KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers

Simone Filice[1], **Danilo Croce**[2],
Alessandro Moschitti[3] and **Roberto Basili**[2]
[1] DICII, University of Roma, Tor Vergata
[2] DII, University of Roma, Tor Vergata
[3] ALT, Qatar Computing Research Institute, HBKU
{filice,croce,basili}@info.uniroma2.it
amoschitti@qf.org.qa

Abstract

This paper describes the KeLP system participating in the SemEval-2016 Community Question Answering (cQA) task. The challenge tasks are modeled as binary classification problems: kernel-based classifiers are trained on the SemEval datasets and their scores are used to sort the instances and produce the final ranking. All classifiers and kernels have been implemented within the Kernel-based Learning Platform called KeLP. Our primary submission ranked first in Subtask A, third in Subtask B and second in Subtask C. These ranks are based on MAP, which is the referring challenge system score. Our approach outperforms all the other systems with respect to all the other challenge metrics.

1 Introduction

This paper describes the KeLP system participating in the SemEval-2016 cQA challenge. In this task, participants are asked to automatically provide good answers in a cQA setting (Nakov et al., 2016). In particular, the main task is: given a *new question* and a large collection of *question-comment threads* created by a user community, rank the most useful comments on the top.

We participated in all English subtasks: the datasets were extracted from *Qatar Living*[1], a web forum where people pose questions about multiple aspects of their daily life in Qatar. Three subtasks are associated with the English challenge:

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, $c_1^q, \ldots, c_{10}^q$, 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.

All the above subtasks have been modeled as binary classification problems: kernel-based classifiers are trained and the classification score is used to sort the instances and produce the final ranking. All classifiers and kernels have been implemented within the Kernel-based Learning Platform[2] (KeLP) (Filice et al., 2015b), thus determining the team's name. The proposed solution provides three main contributions: (*i*) we employ the approach proposed in (Severyn and Moschitti, 2012), which applies tree kernels directly to question and answer texts modeled as pairs of linked syntactic trees. We further improve the methods using the kernels proposed in (Filice et al., 2015c). (*ii*) we extended the features developed in (Barrón-Cedeño et al., 2015), by adopting several features (also derived from Word Embeddings (Mikolov et al., 2013)). (*iii*) we propose

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

[2] https://github.com/SAG-KeLP

Proceedings of SemEval-2016, pages 1116–1123,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

a stacking schema so that classifiers for Subtask B and C exploit the inferences obtained in the previous subtasks.

Our primary submission ranked first in Subtask A, third in Subtask B and second in Subtask C, demonstrating that the proposed method is very accurate and adaptable to different learning problems. These ranks are based on the MAP metric. However, if we consider the other metrics also adopted in the challenge (e.g., F_1 or Accuracy) our approach outperforms all the remaining systems.

In the remaining, Section 2 introduces the system, Sections 3 and 4 describe the feature and kernel modeling, while Section 5 reports official results.

2 The KeLP system: an overview

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 linear combination of kernel functions, each one applied over a specific representation of the targeted examples: (*i*) feature vectors containing linguistic similarities between the texts in a pair; (*ii*) shallow syntactic trees that encode the lexical and morpho-syntactic information shared between text pairs; (*iii*) feature vectors capturing task-specific information; (*iv*) in subtasks B and C, feature vectors encoding stacked information derived by applying the classifiers obtained in the previous subtasks.

While (*i*) and (*ii*) use linguistic information that can be applied in any semantic processing task defined over text pairs (see Sec. 3), the information derived via (*iii*) and (*iv*) is task specific (see Sec. 4).

3 Learning from Text Pairs with Kernels

The problem of deciding whether two questions are related or whether a comment answers a question, can be somehow connected to the problems of recognizing textual entailment, and paraphrase identification. From a machine learning perspective, in these tasks, an example is a pair of texts, instead of a single entity. Conventional approaches convert input text pairs into feature vectors where each feature represents a score corresponding to a certain type of shared information or similarity between the elements within a pair. These intra-pair similarity approaches cannot capture complex relational pattern between the elements in the pair, such as a rewriting rule characterizing a valid paraphrase, or a question-answer pattern. Such information might be manually encoded into specific features, but it would require a complex feature engineering and a deep knowledge of the linguistic involved phenomena.

