Information Processing and Management 56 (2019) 274–290

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Language processing and learning models for community question answering in Arabic

Salvatore Romeo[a,*], Giovanni Da San Martino[a], Yonatan Belinkov[b], Alberto Barrón-Cedeño[a], Mohamed Eldesouki[a], Kareem Darwish[a], Hamdy Mubarak[a], James Glass[b], Alessandro Moschitti[a]

[a] *Qatar Computing Research Institute, HBKU, Doha, Qatar*
[b] *MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA*

A R T I C L E   I N F O

Article history:
Received 1 December 2016
Revised 4 May 2017
Accepted 15 July 2017
Available online 9 August 2017

Keywords:
Community question answering
Constituency parsing in Arabic
Tree-kernel-based ranking
Long short-term memory neural networks
Attention models

A B S T R A C T

In this paper we focus on the problem of question ranking in community question answering (cQA) forums in Arabic. We address the task with machine learning algorithms using advanced Arabic text representations. The latter are obtained by applying tree kernels to constituency parse trees combined with textual similarities, including word embeddings. Our two main contributions are: (*i*) an Arabic language processing pipeline based on UIMA—from segmentation to constituency parsing—built on top of Farasa, a state-of-the-art Arabic language processing toolkit; and (*ii*) the application of long short-term memory neural networks to identify the best text fragments in questions to be used in our tree-kernel-based ranker. Our thorough experimentation on a recently released cQA dataset shows that the Arabic linguistic processing provided by Farasa produces strong results and that neural networks combined with tree kernels further boost the performance in terms of both efficiency and accuracy. Our approach also enables an implicit comparison between different processing pipelines as our tests on Farasa and Stanford parsers demonstrate.

1. Introduction

Community-driven question answering (cQA) on the web typically refers to popular forums in which users ask and answer questions on diverse topics. The freedom to post practically any question and answer in virtual anonymity promotes massive participation. The large amount of posts resulting from this environment demands the implementation of automatic models to filter relevant from irrelevant contents. This scenario has received attention from researchers in both the natural language processing and the information retrieval areas. However, for several reasons, languages other than English—including Arabic—have received relatively less attention.

In this research, we focus on the problem of improving the retrieval of questions from an Arabic forum with respect to a new user question. Our task is formally defined as follows. Let q be a new user question and D the set of question–answer pairs, previously posted in a forum. Rank all $\rho \in D$ according to their relevance against q. The main purpose of the ranking

* Corresponding author.
E-mail addresses: sromeo@hbku.edu.qa (S. Romeo), gmartino@hbku.edu.qa (G. Da San Martino), belinkov@mit.edu (Y. Belinkov), albarron@hbku.edu.qa (A. Barrón-Cedeño), mohamohamed@hbku.edu.qa (M. Eldesouki), kdarwish@hbku.edu.qa (K. Darwish), hmubarak@hbku.edu.qa (H. Mubarak), glass@mit.edu (J. Glass), amoschitti@hbku.edu.qa (A. Moschitti).

http://dx.doi.org/10.1016/j.ipm.2017.07.003
0306-4573/© 2017 Elsevier Ltd. All rights reserved.

model is to improve the user's experience by (*i*) performing a live search on the previously-posted questions, potentially fulfilling the user's information need at once and (*ii*) avoiding the posting of similar questions, particularly if they have already been answered. From the natural language processing point of view this can also be the source of a collection of question paraphrases and near-duplicates, which can be further explored for other tasks.

Our model for question ranking uses Support Vector Machines. We use a combination of tree kernels (TKs) applied to syntactic parse trees, and linear kernels applied to features constituted by different textual similarity metrics computed between q and ρ. We build the trees with the constituency parser of Farasa—which we introduce in this paper for the first time— and compare it against the well-consolidated Stanford parser (Green & Manning, 2010). Additionally, we integrated Farasa in a UIMA-based cQA pipeline[1] which provides powerful machine learning features for question similarity assessment and reranking. Furthermore, we design word embeddings to complement the feature vectors.

In contrast to other question-answering (QA) tasks, forum questions tend to be ill-formed multi-sentence short texts with courtesy fragments, context, and elaborations. As TKs are sensitive to long (irrelevant) texts, we focus on the automatic selection of meaningful text fragments to feed TKs. To do so, we design a selection model based on the weights assigned to each word in the texts by an attention mechanism in a long short-term memory network (LSTM). Such a model can filter out irrelevant or noisy subtrees from the question syntactic trees, significantly improving both the speed and the accuracy of the TKs-based classifier.

The rest of the paper is organized as follows. Section 2 offers the necessary background on general QA and cQA, both in Arabic and in other languages. In Section 3 we take a brief diversion from QA to describe Farasa, the technology we use for Arabic natural language processing. We turn back to QA in Section 4, where we present our question ranking model. Section 5 describes our neural network model designed to improve our tree representation by selecting the most relevant text fragments. Section 6 discusses our experiments and obtained results. Section 7 concluded with final remarks.

2. Background

As models for QA require linguistic resources, work focused on the Arabic language is relatively humble compared to other better-resourced languages, such as English (Strzalkowski & Harabagiu, 2008). Obviously, the scarceness of language resources is not the only issue. In Arabic, characteristics such as a rich morphology, the interaction among multiple dialects, and the common lack of diacritics and capitalization in informal language, pose unprecedented challenges for a QA system to succeed (Ezzeldin & Shaheen, 2012). cQA is one specific scenario of QA. Most of the research work carried out for the Arabic language is focused on standard QA: the search for an answer over a collection of free-text documents. Therefore, this section is divided in three parts. Firstly, we overview some of the literature on Arabic QA. Secondly, we describe the three main stages of a cQA system, including a review of the approaches available to tackle each task, mainly for English. Thirdly, we overview the relatively-scarce literature on cQA for Arabic.

2.1. Question answering in Arabic

Here we overview some of the most representative models proposed to address the three components of a QA system in Arabic: question analysis, passage retrieval, and answer extraction.

In *question analysis*, the task consists of generating the best possible representation for a question q in order to retrieve a subset of relevant documents and, eventually, passages. The question pre-processing applied by Rosso et al. (2005) consists of stopword removal and named entity recognition. Afterwards, they classify q by means of its intended information need—whether q is asking for a *name*, a *date*, a *quantity*, or a *definition*—in order to look for the required information in the retrieved passages. Other approaches also try to extract the question's focus (i.e., the main noun phrase) as well as named entities (Abdelbaki & Shaheen, 2011; Brini W. & L., 2009; Kanaan, Hammouri, Al-Shalabi, & Swalha, 2009).

The resulting representation of q is used for retrieving text passages, p, that might answer the question. One alternative is retrieving those p that include a certain amount of the words or phrases in q. Besides computing a similarity function $sim(q, p)$ (Kanaan et al., 2009), the ranking function can be based on the positional distance among the matching terms in the document (Abouenour, 2011; Benajiba, 2007), i.e., the closer the terms in the document, the more likely it may represent a good answer for q. A semantic expansion on the basis of resources such as the Arabic WordNet can come into play as well (Abouenour, 2011).

Once the most promising text passages have been retrieved, it is time to extract specific answers. Most approaches rely on manually-defined patterns, heuristics, rules, and semantic similarities between question focus and candidate answers; for instance, using n-grams (Abdelbaki & Shaheen, 2011; Trigui, Belguith, & Rosso, 2010).

By addressing these three generic steps, different kinds of questions can be answered. For instance, Al Chalabi (2015) focused on factoid QA by first determining if q is of kind *who, what, when, how*, etc. QASAL (Question-Answering System for Arabic Language) (Brini W. & L., 2009) goes beyond factoid QA by exploiting the linguistic annotation system of NooJ (Silberztein, 2005) to deal with definitional questions as well. Salem, Sadek, Chakkour, and Haskkour (2010) focused on *why* and *how* questions by means of the Rhetorical Discourse Structure (RST) formalism.

[1] It should be noted that our UIMA pipeline with Farasa will be made available to the research community.

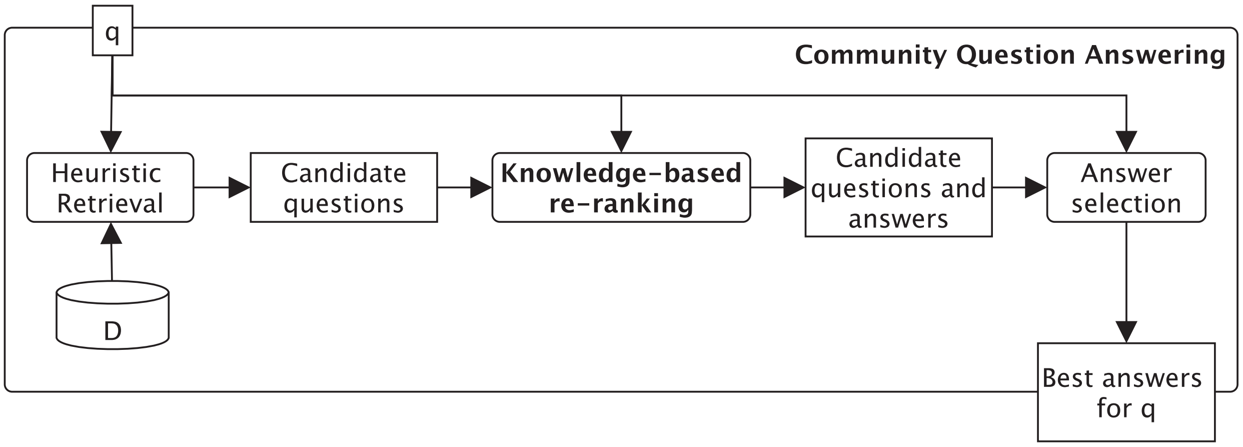

q
Community Question Answering
Heuristic Retrieval
Candidate questions
Knowledge-based re-ranking
Candidate questions and answers
Answer selection
D
Best answers for q

Fig. 1. General architecture of a system for question answering in community-generated forums. q stands for the user question; D is the collection of previously-posted forum questions along with their answers. The re-ranking stage appears highlighted because it is the problem we address in this research work.

