a new example is obtained as the sign of the score function $f()$:

$$f(a, b) = \sum_{1 \leq j \leq |M|} \alpha_j K((a, b), (a_j, b_j)), \quad (1)$$

where $|M|$ is the number of support vectors, i.e., the number of elements of the set M. The higher the absolute value of the score of an example, the more confident the learning algorithm is in classifying it. We exploit such property to devise a strategy for determineing the importance $w(n)$ of a node. Let n be a node of a tree t, $\overset{n}{\triangle}$ is the proper sub-tree rooted at n, i.e., the tree composed of n and all its descendants in t. We use the score of $\overset{n}{\triangle}$ with respect to M to assess the importance of n:

$$w(n) = \begin{cases} \sum_{1 \leq j \leq |M|} \alpha_j PTK(\overset{n}{\triangle}, a_j) & \text{if } n \in a, a \in A \\ \sum_{1 \leq j \leq |M|} \alpha_j PTK(\overset{n}{\triangle}, b_j) & \text{if } n \in b, b \in B. \end{cases} \quad (2)$$

In order to be consistent, only the parse trees of $a_j \in A$ will be used to compute $w(n)$, if n belongs to the first tree of the pair $(a_j, b_j) \in M$. Conversely if n belongs to the second tree of the pair (a_j, b_j) only the parse trees of $b_i \in B$ will be used.

Now we can proceed to prune a tree on the basis of the $w(n)$ importance estimated by model M for each of its nodes and a user-defined threshold. We prune a leaf node n if $-h < w(n) < h$. If n is not a leaf, then it is removed if all its children are going to be removed. Note that the threshold h determines the number of pruned nodes. Our algorithm has a constraint: REL-tagged nodes are never pruned, regardless of their estimated importance. This is because a REL tag indicates that a and b share a common leaf in $\overset{n}{\triangle}$, which conveys useful information, e.g., for paraphrasing (Filice et al., 2015b).

		MAP	AvgR	MRR	P	R	F_1	Acc
2016	IR	59.53	72.60	67.83	-	-	-	-
	KeLP	**79.19**	88.82	86.42	76.96	55.30	64.36	75.11
2017	IR	72.61	79.32	82.37	-	-	-	-
	KeLP	**88.43**	93.79	92.82	87.30	58.24	69.87	73.89

Table 1: Results on subtask A on the 2016 and 2017 official testsets. IR is the baseline system based on the search engine results.

4 Submission and Results

We chose parameters using the 2016 official test set as validation set, and we trained on the official train and development sets[5]. In Subtask C, the stacking features (Section 2.3) need the scores provided by the models on subtasks A and B. Such scores are generated with a 10-fold cross validation. For the final submissions we used all the 2016 data (including the testset) as training. We used the OpenNLP pipeline for lemmatization, POS tagging and chunking to generate the tree representations described in Section 2.2. All the kernel-based learning models are implemented in KeLP (Filice et al., 2015a). For all the tasks, we used the C-SVM learning algorithm (Chang and Lin, 2011). The MAP@10 was the official metric. In addition, results are also reported in Average Recall (AvgR), Mean Reciprocal Rank (MRR), Precision (P), Recall (R), F_1, and Accuracy (Acc).

4.1 Subtask A

Model: The learning model operates on question-comment pairs $p = \langle q, c \rangle$. The kernel is $\text{PTK}^+(p_a, p_b) + \text{LK}_A(p_a, p_b)$. Such kernel linearly combines $\text{PTK}^+(p_a, p_b) = \text{PTK}(q_1, q_2) + \text{PTK}(c_1, c_2)$ (see Section 2.2) with a linear kernel LK_A that operates on feature vectors including: (*i*) the similarity metrics between q and c described in Section 2.1; (*ii*) the heuristic features and (*iii*) the thread-based features discussed in Section 2.3. PTK uses the default parameters (Moschitti, 2006), while the best SVM regularization parameter we estimated was $C = 1$. This system is identical to the one we proposed in the previous year.

Results: Table 1 reports the results on subtask A. We confirmed the excellent results of 2016: the model is very accurate and achieved the first position among 13 systems in terms of MAP.

[5] We merged the official Train1, Train2 and Dev sets.