To automatize relational learning between pairs of texts, e.g., in case of QA, 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; Severyn et al., 2013b; 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., 2015c), we propose new inter-pair methods to directly employ text pairs into a kernel-based learning framework. In the proposed approach, we integrate the information derived from simple intra-pair similarity functions (Section 3.1) and from the structural analogies (Section 3.2).

3.1 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:

- *Cosine similarity*, *Jaccard coefficient* (Jaccard, 1901) and *containment measure* (Broder, 1997) of *n*-grams of word lemmas. It captures lexical information and word ordering information ($n = 1, 2, 3, 4$ was used in all experiments).
- *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).

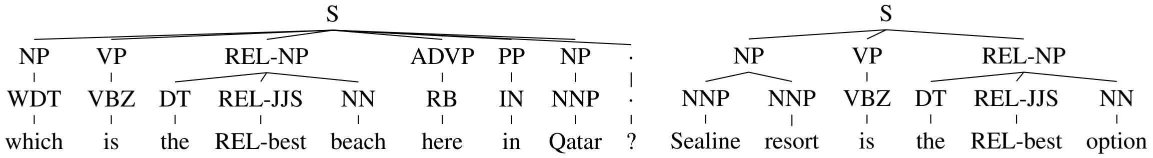

Figure 1: Structural Representation of a question-answer pair.

- *Partial tree kernel* (Moschitti, 2006) between the parse tree of the sentences. It performs a deep syntactic comparison.
- *Longest common substring measure* (Gusfield, 1997) determines the length of the longest contiguous sequence of characters shared by two text segments.
- *Longest common subsequence measure* (Allison and Dix, 1986) drops the contiguity requirement of the previous measure and allows to detect similarity in case of word insertions/deletions.
- *Greedy String Tiling* (Wise, 1996) provides a similarity between two sentences by counting the number of shuffles in their subparts.
- *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]. We derived 250 dimensional vectors for 37,000 words by applying the settings min-count=50, window=5, iter=10 and negative=10. 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.

3.2 Inter-pair kernel methods

The kernels we proposed in (Filice et al., 2015c) can be directly applied to Subtask B and to subtasks A and C for learning question/question and question/answer similarities, respectively. 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:

$$\begin{aligned}\mathrm{TK}^{+}(p_a, p_b) = &\quad \mathrm{TK}(a_1, b_1) + \mathrm{TK}(a_2, b_2)\\ \mathit{All}^{\times}_{\mathrm{TK}}(p_a, p_b) = &\quad \mathrm{TK}(a_1, b_1) \times \mathrm{TK}(a_2, b_2)\\ + &\quad \mathrm{TK}(a_1, b_2) \times \mathrm{TK}(a_2, b_1),\end{aligned}$$

where TK is a generic tree kernel, such as the Partial Tree Kernel (PTK) (Moschitti, 2006), or the Smoothed Partial Tree Kernel (SPTK) (Croce et al., 2011). 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, and can capture emerging pairwise patterns. Therefore this method can be effective in recognizing valid question/answer pairs, or similar questions, even in those cases in which the two texts have few words in common that would cause the failure of any intra-pair approach.

4 Task Specific Features

In this section, we describe features specifically developed for cQA. A single feature vector is generated for each of the following group of features.

4.1 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

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

[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].

and C (for the latter the question rank is given to all the comments within the related question thread).

4.2 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-four boolean features express whether a comment: (*i*) includes URLs or emails (2 feats.). (*ii*) contains the word "yes", "sure", "no", "can", "neither", "okay", and "sorry", as well as symbols '?' and '@' (9 feats.); (*iii*) starts with "yes" (1 feat.); (*iv*) includes a sequence of three or more repeated characters or a word longer than fifteen characters (2 feats.); (*v*) belongs to one of the categories of the forum (*Socializing*, *Life in Qatar*, etc.) (26 feats.); (*vi*) has been posted by the same user who posted the question, such a comment can include a question (i.e., it contains a question mark), and acknowledgment (e.g., it contains *thank**, *acknowl**), or none of them (4 feats.);

An additional feature captures the length of the comment (as longer —*good*— comments usually contain detailed information to answer a question).