2.2. The architecture of a community question answering system

The cQA scenario is slightly different: a new question q formulated by the forum user tends to be less factual and more elaborated, often including contextual information, elaborations, multiple questions, and even irrelevant text fragments. The reference collection D is not composed of free-text documents, but of previously-posted forum questions, together with their answers provided by other users (if any). This leads to building a system architecture as the one represented in Fig. 1, which is inspired by Potthast, Barrón-Cedeño, Stein, and Rosso (2011).

The first step in the cQA architecture is that of heuristic retrieval. Given question q and a relatively-large collection of forum question–answer pairs $\langle \rho, \alpha \rangle \in D$, an inexpensive mechanism is applied to retrieve the most similar (related) questions ρ. Standard information retrieval technology (e.g., a search engine based on inverted indexes), can be applied to solve this task. The creators of the corpus (Nakov et al., 2016) we use for our experiments (Section 6) used Solr[2] to deal with this stage. This step results in the subset of potentially-relevant candidate pairs $D_q \subset D$.

Having q and D_q as input, the *knowledge-based re-ranking* stage is in charge of performing a more refined ordering of the questions. The objective is locating those pairs $\langle \rho, \alpha \rangle \in D$ such that ρ are semantically-equivalent (or at least highly relevant) to q. The relatively-small size of D_q allows for the use of more sophisticated—generally more expensive—technology. This is the task we address in this research work, by applying a combination of kernels on both structural and deep learning features (cf. Section 4).

Extensive work has been carried out to design models for this crucial stage of cQA. Although most of them have been devised for English forums, it is worth mentioning some of the approaches. Cao, Duan, Lin, Yu, and Hon (2008) tackled this problem by judging topic similarity, whereas Duan, Cao, Lin, and Yu (2008) searched for equivalent questions by considering the question's focus as well. Zhou, Cai, Zhao, and Liu (2011) dodged the lexical gap[3] between q and ρ by assessing their similarity on the basis of a (monolingual) phrase-based translation model (Koehn, Och, & Marcu, 2003), built on question–answer pairs in a similar fashion to Jeon, Croft, and Lee (2005). Wang, Ming, and Chua (2009) computed the similarity between q and ρ on top of syntactic-tree representations: the more substructures the trees have in common, the more similar the questions are. The recent boom in neural network approaches has also impacted question re-ranking. dos Santos, Barbosa, Bogdanova, and Zadrozny (2015) applied convolutional neural networks to retrieve semantically-equivalent questions' subjects. They had to aggregate a bag-of-words neural network when dealing with whole questions; that is, subject and (generally long) body. Support vector machines have shown to be highly competitive in this task. For instance, Franco-Salvador, Kar, Solorio, and Rosso (2016) used SVM^{rank} (Joachims, 2006) on a manifold of features, including distributed representations and semantic information sources, such as BabelNet (Navigli & Ponzetto, 2012) and Framenet (Baker & Sato, 2003). Both Barrón-Cedeño et al. (2016) and Filice, Croce, Moschitti and Basili (2016)) achieved a good performance using KeLP (Filice et al., 2015) to combine various kernels with different vectorial and structural features.

Once the most promising questions ρ in the forum are retrieved, potential answers to the new query q are *selected*. The answers α attached to ρ are compared against q in order to estimate their relevance. This is not a trivial problem because the anarchy of Web forums allows users to post irrelevant contents. One of the first approaches to answer selection relied completely on the website's metadata (Jeon, Croft, Lee, & Park, 2006), such as an author's reputation and click counts. Agichtein, Gionis, Castillo, Mishne, and Donato (2008) explored a graph-based model of contributors relationships together

[2] https://lucene.apache.org/solr

[3] The classical IR problem of matching the few query terms in relevant documents.

with both content- and usage-based features. These approaches depend heavily on the forum's meta-data and social features. Still, as Surdeanu, Ciaramita, and Zaragoza (2008) stress, relying on these kinds of data causes the model portability to be difficult; a drawback that disappears when focusing on the content of the questions and answers only. Tran, Tran, Vu, Nguyen, and Bao Pham (2015) applied machine translation in a similar fashion as Jeon et al. (2005) and Zhou et al. (2011), together with topic models, embeddings, and similarities. Hou et al. (2015) and Nicosia et al. (2015) applied supervised models with lexical, syntactic and meta-data features. Some of the most recent proposals aim at classifying whole threads of answers (Joty et al., 2015; Zhou, He, Zhao, & Hu, 2015) rather than each answer in isolation.

This cQA architecture assumes q is a newly-posted question. A hybrid scenario is that of question deduplication. In this case, q is just another question in the forum, together with its corresponding thread of answers. As a result, the information of both the question and its thread of comments can be used to determine if two posts are asking the same or similar questions. Both Ji, Xu, Wang, and He (2012) and Zhang, Wu, Wu, Li, and Zhou (2014) used LDA topic modeling to learn the latent semantic topics that generate question–answer pairs and used the learned topic distribution to retrieve similar historical questions.

It is worth noting that many of the aforementioned approaches (Barrón-Cedeño et al., 2016; Filice et al., 2016; Franco-Salvador et al., 2016; Hou et al., 2015; Nicosia et al., 2015; Tran et al., 2015) were applied during the two editions of SemEval Task 3 on cQA (Nakov et al., 2015; 2016). In this work we take advantage of the evaluation framework developed for Arabic in the 2016 edition (Nakov et al., 2016) (cf. Section 6.1).

2.3. Community question answering for Arabic

As the reader can observe, most of the work on cQA has been carried out for other languages than Arabic, including LiveQA (Agichtein, Carmel, Harman, Pelleg, & Pinter, 2015), which allowed participants to provide answers to *real user questions*, live on the Yahoo! Answers site. To the best of our knowledge, the first effort to come out with a standard framework for the evaluation of cQA models for Arabic is precisely that of Nakov et al. (2015); 2016).

This resource promoted the design of five models for question re-ranking in Arabic. The most successful approach (Mohtarami et al., 2016) included text similarities at both word and sentence level on the basis of word embeddings. Such similarities were computed both between q and ρ, new and retrieved question, respectively, and between q and α, with α being the answer linked to the forum question ρ after performing term selection as a pre-processing step. Barrón-Cedeño et al. (2016) used tree kernels applied to syntactic trees together with some features in common with Mohtarami et al. (2016). A combination of rule-based, text similarities, and word embeddings has shown to give some benefit in Arabic cQA (Belinkov, Barrón-Cedeño, & Mubarak, 2015). Our cQA system reuses ideas and some of the models we developed in Barrón-Cedeño et al. (2016) and Mohtarami et al. (2016).

Magooda et al. (2016) applied language models enriched with medical terms extracted from the Arabic Wikipedia. Finally, Malhas, Torki, and Elsayed (2016) exploited embeddings in different ways, including the computation of average word vectors and covariance matrices. The performance of these models is included in Table 7, as they represent the state-of-the-art in the testbed we use for our experiments.

3. The Farasa Arabic NLP toolkit

For our Arabic processing, we used our in-house pipeline of Arabic tools called *Farasa*[4]—*insight* or *chivalry* in Arabic. The pipeline includes a segmenter, a POS tagger, a named entity recognizer, a dependency parser, a constituency parser, and a diacritizer. The syntactic parser is a new contribution, introduced in this paper for the first time. Farasa is tuned for the news domain and for Modern Standard Arabic (MSA). Still, Farasa can handle other genres along with classical and dialectal Arabic, but at reduced accuracy. This is possible because of the large overlap between MSA and other varieties of Arabic. Farasa fills an important gap in the span of available tools. It is the only comprehensive suite of Arabic tools that is both open source and whose internal subcomponents are competitive with the state of the art. Here we focus on the relevant components for our current task: segmenter, POS tagger, and constituency parser. We pose both segmentation and POS tagging as ranking problems, using kernel-based machines. We pose constituency parsing as a sequence labeling problem, where we use a CRF labeler that uses features from the segmenter and POS tagger. Both SVM and CRF have the advantage of being robust and computationally efficient.