		MAP	AvgR	MRR	P	R	F_1	Acc
2016	IR	74.75	88.30	83.79	-	-	-	-
	KeLP	**78.50**	91.95	84.52	71.30	70.39	70.84	80.71
	KC1	75.47	90.68	82.48	70.42	72.53	71.46	80.71
2017	IR	41.85	77.59	46.42	-	-	-	-
	KeLP	46.66	81.36	50.85	36.01	85.28	50.64	69.20
	KC1	**49.00**	83.92	52.41	36.18	88.34	51.34	68.98

Table 2: Results on subtask B on the 2016 and 2017 official test sets. KeLP is our primary submission, while KC1 is the contrastive one. IR is the baseline system based on the search engine results.

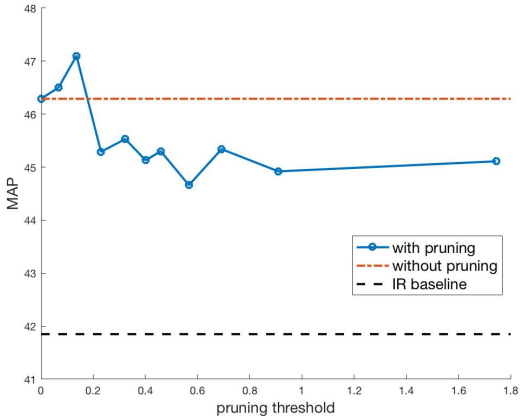

Figure 2: 10 fold cross validation results on the official 2017 test set with different pruning thresholds on subtask B.

4.2 Subtask B

Model: The proposed system operates on question-question pairs $p = \langle o, q \rangle$. The kernel is $\text{PTK}^{\times}(p_a, p_b) + \text{LK}_B(p_a, p_b)$, by adopting the kernels defined in Section 2.2. The product in the $\text{PTK}^{\times}$ combination acts like a *logic and*, as, when comparing two pairs, we want a strict match in which both the elements of the first pair must be similar to the counterpart elements in the second pair. Conversely, in subtasks A and C, the adopted $\text{PTK}^{+}(p_a, p_b)$ applies a sort of *logic or* as we noticed that some form of comments may be considered *good* (or *bad*) regardless the question they are answering. We pruned the question trees according to the criterion described in Section 3. The best pruning threshold we estimated on the 2016 test set was $h = 0.91$. The previous year model adopted the Smoothed Partial Tree Kernel (SPTK) (Croce et al., 2011) in place of the PTK. This year we decided to use the PTK kernel as our preliminary experiments did not justified the usage of the slower SPTK.

LK_B is a linear kernel that operates on feature vectors including: (*i*) the similarity metrics between o and q, and between o and the entire answer thread of q, as described in Section 2.1; (*ii*) ranking features, described in Section 2.3. With respect to the previous year challenge we did not include some features derived from subtask A, because in subsequent experiments they did not demonstrate a significant impact.

The best SVM regularization parameter estimated during the tuning stage is $C = 1$.

We made an additional submission in which the pruning in not applied.

Results: Table 2 shows the results on subtask B. On the official test set, our primary submission achieved the third position w.r.t. MAP among 13 systems. Differently from what observed in the tuning stage, on the official test set the contrastive system achieves the highest MAP and would have ranked first in the challenge.

In general, the difference between the system accuracy obtained in 2016 and 2017 suggests that the two test sets are rather different. To verify this hypothesis, we performed a 10-fold cross validation using only the data from 2017 test set. We kept the same pruning strategy and weights computed on the 2016 training set that we applied to the entire test set of 2017 for our official submission. We evaluated different pruning thresholds. Figure 2 reports the MAP averaged over the results of a 10 fold cross validation on the official 2017 test set (the 2016 dataset is not used at all).

The results show that (*i*) our best system with or without pruning is less accurate than the submitted results, i.e., producing an MAP of 46.29: this is reasonable since the model uses less training data. (*ii*) our pruning can improve our best system from 46.29 to 47.10 MAP.

Thus, it would seem that the difference between 2016 and 2017 dataset plays an important role for the pruning approach as removing some subtrees makes the TK approach more effective but probably also more specific to the data used for training the model. Another possible explanation is that it is easier to improve a weaker model, using less data. Finding out the properties of tree pruning is surely an interesting research line we would like to pursue in the future.

		MAP	AvgR	MRR	P	R	F_1	Acc
2016	IR	40.36	45.97	45.83	-	-	-	-
	KeLP	**55.91**	59.57	60.99	52.16	22.17	31.12	90.83
2017	IR	9.18	21.72	10.11	-	-	-	-
	KeLP	**14.35**	30.74	16.07	6.48	89.02	12.07	63.75

Table 3: Results on subtask C on the 2016 and 2017 official test sets. KeLP is our primary submission, while IR is the baseline system based on the search engine results.