4.3 Thread-based features

In cQA, comments in a thread typically reflect an underlying concrete discussion, which contains more information than in a sequence of independent answers retreived from different documents. For instance, users replicate to each other, ask for further details, or can tease other users. Therefore, as discussed in (Barrón-Cedeño et al., 2015), comments in a common thread are strongly interconnected. To exploit such thread-level dependencies, we used some specific features for subtasks A and C. The following notation will be adopted: q is the question posted by user u_q, c is a comment from user u_c, in the comment thread.

The first four features indicate whether c appears in the proximity of a comment by u_q. The assumption is that an acknowledgment or further questions by u_q in the thread could signal a *good* answer. More specifically, they test if among the comments following c there is one by u_q (*i*) containing an acknowledgment, (*ii*) not containing an acknowledgment, (*iii*) containing a question, and, (*iv*) if among the comments preceding c there is one by u_q containing a question. These four features depend on the distance k, in terms of the number of comments, between c and the closest comment by u_q:

$$f(c) = \begin{cases} 1.1 - 0.1k \\ 0 \text{ if no comments by } u_q \text{ exist,} \end{cases}$$

that is, the closer the comment to u_q, the higher the value assigned to this feature. Other features try to model potential dialogues, which at the end represent *bad* comments, by identifying interlacing comments between two users. These dialogue features are identifying conversation chains:

$$u_i \rightarrow \ldots \rightarrow u_j \rightarrow \ldots \rightarrow u_i \rightarrow \ldots \rightarrow [u_j]$$

Comments by other users can appear in between the nodes of this "pseudo-conversation" chain. Three features consider whether a comment is at the beginning, in the middle, or at the end of such a chain. Three more features are defined when $u_j = u_q$, i.e., the user who asked the question is one of the participants of these pseudo-conversations.

Another interesting aspect is whether a user u_i has been particularly active in a question thread. One boolean feature captures whether u_i wrote more than one comment in the current thread. Three more features identify the first, the middle and the last comments by u_i. One extra feature counts the total number of comments written by u_i. Moreover, it can be empirically observed that the likelihood of a comment being *good* decreases with its position in the thread. Therefore, another real-valued feature was included: $i/10$, where i represents the position of the comment in the thread.

4.4 Stacking classifiers across subtasks

The three subtasks are interconnected: the predictions from a subtask can provide useful information to carry out the other subtasks. This suggests the use of a stacking strategy.

Stacking classifiers in Subtask B. If the comments in the question thread of q are *good* answers for an original question o, we can suppose that o and q are strongly related. In Subtask B, we thus exploit the model trained on Subtask A. In particular, given the original question o, and the related question q with its comments, $c_1 \ldots c_n$, we use the model from Subtask A to classify the question/comment pairs, $\langle q, c_i \rangle$ and $\langle o, c_i \rangle$, obtaining respectively the scores p_{q,c_i} and p_{o,c_i}. We consider these scores as distributions and derive the following features: (*i*) *mean squared* *error* (MSE) = $\sum_i (p_{q,c_i} - p_{o,c_i})^2$; (*ii*) *Pearson correlation coefficient* between the $p_{q,c_1}, \ldots, p_{q,c_n}$ and the $p_{o,c_1}, \ldots, p_{o,c_n}$; (*iii*) ten features corresponding to the sorted differences between p_{q,c_i} and p_{o,c_i}; (*iv*) *agreement percentage*, i.e., percentage of times $sign(p_{q,c_i}) = sign(p_{o,c_i})$; (*v*) *max score* = $\max_i(p_{o,c_i})$; (*vi*) *mean score* = $\frac{1}{n}\sum_i p_{o,c_i}$; (*vii*) *positive percentage*, i.e., percentage of times $p_{o,c_i} > 0$; (*viii*) *normalized positive percentage*, i.e., percentage of times $p_{o,c_i} > 0$ when $p_{q,c_i} > 0$.

Stacking classifiers in Subtask C. 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* = $\frac{1}{2}(p_{q,c_i} + p_{o,q})$.