3.1. UIMA architecture for Arabic Natural Language Processing

Our Arabic natural language processing pipeline is based on UIMA.[5] UIMA is a framework that allows for the integration of systems to analyze unstructured information (e.g., text documents) whose aim is to extract new knowledge relevant to the particular application context.

UIMA enables to compose applications with self-contained components. Each UIMA component implements an interface defined by the framework and both the input and output structures are described by means of XML descriptor files. The

[4] Available at http://farasa.qcri.org
[5] https://uima.apache.org

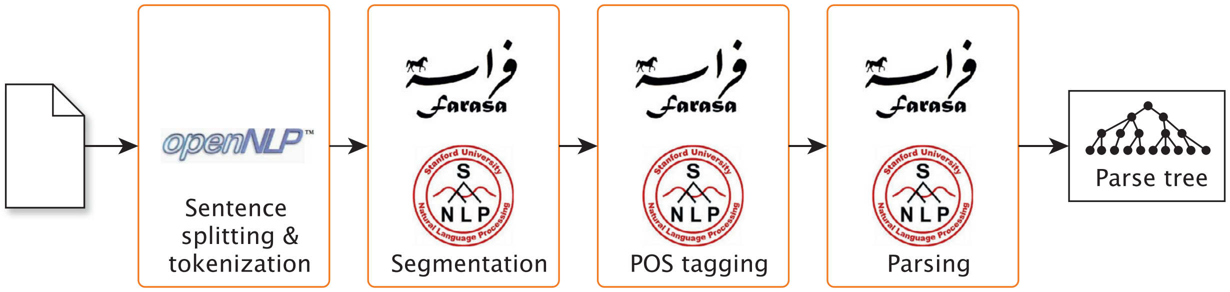

openNLP
Sentence splitting & tokenization
Farasa
Stanford University NLP Natural Language Processing
Segmentation
POS tagging
Parsing
Parse tree

Fig. 2. Our UIMA-based Arabic natural language processing architecture. Each block represents an analysis engine and includes the (alternative) technology it encompasses.

Table 1
Accuracy of segmentation and POS tagging for Farasa and Madamira.

Task|System	Farasa (%)	Madamira (%)
Segmentation	98.9	98.8
POS tagging	94.9	95.3

framework is in charge of managing these components, connecting the analysis engines and controlling the data flow. An analysis engine (AE) is a software module that analyzes artifacts (e.g., text) and infers information from them. The analysis engines are built starting from building units called *annotators*. An annotator is a component that analyzes artifacts and produces additional data and/or metadata (e.g., annotation on the analyzed artifact). An AE can contain a single annotator (*primitive AE*) or multiple annotators (*aggregate AE*).

Fig. 2 shows the architecture of our pipeline, composed of four AEs. The modularity and flexibility of UIMA allows us for opting for different software modules to perform each of the tasks painlessly. The first AE uses OpenNLP[6] for sentence splitting, besides performing tokenization. We trained the sentence splitting model on 5*k* sentences from the AQMAR Arabic Wikipedia Supersense corpus (Schneider, Mohit, Oflazer, & Smith, 2012) and NIST's MT06 corpus.[7] For the rest of the AEs, we can opt for using either Farasa's or Stanford's (Green & Manning, 2010) technology. They are in charge of segmentation into clitics, Part of Speech (POS) tagging, and parsing. In Section 6, we will show the impact of using Farasa or Stanford to process the texts, by comparing different question rankers, each using one of the two parsing systems.

In the following subsections we describe the Farasa segmenter, POS tagger, and parser.

3.2. Farasa segmenter

The Farasa segmenter is described in detail in Abdelali, Darwish, Durrani, and Mubarak (2016) and Darwish and Mubarak (2016). The segmenter breaks words into their underlying clitics. For example, the word `wktAbhm` (and their book) is segmented into `w+ktAb+hm`. We pose segmentation as a ranking problem, where the ranker attempts to rank possible segmentations of a word. The segmenter uses SVM^{rank} (Joachims, 2006) with a linear kernel to determine the best segmentation for each word. We used a linear kernel with a trade-off factor between training errors and margin equal to 100 (parameters tuned on offline experiments carried out over a development set). The ranker uses a dense vector of features which is able to generalize well beyond the cases that are observed during training. Additionally, decoding using SVM^{Rank} is computationally efficient as it involves simple vector multiplication, where speed is highly desirable in processing large amounts of data. We also experimented with using CRF-based sequence labeling (Darwish, Abdelali, & Mubarak, 2014), and our SVM^{Rank} approach yields better segmentation results with higher speed. Further, we conducted offline experiments to compare our approach to bidirectional Long Short Term Memory (bi-LSTM) over CRF and the results were comparable. It was trained on parts 1 (v. 4.1), 2 (v. 3.1), and 3 (v. 2) of the Penn Arabic Treebank (ATB) (Maamouri, Bies, Buckwalter, & Mekki, 2004). Instead of testing the segmenter on a subset of ATB (which may lead to artificially-high results due to its limited lexical diversity), we tested our segmenter on a corpus of seventy WikiNews articles from 2013 and 2014 (Darwish & Mubarak, 2016). It contains 18,300 manually-segmented and POS tagged words from articles on seven domains: politics, economics, health, science and technology, sports, arts, and culture.[8]

Table 1 reports on the segmentation accuracy of Farasa and compares it to that of Madamira (Pasha et al., 2014)—a popular state-of-the-art system—on the WikiNews corpus. The performance of the Farasa segmenter is competitive.

[6] https://opennlp.apache.org
[7] https://www.nist.gov/programs-projects/machine-translation
[8] The corpus is available at https://github.com/kdarwish/Farasa.

Table 2
Part-of-speech tag set of Farasa.

POS	Description	POS	Description
ADV	Adverb	ADJ	Adjective
CONJ	Conjunction	DET	Determiner
NOUN	Noun	NSUFF	Noun suffix
NUM	Number	PART	Particles
PREP	Preposition	PRON	Pronoun
PUNC	Punctuation	V	Verb
ABBREV	Abbreviation	CASE	Alef of tanween fatha
FOREIGN	Non-Arabic as well as non-MSA words	FUT_PART	Future particle “s” prefix and “swf”

3.3. Farasa part-of-speech tagger

Our Arabic part-of-speech tagger uses the simplified PATB tag set proposed by Darwish et al. (2014). Table 2 includes the tags. The POS tagger attempts to find the optimal tag for each clitic produced by the segmenter, as well as determining the gender (masculine or feminine) and number for nouns and adjectives (singular, dual, or plural). Like the segmenter, the POS tagger uses SVM^{Rank} to find the best tag for each clitic. We decided to adopt SVM^{Rank} for POS tagging for the reasons mentioned earlier for segmentation. Additionally, our SVM^{Rank} outperforms a CRF sequence labeling model (Darwish et al., 2014) and is on par with using a bi-LSTM model (Darwish, Mubarak, Abdelali, & Eldesouki, 2017). Thus we construct a feature vector for each possible POS tag for each clitic. We supply these vectors to SVM^{Rank} indicating which vector should be ranked the highest given the weights. We then used SVM^{Rank} (Joachims, 2006) to learn feature weights. As for the segmenter, we used a linear kernel with a trade-off factor between training errors and margin equal to 100 (parameters tuned on offline experiments carried out over a development set). All possible POS tags for a clitic are scored using the classifier, and the POS with the highest score is picked.

Given a sentence composed of the clitics $c_{-n}, \ldots, c_0, \ldots, c_m$, where c_0 is the current clitic and its proposed POS tag, we train the classifier using the following features, computed by maximum-likelihood estimation on our training corpus:

- $p(POS|c_0)$ and $p(c_0|POS)$.
- $p(POS \mid c_{-i}, \ldots, c_{-1})$ and $p(POS \mid c_1, \ldots, c_j) \mid i, j \in [1, 4]$.
- $p(POS \mid c_{-i_{POS}}, \ldots, c_{-1_{POS}})$ and $p(POS \mid c_{1_{POS}}, \ldots, c_{j_{POS}})$; $i, j \in [1, 4]$. Since we don’t know the POS tags of these clitics *a priori*, we estimate the conditional probability as

  $$\sum p(POS \mid c_{-i_{possible\_POS}}, \ldots, c_{-1_{possible\_POS}}).$$

  For example, if the previous clitic could be a NOUN or an ADJ, then $p(POS \mid c_{-1}) = p(POS \mid NOUN) + p(POS \mid ADJ)$.

If the clitic is a stem, we also compute the following features:

- $p(POS \mid stem\_template)$. Arabic words are typically derived from a closed set of roots that are placed in so-called stem templates to generate stems. For example, the root `ktb` can be fit in the template `CCAC` to generate the stem `ktAb` (book). Stem templates may conclusively have one POS tag (e.g., `yCCC` is always a verb) or favor one tag over another (e.g., `CCAC` is more likely a NOUN than an ADJ).
- $p(POS|prefix)$ and $p(POS|suffix)$. Some prefixes and suffixes restrict the possible POS tags for a stem. For example, a stem preceded by DET is either a NOUN or an ADJ.
- $p(POS \mid prefix, prev\_word\_prefix)$, $p(POS \mid prev\_word\_suffix)$ and $p(POS \mid prev\_word\_POS)$. Arabic has agreement rules for noun phrases and idafa constructs (Noun+Noun relation) that cover definiteness, gender, and number. Both these features help capture agreement indicators.