4.3 Subtask C

Model: The learning model operates on the triplet, $\langle o, q, c \rangle$, using the kernel, $\text{PTK}^+(p_a, p_b) + \text{LK}_C(t_a, t_b)$, where $\text{PTK}^+(p_a, p_b) = \text{PTK}(o_1, o_2) + \text{PTK}(c_1, c_2)$ (see Section 2.2) and LK_C is a linear kernel operating on feature vectors, which include: (*i*) the similarity metrics between o and c, between o and q, and between o and the entire thread of q, as described in Section 2.1; (*ii*) the heuristic features, (*iii*) the thread-based features, (*iv*) the ranking features, and (*v*) the features derived from the scores of subtasks A and B, described in Section 2.3. PTK uses the default parameters. The subtask training data is rather imbalanced, as the number of negative examples is about 10 times the positive ones. We took this into account by setting the regularization parameter for the positive class, $C_p = \frac{\#\text{negatives}}{\#\text{positives}} C$, as in (Morik et al., 1999). The best SVM regularization parameter estimated during the tuning stage is $C = 5$. The system is identical to the one proposed the previous year.

Results: Table 3 shows the results for subtask C. Our primary submission achieved the third highest MAP among 5 systems. The large difference among the 2016 and 2017 MAP is mainly due to the much lower presence of relevant examples in the 2017 test set, indeed, more than 97% of instances are *irrelevant*.

Acknowledgements

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action).

References

Lloyd Allison and Trevor I. Dix. 1986. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.* 23(6):305–310. http://dl.acm.org/citation.cfm?id=8871.8877.

Alberto Barrón-Cedeño, Simone Filice, Giovanni Da San Martino, Shafiq Joty, Lluís Màrquez, Preslav Nakov, and Alessandro Moschitti. 2015. Thread-level information for comment classification in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China, pages 687–693. http://www.aclweb.org/anthology/P15-2113.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Salvatore Romeo, and Alessandro Moschitti. 2016. Selecting sentences versus selecting tree constituents for automatic question ranking. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2515–2525.

A. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*. IEEE Computer Society, Washington, DC, USA, SEQUENCES '97, pages 21–. http://dl.acm.org/citation.cfm?id=829502.830043.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:27:1–27:27.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 1034–1046. http://dl.acm.org/citation.cfm?id=2145432.2145544.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Antonio Uva, and Alessandro Moschitti. 2016. Learning to re-rank questions in community question answering using advanced features. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. pages 1997–2000. https://doi.org/10.1145/2983323.2983893.

Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. 2015a. KeLP: a Kernel-based Learning Platform in java. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*. International Conference of Machine Learning, Lille, France.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 task 3: Learning semantic relations between questions and comments. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. San Diego, California, SemEval '16.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015b. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1003–1013. http://www.aclweb.org/anthology/P15-1097.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

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* 37:547–579.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Katharina Morik, Peter Brockhausen, and Thorsten Joachims. 1999. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *ICML*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 268–277.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*. Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Proceedings, Berlin, Germany, pages 318–329.

Alessandro Moschitti. 2008. Kernel Methods, Syntax and Semantics for Relational Text Categorization. In *Proceeding of ACM 17th Conf. on Information and Knowledge Management (CIKM'08)*. Napa Valley, CA, USA.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 776–783. http://www.aclweb.org/anthology/P07-1098.

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*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

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*. Association for Computational Linguistics, San Diego, California, SemEval '16.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeño, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James R. Glass. 2016. Neural attention for learning to rank questions in community question answering. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 1734–1745.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '12, pages 741–750. https://doi.org/10.1145/2348283.2348383.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*. ACL, pages 458–467.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Building structures from classifiers for passage reranking. In *Proceedings of the 22nd ACM international Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '13, pages 969–978. https://doi.org/10.1145/2505515.2505688.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning adaptable patterns for passage reranking. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 75–83. http://www.aclweb.org/anthology/W13-3509.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Learning to rank non-factoid answers: Comment selection in web forums. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. pages 2049–2052. https://doi.org/10.1145/2983323.2983906.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '15, pages 1451–1460. https://doi.org/10.1145/2806416.2806490.

Kateryna Tymoshenko, Alessandro Moschitti, and Aliaksei Severyn. 2014. *Encoding semantic resources*

in syntactic structures for passage reranking, Association for Computational Linguistics (ACL), pages 664–672.

Michael J. Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *Proceedings of the Twenty-seventh SIGCSE Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, SIGCSE '96, pages 130–134. https://doi.org/10.1145/236452.236525.