5 Submission and Results

We chose parameters using a 10 fold cross validation (cv) on the official train and development sets[5]. In Subtask B, some features depend on the scores provided on Subtask A, while in Subtask C the dependency is from both Subtasks A and B. Such scores are generated with the 10-fold cv. We used the OpenNLP pipeline for lemmatization, POS tagging and chunking to generate the tree representations described in Section 3.2. All the kernel-based learning models are implemented in KeLP (Filice et al., 2015b). 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).

		MAP	AvgR	MRR	P	R	F_1	Acc
CV	IR	57.70	72.75	66.82	-	-	-	-
	KeLP	**74.76**	88.24	80.90	69.64	61.51	65.32	76.02
test	IR	59.53	72.60	67.83	-	-	-	-
	KeLP	**79.19**	88.82	86.42	76.96	55.30	64.36	75.11

Table 1: Results on Subtask A on a 10 fold CV on the training and development and on the official test set. IR is the baseline system based on the search engine results

5.1 Subtask A

Model: The learning model operates on question-comment pairs $p = \langle q, c \rangle$. The kernel is PTK$^+(p_a, p_b)$ + LK$_A(p_a, p_b)$. Such kernel linearly combines PTK$^+(p_a, p_b)$ = PTK(q_1, q_2) + PTK(c_1, c_2) (see Section 3.2) with a linear kernel LK$_A$ that operates on feature vectors including: (*i*) the similarity metrics between q and c described in Section 3.1; (*ii*) the heuristic features introduced in Section 4.2; (*iii*) the thread-based features discussed in Section 4.3. PTK uses the default parameters (Moschitti, 2006), while the best SVM regularization parameter we estimated during cv is $C = 1$.

Results: Table 1 reports the outcome on Subtask A. The good results on the 10 fold cross validations are confirmed on the official test set: the model is very accurate and achieved the first position among 12 systems, with the best MAP. In this task the data distribution among classes is quite balanced and the accuracy is also a good performance indicator. In this case we achieved the second position.

5.2 Subtask B

Model: The proposed system operates on question-question pairs $p = \langle o, q \rangle$. The kernel is $All^{\times}_{\text{SPTK}}(p_a, p_b)$ + LK$_B(p_a, p_b)$, by adopting the kernels defined in Section 3.2. This task is close to Paraphrase Identification, which is inherently symmetric. Therefore, in our primary submission, we adopted a tree kernel combination that exploits such characteristic, performing cross comparisons between the questions within pairs: $All^{\times}_{\text{SPTK}}(p_a, p_b)$ = SPTK(o_1, o_2) × SPTK(q_1, q_2) + SPTK(o_1, q_2) × SPTK(q_1, o_2). Such combination is based on the SPTK with standard parameters and a word similarity derived from the word embeddings described in Section 3.1. LK$_B$ is a linear kernel that oper-

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

		MAP	AvgR	MRR	P	R	F_1	Acc
CV	IR	66.27	83.14	73.87	-	-	-	-
	KeLP	**70.37**	87.50	77.44	71.47	75.71	**73.53**	**77.69**
	KC1	69.97	87.22	76.71	70.19	75.87	72.92	76.93
	KC2	70.06	87.26	76.92	68.96	75.02	71.86	75.95
test	IR	74.75	88.30	83.79	-	-	-	-
	KeLP	75.83	91.02	82.71	66.79	75.97	**71.08**	**79.43**
	KC1	**76.28**	91.33	82.71	63.83	77.25	69.90	77.86
	KC2	76.27	91.44	84.10	64.06	77.25	70.04	78.00

Table 2: Results on Subtask B on a 10 fold Cross-Validation (CV) and on the official test set. KeLP is our primary submission, while KC1 and KC2 are the contrastive ones. IR is the baseline system based on the search engine results