In case we could not compute a feature value during training (e.g., a clitic was never observed with a given POS tag), the feature value is set to $\epsilon = 10^{-10}$. If the clitic is a prefix or a suffix, stem-specific features are assigned the same ϵ value.

In order to improve efficiency and reduce the choices the classifier needs to pick from, we employ some heuristics that restrict the possible POS tags to be considered by the classifier: (i) If the clitic is a number (composed of digits or spelled in words), restrict to “NUM”. (ii) If all the characters are Latin, restrict to “FOREIGN”. (iii) If it is a punctuation mark, restrict to “PUNCT”. (iv) If the clitic is a stem and we can figure out the stem-template, restrict to POS tags that have been seen for that stem-template during training. (v) If the clitic is a stem, restrict to POS tags that have been seen during training, given the prefixes and suffixes of the word.

We trained the POS tagger using the same partitions of the ATB that we used for the segmenter (cf. Section 3.2). Table 1 shows the accuracy of our POS tagger on the WikiNews dataset (Darwish & Mubarak, 2016) and compares it to Madamira. Madamira edges Farasa by 1.6%. A manual inspection on a random sample of 100 errors showed that 54% of the miss-classifications come from the confusion between adjectives and nouns, whereas 13% are between verbs and nouns. Errors in the preliminary segmentation step cause 21% of the POS mistakes. In such cases, any assigned POS would be incorrect. Table 3 lists the observed error types (covering 95% of errors) including examples.

Table 3
POS tagging error types and examples; covering 95% of the errors.

Error Type	%	Example
ADJ → NOUN	29	"Al < ElAm *Albdyl*" (*alternative* media) "*Albdyl*" recognized as NOUN
NOUN → ADJ	25	"m$AryE *wykymAnyA*" (*Wikimania* projects) "*wykymAnyA*" recognized as ADJ
Segment Error	21	"blgp *AlbAyvwn*" instead of "*Al+bAyvwn*" (in *Python* language)
V → NOUN	10	"hw *Elm* AlErbyp" (he *taught* Arabic) "*Elm*" recognized as NOUN (*science*)
Function words	7	"mnhA" (from it) recognized as ADJ
NOUN → V	3	"*k$f* Avry" (archaeological *discovery*) "*k$f*" recognized as V (*discovered*)

The POS tagger also assigns gender and number tags to nouns and adjectives. This module is carried over from the Qatara POS tagger (Darwish et al., 2014) and uses the random forest classifier from Weka (Breiman, 2001). The classifier generated 10 trees, with 5 attributes for each tree with unlimited depth, and was trained using 8400 randomly selected unique nouns and adjectives from ATB. The classifier uses the following features: (*i*) stem template; (*ii*) stem template length; (*iii*) POS tag; (*iv*) attached suffix(es); (*v*) whether the word ends with a feminine marker ("At" or "p"); (*vi*) tags that were obtained from a large word list that was extracted from the Modern Arabic Language Dictionary;[9] (*vii*) the 2-gram language-model probability that the word is preceded by masculine or feminine demonstrative articles; and (*viii*) whether the word appears in a gazetteer of proper nouns that have associated gender tags.[10]

For testing, 20-fold cross validation was used. The average accuracy for gender and number classification were 95.6% and 94.9%, respectively (Darwish et al., 2014).

3.4. Farasa constituency parser

The Farasa constituency parser is an in-house re-implementation of the Epic parser (Hall, Durrett, & Klein, 2014); the best-performing Arabic parser in the SPMRL 2013 multilingual constituency parsing shared task (Björkelund, Çetinoğlu, Farkas, Mueller, & Seeker, 2013). The parser uses a CRF model trained on features derived from the Farasa POS tagger. In compliance with the ATB segmentation, we attached determiners and noun suffixes to the stems. For each clitic, we obtain the information provided by the POS tagger, namely the POS, gender, number, whether the clitic has a determiner, and whether the clitic ends with *ta-marbouta*—the feminine singular noun suffix. Given such information, the parser generates surface features for each clitic c_0. Some of these features include the leading and trailing letters in a clitic. The parser uses the leading n letters in the clitic as features ($n \in [1, 5]$). For example, given the clitic `AlktAb` (the book), these features would be {`A`,`Al`,`Alk`,`Alkt`,`AlktA`}. Similarly, the parser uses the trailing l letters in each clitic as features, ($l \in [1, 5]$). A constraint is placed on the leading and trailing letters: the resulting sequence needs to occur 100+ times in the training data. Furthermore, the parser considers span features, where a span is a bracketed sub-tree (e.g., "(NP (NOUN `AlktAb`))"). The span features include the span's first word, last word, and length; the words before and after the span; split point feature; and span shape feature. To ensure a well-formed nested tree, the parser deduces a minimal probabilistic context-free grammar (PCFG). The parser depends primarily on surface features (i.e. derived only from the clitics in the sentence) to provide context and deep syntactic cues.

Depending primarily on the surface features gives the parser two advantages. Firstly, it greatly simplifies the structural components of the parser, which would not affect the parser's efficiency since so many deep syntactic cues have surface manifestations. Secondly, it allows for an easy adaptation to new languages.

We used the SPMRL 2013 shared task dataset (Seddah et al., 2013) considering the same training/dev/test partitions for evaluation. In our first experiment, we used the original gold POS tags from the dataset. In our second experiment, we use the segmentation and POS tagging as generated by Farasa. Table 4 compares Farasa (with the two setups) and the Epic parser (Hall et al., 2014). Although the Farasa parser is a re-implementation of EPIC, the obtained results differ. Farasa parser when trained with the same dataset as the EPIC parser outperforms it on the dev set, but lags behind on the test with a 1.74 drop in F_1 measure. When using the Farasa segmenter and POS tagger to tag words instead of the gold tags we observe a drop of 2.76 and 0.67 for the dev and test sets, respectively. The drop can be attributed to tagging errors that are propagated to the parser. However, the drop of 0.67 on the test is an affordable cost for the automation process.

As aforementioned, the Farasa tools are trained on the news genre written in Modern Standard Arabic (MSA), whereas Web forums commonly contain texts written in informal or Dialectal Arabic (DA). Farasa recognizes most of the dialectal

[9] http://www.sh.rewayat2.com/gharib/Web/31852/

[10] We crawled the gazeteer from a list of Palestinian high school graduates including names and genders and Arabic Wikipedia articles (snapshot from September 28, 2012) that have English equivalents and belong to the Wikipedia categories containing the words 'person', 'birth', and 'death' if it has gender information.

Table 4
F_1-measure for the Farasa parser compared to the EPIC parser on the SPMRL 2013 shared task dataset. The values are for sentences of all lengths using the *evalb* evaluation script provided by the shared task.

	POS	Dev set	Test set
Farasa parser	Golden	**79.70**	77.01
Farasa parser	Farasa	76.94	76.34
EPIC parser	Golden	78.89	**78.75**

words as out of vocabulary (OOV), which affects negatively POS tagging, NER, and syntactic parsing. For a sample of 100 random questions and answers from the Altibbi question-and-answering medical forum,[11] we found that 20% of questions contain at least one dialectal word while answers are written in MSA by professional doctors. In this domain, we found that the majority of the DA words are function words, whereas content words and terms, such as diseases and body parts, are written in MSA. At the semantic level, this is less important compared to the effect at the syntactic level.

A small degradation in accuracy in Arabic QA systems may occur when using Farasa, designed for MSA, when dealing with DA. Nevertheless, as our results in Section 6 show, this degradation is not important.

4. Kernels for question re-ranking

Now we focus on the re-ranking step of cQA, having as input a query question and a set of question-answer pairs, previously retrieved from a Web forum (cf. Section 2.2). Let $\mathcal{Q}$ and $\mathcal{A}$ be the set of questions and answers (passages) from the forum, respectively. Let q be a new question. Our task is to model a scoring function, $r : \mathcal{Q} \times \mathcal{Q} \times \mathcal{A} \to \mathbb{R}$, which reranks k question–answer pairs, $\langle \rho, \alpha \rangle$, where $\rho \in \mathcal{Q}$, $\alpha \in \mathcal{A}$, with respect to their relevance to q. Please note that $\mathcal{Q} \times \mathcal{A} = D$, which we used in other sections for a more compact reference. We design our scoring function as:

$$r(q, \rho, \alpha) = \vec{w} \cdot \phi(q, \rho, \alpha). \tag{1}$$

We can use implicit representations in kernel-based machines, e.g., SVMs, by expressing $\vec{w}$ as

$$\vec{w} = \sum_{i=1}^{n} \tau_i y_i \phi(q_i, \rho_i, \alpha_i), \tag{2}$$

where n is the number of training examples, τ_i are weights, y_i are the example labels (*Relevant* and *Irrelevant*), and $\phi(q_i, \rho_i, \alpha_i)$ is the representation of the question pairs. This leads to the following scoring function:

$$r(q, \rho, \alpha) = \sum_{i=1}^{n} \tau_i y_i \phi(q, \rho, \alpha) \cdot \phi(q_i, \rho_i, \alpha_i) \qquad (3)$$

$$= \sum_{i=1}^{n} \tau_i y_i K\big(\langle q, \rho, \alpha \rangle, \langle q_i, \rho_i, \alpha_i \rangle\big), \qquad (3)$$

where the kernel $K(\cdot, \cdot)$ intends to capture the similarity between pairs of objects constituted by the query and the retrieved question answer pairs. To any $\phi()$ whose codomain is finite corresponds a kernel function $K(x, x')$, defined on the input space such that $\forall x, x'$, $K(x, x') = \phi(x) \cdot \phi(x')$ (Cristianini & Shawe-Taylor, 2000). We used three types of representations: parse trees, features derived from word embeddings (word2vec), and text similarity metrics. We combine them as follows:

$$K\big(\langle q, \rho, \alpha \rangle, \langle q_i, \rho_i, \alpha_i \rangle\big) = \phi_{tk}(q, \rho) \cdot \phi_{tk}(q_i, \rho_i) \tag{4}$$

$$+ \phi_{w2v}(q, \rho, \alpha) \cdot \phi_{w2v}(q_i, \rho_i, \alpha_i) \tag{5}$$

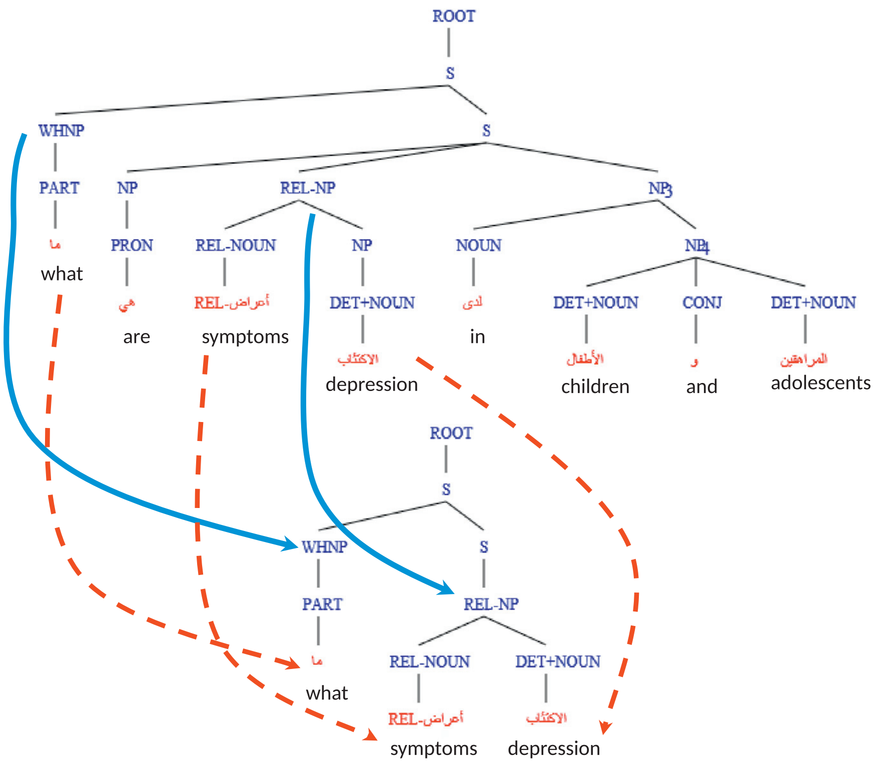
$$+ \phi_{bow}(q, \rho, \alpha) \cdot \phi_{bow}(q_i, \rho_i, \alpha_i). \tag{6}$$

4.1. Tree kernels

We define Eq. (4) as follows

$$\phi_{tk}(q, \rho) \cdot \phi_{tk}(q_i, \rho_i) = TK(t(q, \rho), t(q_i, \rho_i)) + TK(t(\rho, q), t(\rho_i, q_i)), \tag{7}$$

[11] http://www.altibbi.com; this is the source of the corpus we use in this research.



q: ما هي اعراض الاكتئاب لدى الاطفال والمراهقين

(*What are the symptoms of depression in children and adolescents?*)

ρ: ما أعراض الاكتئاب

(*What are depression symptoms?*)

Fig. 3. Constituency trees of two questions connected by REL links. The questions correspond to ids 200,430 and 47,524 in the CQA-MD corpus (Nakov et al., 2016) (cf. Section 6.1).

where *TK* is a tree-kernel function; e.g., the SubSet Tree (SST) Kernel (Moschitti, 2006), which measures the similarity between trees. This way, we do not need to extract syntactic feature vectors from the text pairs (i.e., engineering ϕ_{tk} is unnecessary). We just need to apply TKs to the pairs of syntactic trees, which provides a score representing the structural similarity. We opt for the state-of-the-art TK model proposed by Severyn and Moschitti (2012) and previously used for question ranking in cQA by Barrón-Cedeño, Da San Martino, Romeo, and Moschitti (2016) and Romeo et al. (2016). As described in Eq. (4), we apply TKs to pairs of questions rather than questions with their answers.

The function $t(x, y)$ in Eq. (7) is a string transformation method that returns the parse tree from the text x—the tree computed with Farasa—further enriching it with the REL tags computed with respect to the syntactic tree of y (Severyn & Moschitti, 2012). The REL tags are added to the terminal nodes of the tree of x: a REL tag is added whenever a terminal node of the parse tree of x matches a word in y. Typically, REL tags are also propagated to the parent and grandparent nodes (i.e., up to 2 levels). Fig. 3 shows the syntactic tree of a query and one of its associated forum questions. The dashed arrows indicate a matching between words of the two questions, e.g., *Does treatment* or *effect*, whereas the solid arrows are drawn when entire noun phrases or clauses are (partially) matched, i.e., REL-NP or REL-WHNP. The tree nodes are augmented with the REL tag to mark the connection between the constituents of the two syntactic trees.

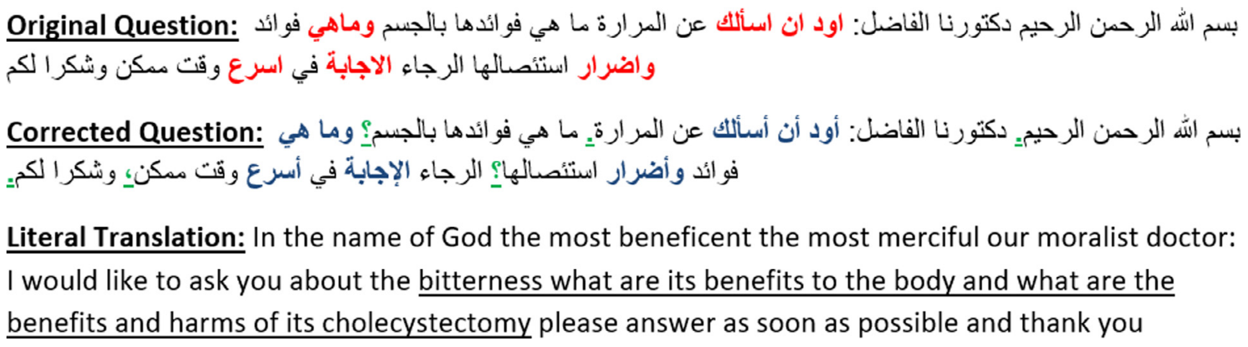
4.2. Representation with embeddings and similarity metrics

Eqs. (5) and (6) convey a combination of distributional, lexical, and morphosyntactic information from the texts.

To generate the vector $\phi_{w2v}(q, \rho, \alpha)$, we use word vectors obtained with the word2vec tool (Mikolov, Yih, & Zweig, 2013), which is trained (with default settings) on the raw corpus provided with the Arabic cQA task. We compute features that capture similarity between q and ρ, and between q and α, in the following way. First, we generate a vector representation for every sentence in q, ρ, and α, by averaging the word vectors in the sentence (excluding stopwords). Then, we find

Table 5
Overview of string, lexical, and syntactic similarity measures.

Metric		Details
String similarity		
Greedy string tiling	(Wise, 1996)	Considering a minimum matching length of 3.
Longest common subsequence	(Allison & Dix, 1986)	Both standard and normalized by the first string.
Longest common substring	(Gusfield, 1997)	Based on generalized suffix trees.
Lexical similarity		
Jaccard coefficient	(Jaccard, 1901)	Over stopworded $[1, \ldots, 4]$-grams.
Word containment	(Lyon, Malcolm, & Dickerson, 2001)	Over stopworded $[1, \ldots, 2]$-grams.
Cosine		Over stopworded $[1, \ldots, 4]$-grams.
		Over $[1, \ldots, 4]$-grams.
		Over $[1, \ldots, 3]$-grams of part of speech.
Syntactic similarity		
PTK	(Moschitti, 2006)	Similarity between shallow syntactic trees.