		MAP	AvgR	MRR	P	R	F_1	Acc
dev	IR	30.65	34.55	35.97	-	-	-	-
	KeLP	**38.57**	44.09	41.28	21.55	64.64	32.32	81.32
	KC1	38.00	43.74	41.18	21.97	64.64	32.79	81.72
	KC2	37.22	42.63	42.80	22.30	71.30	33.98	80.88
test	IR	40.36	45.97	45.83	-	-	-	-
	KeLP	52.95	59.27	59.23	33.63	64.53	44.21	84.79
	KC1	52.95	59.34	58.06	34.08	65.29	**44.78**	**84.96**
	KC2	**55.58**	63.36	61.19	32.21	70.18	44.16	83.41

Table 3: Results on Subtask C on the official development set, and on the official test set. KeLP is our primary submission, while KC1 and KC2 are the contrastive ones. IR is the baseline system based on the search engine results

ates 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 3.1; (*ii*) ranking features discussed in Section 4.1; (*iii*) the features derived from the Subtask A scores (see Section 4.4). The best SVM regularization parameter estimated during the tuning stage is $C = 5$.

We made two additional submissions in which the model has minor variations: in the Contrastive 1 (KC1), we substituted $All^{\times}_{\text{SPTK}}$ with SPTK^{+} whereas in the contrastive 2 (KC2) we do not include the features derived from the Subtask A scores.

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 11 systems. Differently from what observed in the tuning stage, on the official test set the contrastive systems achieve a higher MAP and would have ranked second. The primary system achieves the highest F_1 and accuracy on both tuning and test stages. Considering these two metrics, our primary submission is overall the best model.

5.3 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 3.2) 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 3.1; (*ii*) the heuristic features introduced in Section 4.2; (*iii*) the thread-based features discussed in 4.3; (*iv*) the ranking features (see Section 4.1); (*v*) the features derived from the scores of subtasks A and B, described in Section 4.4. PTK uses the default parameters. The subtask 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 $C = 5$. We also submitted a Contrastive 1 (KC1) with PTK^{+} using $All^{\times}_{\text{PTK}}$ and a Constrastive 2 (KC2) identical as KC1 but with C set to 2.

Results: Table 3 shows the results for Subtask C. The organizers reported that the training labels were affected by noise, while the development labels were double-checked. Therefore, we decided to perform parameter tuning applying cv to the development set only. Our primary submission achieved the second highest MAP, while our Contrastive 2 is the best result. It should be also noted that the F_1 our system is the best among 10 primary submissions. In this subtask, accuracy is not a reliable measure, as the data is significantly imbalanced.

In a future work we would like to change the learning paradigm from classification, e.g., demonstrated in (Filice et al., 2015a) for several NLP applications, to a learning to rank problem. This can be enabled by the preference kernel (Severyn and Moschitti, 2012) and should have a positive impact on the MAP metric since the SVM classification algorithm we used optimizes accuracy.

Acknowledgements

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action), the PROGRESS-IT project (FILAS-CR-2011-1089) and by an IBM Faculty Award.

References

Lloyd Allison and Trevor I. Dix. 1986. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.*, 23(6):305–310, December.

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)*, pages 687–693, Beijing, China, July.

A. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA. IEEE Computer Society.

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*, EMNLP '11, pages 1034–1046, Stroudsburg, PA, USA. Association for Computational Linguistics.

Simone Filice, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2015a. Kelp: a kernel-based learning platform for natural language processing. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 19–24, Beijing, China, July. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

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

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015c. 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)*, pages 1003–1013, Beijing, China, July.

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.

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*, pages 268–277, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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*, pages 776–783, Prague, Czech Republic, June. Association for Computational Linguistics.

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

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.

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 '16, San Diego, California, June. Association for Computational Linguistics.

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*, SIGIR '12, pages 741–750, New York, NY, USA. ACM.

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

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*, CIKM '13, pages 969–978, New York, NY, USA. ACM.

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*, pages 75–83, Sofia, Bulgaria, August. Association for Computational Linguistics.

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*, CIKM '15, pages 1451–1460, New York, NY, USA. ACM.

Kateryna Tymoshenko, Alessandro Moschitti, and Aliaksei Severyn, 2014. *Encoding semantic resources in syntactic structures for passage reranking*, pages 664–672. Association for Computational Linguistics (ACL), 1.

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*, SIGCSE '96, pages 130–134, New York, NY, USA. ACM.