Original Question: بسم الله الرحمن الرحيم دكتورنا الفاضل: اود ان اسألك عن المرارة ما هي فوائدها بالجسم وماهي فوائد واضرار استئصالها الرجاء الاجابة في اسرع وقت ممكن وشكرا لكم

Corrected Question: بسم الله الرحمن الرحيم. دكتورنا الفاضل: أود أن أسألك عن المرارة. ما هي فوائدها بالجسم؟ وما هي فوائد وأضرار استئصالها؟ الرجاء الإجابة في أسرع وقت ممكن، وشكرا لكم.

Literal Translation: In the name of God the most beneficent the most merciful our moralist doctor: I would like to ask you about the bitterness what are its benefits to the body and what are the benefits and harms of its cholecystectomy please answer as soon as possible and thank you

Fig. 4. Example of forum question with long greetings and introductions, spelling errors, and missing punctuation marks. The most relevant part of the question is underlined.

the two most similar sentences in q and ρ, determined by the cosine similarity between their vector representations, and concatenate their vector representations. We repeat the process for q and α and use their two most similar sentence vectors. Finally, we also find the two most similar word vectors between q and ρ (and between q and α), according to the cosine similarity, and add them to the feature representation.

The features in $\phi_{bow}(q, \rho, \alpha)$ from Eq. (6) are obtained using three kinds of text similarity measures applied between q and ρ, and between q and α: string, lexical, and syntactic. They are included in Table 5.

Our combination of kernels and their corresponding representations is coded in a binary SVM (Joachims, 1999).[12] This formulation combines two of the best models presented at SemEval 2016 Task 3 (Barrón-Cedeño et al., 2016; Belinkov, Mohtarami, Cyphers, & Glass, 2015; Mohtarami et al., 2016) (cf. Section 6.1).

5. Text selection based on neural networks

As shown in Section 2, several neural network approaches have been successfully applied to QA tasks. Unfortunately, question retrieval in cQA is heavily affected by a large amount of noise and a rather different domain, which make it difficult to effectively use out-of-domain embeddings to pre-train neural networks. Fig. 4 illustrates some of the difficulties in cQA questions: long greetings and introductions, spelling errors, and incorrect or missing punctuation marks. Correct grammar and usage of punctuation marks is important for sentence splitting and syntactic parsing. This probably prevented the participants to SemEval tasks from achieving satisfactory results with such models (Nakov et al., 2016). Inspired by Tymoshenko, Bonadiman, and Moschitti (2016), in Romeo et al. (2016) we tried to exploit neural models using their top-level representations for the (q, ρ) pair and fed them into the TK classifier. Nevertheless, this combination proved to be ineffective as well.

Instead of trying to combine the models, we use neural networks to identify the most important pieces of text in both q and ρ. We use an LSTM (Graves, Mohamed, & Hinton, 2013; Hochreiter & Schmidhuber, 1997), augmented with an attention mechanism. LSTMs have proven to be useful in a number of language understanding tasks. Recently Rocktäschel, Grefenstette, Hermann, Kočiský, and Blunsom (2016) adapted an attentional LSTM model (Bahdanau, Cho, & Bengio, 2014) to textual entailment, and a similar model has been applied to cQA (Hsu, Zhang, & Glass, 2016). We follow the same setup of the latter (Section 5.1). Then, we use the attention weights for our text selection algorithm, which aims at removing subtrees containing useless or noisy information (Section 5.2).

[12] Binary SVMs showed comparable results to SVM^{rank} (Joachims, 2002).

5.1. Learning word importance with LSTM

The main idea of learning the importance of words for a task is to use the data and labels about the task itself. Given a pair (q, ρ), we learn two serial LSTM models: LSTM$_q$ reads the word vectors of q, one by one, and records the corresponding memory cells and hidden states; the final memory cell is used to initialize LSTM$_\rho$, which reads the word vectors of ρ.

Formally, an LSTM computes the hidden representation for input x_t with the following iterative equations:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{mi}m_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{mf}m_{t-1} + b_f) \\ m_t &= f_t \odot m_{t-1} + i_t \odot \tanh(W_{xm}x_t + W_{hm}h_{t-1} + b_m) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{mo}m_t + b_o) \\ h_t &= o_t \odot \tanh(m_t) \end{aligned}$$

where σ is the sigmoid function, $\odot$ is element-wise multiplication, and i, f, o, and m are input, forget, output, and memory cell activation vectors. The crucial element is the memory cell m that is able to store and reuse long term dependencies over the sequence. The W matrices and b bias vectors are learned during training.

The final hidden state of LSTM$_\rho$, $\vec{h}_{\rho,N}$, is used as a feature vector to feed a multi-layer perceptron (MLP) with one hidden layer, followed by a softmax classifier. The objective function is the cross-entropy objective over binary relevant/irrelevant target labels.

Given the hidden states produced by LSTM$_q$, we compute a weighted representation of q:

$$\vec{h}_q = \sum_{i=1}^{L} \beta_i \vec{h}_{q,i} \ , \tag{8}$$

where $\vec{h}_{q,i}$ are the hidden states corresponding to the words of q, and the attention weights β_i are computed as:

$$\beta_i = \frac{exp(a(\vec{h}_{q,i}, \vec{h}_{\rho,N}))}{\sum_{j=1}^{L} exp(a(\vec{h}_{q,j}, \vec{h}_{\rho,N}))} \ . \tag{9}$$

Here $a()$ is parameterized as a MLP with one hidden layer and a *tanh* non-linearity (Rocktäschel et al., 2016). The input to the MLP is then a concatenation of $\vec{h}_q$ and $\vec{h}_{\rho,N}$.

Intuitively, β_i assigns a higher weight to words in q if they are useful for determining the relation to ρ. As we will see, these attention weights turn out to be useful for selecting important parts of the questions for the TK models. Note also that the attention here is one-sided —only on q. In practice, we train another model, with attention on ρ, and use its weights as well.

5.2. Parse tree pruning based on neural networks

Our tree-pruning approach to text selection is illustrated in Algorithm 1. Its main idea is to filter out the leaf nodes of the parse tree corresponding to words associated with weights lower than a user-defined threshold, where the word

```
1  Function PruneTree ($T$, $th$);
   Input  : a tree T;
            a pruning threshold $th$;
   Output: a pruned version of $T$
2  pruneNode(root($T$), $th$);
3  Function pruneNode ($o$, $th$);
4  if $|children(o)| > 0$ then
5      for $ch \in children(o)$ do
6          pruneNode($ch$, $th$);
7      end
8      if $|children(o)| = 0$ && !REL_Node(o)) then
9          remove ($o$, $T$);
10     end
11 else
12     if $o.weight < th$ && !REL_Node(o)) then
13         remove ($o$, $T$);
14     end
15 end
```

Algorithm 1: Function *PruneTree* for pruning a tree according to attention weights.

Table 6
Statistics about the CQA-MD corpus (borrowed from Nakov et al., 2016).

Category	Train	Dev	Test	Total
Questions	**1031**	**250**	**250**	**1531**
QA Pairs	**30,411**	**7384**	**7369**	**45,164**
– *Direct*	*917*	*70*	*65*	*1052*
– *Related*	*17,412*	*1446*	*1353*	*20,211*
– *Irrelevant*	*12,082*	*5868*	*5951*	*23,901*

Table 7
MAP scores of the official submissions to the SemEval 2016 Task 3-D. In addition we report MAP values for the development set of our systems.

		Submission	Dev.	Test
1	(Mohtarami et al., 2016)	SLS	47.31	45.83
2	(Barrón-Cedeño et al., 2016)	ConvKN	42.67	45.50
3	(Magooda et al., 2016)	RDI_team	–	43.80
4	(Malhas et al., 2016)	QU-IR	–	38.63
5	(El Adlouni, Lahbari, Rodríguez, Meknassi, & El Alaoui, 2016)	UPC_USMBA	–	29.09
	Random baseline		–	29.79

weights are provided by Eq. (9). The most important step of Algorithm 1 is the recursive function *pruneNode*, which is initially invoked for the root node of the tree. Function *pruneNode* checks whether the node n is a leaf (Line 4) and then applies the appropriate strategy: (i) for non-leaf nodes, *pruneNode* is invoked for the children of o, then o is removed if all of its children are removed and (ii) a leaf node is removed if its weight is lower than the user-defined threshold, *th*. REL-tagged nodes are never removed, regardless of their weight. Different thresholds determine different percentages of pruned nodes, and we explore various thresholds as part of our experiments.

6. Evaluation of question re-ranking models

In this section, we aim at analyzing the impact of the different representation components in the cQA question re-ranking task. Section 6.1 describes the experimental settings. Section 6.2 illustrates the experimental methodology. Our experiments evaluate four aspects: (i) the impact of the NLP processors, (ii) the performance of kernels on vectorial features and tree kernels used in isolation, (iii) the performance of kernel combinations, and (iv) the impact of text selection using tree pruning. We analyze and discuss the results in Section 6.3.

6.1. Evaluation framework

We perform our experiments using the evaluation framework released in the SemEval 2016 Task 3-D (Nakov et al., 2016). The framework consists of a corpus in Arabic from the medical domain—the CQA-MD corpus—and a set of evaluation metrics. Nakov et al. (2016) queried different Web forums to build up a collection of query questions linked to a set of 30 candidate forum questions–answer pairs. The outcome: a total of 45,164 question–answer forum pairs attached to one of 1,531 query questions. The relevance of each $\rho \in D$ was manually annotated by means of *crowdsowrcing* considering three labels: *Direct* if ρ contains a direct answer to q; *Related* if ρ covers some of the aspects asked by q; and *Irrelevant* if ρ and q are unrelated. An ideal ranking should place all direct and relevant $\rho \in D$ on top, followed by the irrelevant pairs. Table 6 shows some statistics of the dataset. The answer associated with each of the 30 forum questions was provided by a professional physician and it is considered correct.

The official evaluation measure is Mean Average Precision (MAP); a standard evaluation metric in information retrieval computed as

$$MAP = \frac{\sum_1^{|Q|} AveP(q)}{|Q|} \quad , \tag{10}$$

where Q is the set of test questions and *AveP* is the average precision value for each query, computed as

$$AveP(q) = \frac{\sum_{k=1}^{|D_q|} (P(k) \times rel(k))}{|\{relevant\ documents\}|} \quad , \tag{11}$$

where $|D_q|$ is the number of retrieved pairs in the ranking, $rel(k) = 1$ if ρ at position k is relevant, and $P(k)$ is computed as

$$P(k) = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|_k}{k} \quad ; \tag{12}$$

that is, the size of the intersection between relevant and retrieved documents up to rank k divided by k.

Table 8
MAP performance for our ranking models when applied in isolation on the development and test partitions.

Model	Dev.	Test
Linear-kernel SVM on Word2vec and sims	44.94	40.73
Tree-kernel SVM on Farasa Parse trees	42.53	40.87
NN (attention on q)	34.85	33.40
NN (attention on ρ)	37.47	35.09

6.2. Experiments and methodology

Our experiments address the question re-ranking stage in the architecture for community question answering (cf. Section 2). That is, given a query q, re-rank a collection of related question–answer pairs in D_q. In order to do that, we stick to the same training/development/test partition defined by Nakov et al. (2016) for the SemEval 2016 cQA challenge. Regarding the implementation of the models, for the word2vec representations, we trained the embeddings on 26M words of unsupervised data, provided together with the CQA-MD corpus.

We designed four follow-up experiments of increasing complexity:

Experiment 1: Impact of NLP processors. Our first experiment uses only a tree-kernel SVM on parse trees. The difference between our two runs is that we either use Farasa or Stanford's (Green & Manning, 2010) technology to generate the parse-tree representations. This allows for an implicit comparison of these two parsers.

Experiment 2: Isolated models. We perform tests on our three re-ranking models in isolation. Beside the tree-kernel SVM on parse trees from Experiment 1, we experiment with a linear-kernel SVM on word2vec and similarity representations and with the attentional LSTM neural network.

Experiment 3: Kernel combination. We combine two SVM kernels on different features: tree kernels on the parse trees and the linear kernel on the word2vec and similarity representations.

Experiment 4: Tree Pruning. We explore different thresholds to prune the parse trees on the basis of the LSTM attention weights before learning the scoring function with an SVM. Specifically, we perform experiments combining tree kernels with the linear kernel on word2vec and similarity features.

6.3. Results and discussion

In order to provide a more comprehensive perspective of our experimental results, Table 7 reports the MAP values obtained by the participant systems on the test set of SemEval 2016 Task 3-D. It should be noted that we designed both the two top systems, SLS and ConvKN. The first one was based on a committee of four different systems using different embedding versions as well as methods for filtering the initial word representation, whereas the second applied tree kernels and similarity metrics. In this paper, we only used one system from SLS, corresponding to our linear kernel, which performs relatively more stably with respect to both development and test sets. Although committees are rather effective and typically produce higher accuracy than a single system, they tend to obscure the contribution of the different representations, which are the main target of our study.

It is worth noting that the test set results in Table 7 are obtained by models trained on the training data merged with the development set. Thus, such results are generally higher than those we obtain in this paper on the test set, where we only use the training set for learning all our models. We preferred this approach for our experiments so that we can better compare the results between development and test sets and, at the same time, have a faster training and test processing.

6.3.1. Experiment 1: impact of NLP processors

As a way to compare Farasa and Stanford parsers, we ran a set of experiments in which the only difference was the processor used to generate the trees. We used an SVM with $C = 1$ and the normalized SST kernel (Collins & Duffy, 2001) as TK in Eq. (7) with the following values for the parameter $\lambda = \{0.001, 0.01, 0.05, 0.1, 0.2\}$, which provide different weights to subtrees of different size. Changing λ, we can emphasize different portions of the parse trees and thus carry out a more systematic comparison between the parsers.

Fig. 5 shows the MAP evolution for the two models, with respect to the λ parameter of the kernel. The highest MAP values on development (39.93) and test (38.49) sets are obtained when using Farasa. In such cases the increment with respect to Stanford is of 1.44 and 0.88 MAP points, respectively. This is an interesting result as it is in line with our linguistic expert of Arabic who, analyzing some of the trees generated on our data by both parsers, observed a better quality of the Farasa POS-tagger than the one used in the Stanford parser. This different quality also affects chunk definition and their dependencies. It seems that using the entire structure of the parse tree allows TKs to benefit from an overall better quality of Farasa parser to produce better rankings.

6.3.2. Experiment 2: isolated models

Table 8 shows the performance of our ranking models when applied in isolation. The linear- and the tree-kernel models perform on par with each other on the test set, both obtaining competitive results. Still, they lie behind the top 2 systems included in Table 7, at MAP values of ∼40.8 on the test set.

As aforementioned, the neural network does not reach a competitive performance, maybe due to the small amount of data available for training. However, this is not the only contribution the network model can provides as we can use its weights for text selection.

6.3.3. Experiment 3: kernel combination

The first row of Table 9 reports the performance of the combination of the tree kernel on parse trees built with Farasa and the linear kernel on word2vec and similarity features. Note that the combination improves over tree kernel and linear kernel in isolation. With respect to our previous systems, i.e., SLS and ConvKN, we got lower values for the test set: as previously pointed out, (*i*) SLS is a combination of four different systems; and (*ii*) in this paper, we only use the training data, whereas we trained SLS and ConvKN on both the training and development sets to obtain the test set results.

6.3.4. Experiment 4: tree pruning

While combining feature vectors and tree kernels improves the MAP scores in our experiments, the use of tree kernels has a negative impact on the running time. Thus, we prune parse trees as described in Section 5.2.

In this experiment, we evaluate the combination of the linear kernel on word2vec and similarity features with the SST kernel over syntactic trees. Both kernels are not normalized. The top two plots of Fig. 6 show prediction and learning time (in minutes) as a function of the ratio of pruned nodes. As expected both learning and prediction times decrease roughly linearly with respect to the number of pruned tree nodes.

The plot at the bottom shows the corresponding MAP values, again as a function of the ratio of pruned nodes. Rather than decreasing due to the reduced representation, the MAP scores increase, reaching 46.78 (+0.20 with respect to no pruning) on the development set and 42.20 (+1.11) on the test set. This occurs because our pruning model manages to filter out irrelevant fragments from the trees. For instance, discarding the phrase “in children and adolescents” in Fig. 3 would allow a model to better determine that the two questions are practically equivalent.

The threshold maximizing MAP on the development set is the one corresponding to 0.74 pruning ratio (see second line of Table 9). Its MAP score on the test set is 41.93 (+0.84) and the learning and prediction times decrease from 887 to 295 min and from 98 to 20 min, respectively, with respect to the unpruned data. This means that learning and prediction processes are 3 and 4.9 times faster than the kernel combination without pruning.

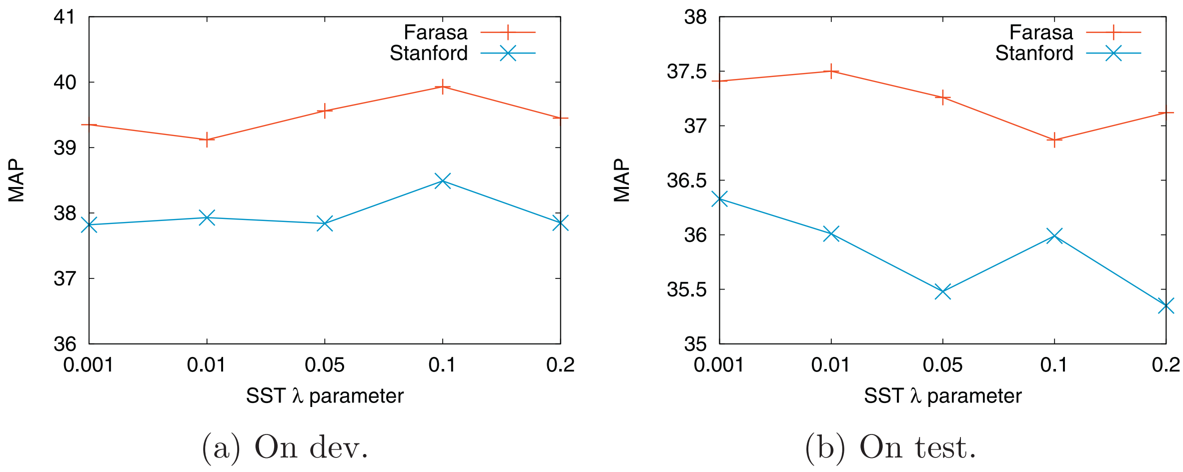


Fig. 5. MAP as a function of the λ parameter of the SST kernel. We compare the performance of our tree-kernel model when the parse-tree representation is built with either Farasa or Stanford.

Table 9
MAP performance for our ranking models when applied in combination and after pruning. The latter was applied with two different thresholds, 0.74 and 0.82, which obtained the highest MAP on development and test sets, respectively.

Model	Dev.	Test
Tree-kernel (no pruning) + Word2vec and sims.	46.58	41.09
Tree-kernel (pruning ratio 0.74) + Word2vec and sims.	46.78	41.93
Tree-kernel (pruning ratio 0.82) + Word2vec and sims.	46.01	42.20

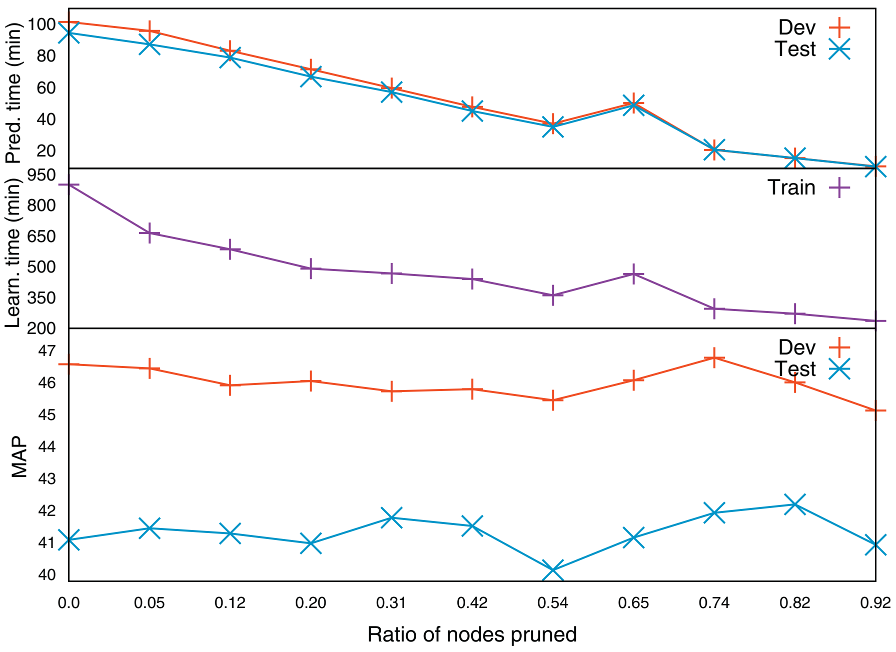


Fig. 6. Experiments with pruned trees. From top to bottom the plots show the prediction time, the learning time and MAP as a function of the ratio of pruned nodes.

7. Conclusions

Recently, community-driven question answering in websites (cQA) has seen a renewed interest both from natural language processing and information retrieval researchers. Most work in cQA has been carried out for the English language, resulting in a lack of techniques and resources available to deal with other languages, such as Arabic. Motivated by this aspect, in this paper we addressed the problem of cQA in an Arabic forum. In particular, we focused on the task of question re-ranking: given a newly-posted question, retrieve equivalent or similar questions already in the forum. If similar questions have been addressed in the past, the users can quickly obtain an answer to their question.

In order to deal with the necessary processing of the Arabic texts, for the first time, we introduced some components of our in-house pipeline of Arabic NLP tools called Farasa. This includes a segmenter, a POS tagger, a named entity recognizer, a dependency parser, a constituency parser, and a diacritizer. We integrated Farasa into our cQA architecture using the UIMA-based framework. This way, we could extract effective features, such as lexical and syntactic information from Arabic text, and feed them into our machine learning models. Our evaluation on a realistic collection of forum questions in the medical domain allowed us to test Farasa's capabilities when dealing with a real-world application.

In particular, we addressed the task of question re-ranking as a binary classification problem, where each example represents a pair {user-question, forum-question}. We proposed an effective combination of tree kernels built on top of the constituency parse trees provided by Farasa and Arabic word embeddings based on neural networks. This combination allowed for better capturing the semantic relatedness between two short pieces of text, i.e., questions and pairs of questions and answers, and achieved state-of-the-art performance for Arabic question re-ranking.

Additionally, we designed models for selecting meaningful text in order to reduce noise and computational cost. For this purpose, we applied long short-term memory neural networks to identify the best subtrees in the syntactic parsing of questions, which are then used in our tree-kernel-based ranker. We combined the text selection approach with word embeddings based on neural networks, boosting the performance. With thorough experiments we showed that (*i*) syntactic information is very important for the question ranking task, (*ii*) our model combining tree kernels, word embeddings and neural networks for text selection is an effective approach to fully exploit advanced Arabic linguistic processing and (*iii*) our reranker based on tree kernels can be used to implicitly evaluate the performance of different syntactic parsers.

Finally, our UIMA pipeline for Arabic NLP as well as for cQA will be made available to the research community.

Acknowledgments

This research was performed by the Arabic Language Technologies (ALT) group at the Qatar Computing Research Institute (QCRI), HBKU, part of Qatar Foundation. It is part of the Interactive sYstems for Answer Search (Iyas) project, which is developed in collaboration with MIT-CSAIL.

References

Abdelali, A., Darwish, K., Durrani, N., & Mubarak, H. (2016). Farasa: A fast and furious segmenter for Arabic. In Proceedings of association for computational linguistics: Human language technologies, 11–16

Abdelbaki, H., & Shaheen, M. (2011). ARQA high-performance arabic question answering system. *Arab academy for science and technology and maritime transport*. Alexandria, Egypt.

Abouenour, L. (2011). On the improvement of passage retrieval in Arabic question/answering (Q/A) systems. In *Proceedings of international conference on application of natural language to information systems* (pp. 336–341). Springer.

Agichtein, E., Carmel, D., Harman, D., Pelleg, D., & Pinter, Y. (2015). Overview of the TREC 2015 LiveQA track. In *Proceedings of Trec*.

Agichtein, E., Gionis, A., Castillo, C., Mishne, G., & Donato, D. (2008). Finding high-quality content in social media with an application to community-based question answering. In *Proceedings of international conference on web search and data mining, WSDM*.

Al Chalabi, H. M. (2015). *Question processing for Arabic question answering system*. The British University in Dubai Ph.D. thesis..

Allison, L., & Dix, T. (1986). A bit-string longest-common-subsequence algorithm. *Information Processing Letters, 23*(6), 305–310.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Baker, C. F., & Sato, H. (2003). The framenet data and software. In K. Funakoshi, S. Kübler, & J. Otterbacher (Eds.), *Proceedings of 41st annual meeting of the association for computational linguistics, ACL 2003, companion volume to the proceedings, 7–12 july 2003, Sapporo convention center, Sapporo, Japan* (pp. 161–164). The Association for Computer Linguistics.

Barrón-Cedeño, A., Da San Martino, G., Joty, S., Moschitti, A., Al-Obaidli, F., Romeo, S., et al. (2016). ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on Arabic and English fora. In *Proceedings of SemEval 2016* (pp. 896–903).

Barrón-Cedeño, A., Da San Martino, G., Romeo, S., & Moschitti, A. (2016). Selecting sentences versus selecting tree constituents for automatic question ranking. In *Proceedings of COLING 2016* (pp. 2515–2525).

Belinkov, Y., Barrón-Cedeño, A., & Mubarak, H. (2015). Answer selection in Arabic community question answering: A feature-rich approach. In *Proceedings of the second workshop on arabic natural language processing* (pp. 183–190). Beijing, China: Association for Computational Linguistics.

Belinkov, Y., Mohtarami, M., Cyphers, S., & Glass, J. (2015). VectorSLU: A continuous word vector approach to answer selection in community question answering systems. In *Proceedings of SemEval 2015*. http://www.aclweb.org/990anthology/S15-2038.

Benajiba, Y. (2007). *Arabic question answering*. Spain: Technical University of Valencia Master's thesis.

Björkelund, A., Çetinoğlu, Ö., Farkas, R., Mueller, T., & Seeker, W. (2013). (Re) Ranking meets morphosyntax: state-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of SPMRL 2013* (pp. 135–145).

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

Brini, W., Ellouze M., & Mesfar, S., Belguith, L. H. (2009). An Arabic question-answering system for factoid questions. In Proceedings of IEEE international conference on natural language procesing and knowledge engineering, IEEE NLP-KE 2009, Dalian, China(pp. 1–7).

Cao, Y., Duan, H., Lin, C.-Y., Yu, Y., & Hon, H.-W. (2008). Recommending questions using the Mdl-based tree cut model. In *Proceedings of the 17th international conference on world wide web, WWW 2008* (pp. 81–90). New York, NY, USA: ACM. doi:10.1145/1367497.1367509.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Proceedings of Neural information processing systems, NIPS* (pp. 625–632). MIT Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods* (1st). Cambridge University Press.

Darwish, K., Abdelali, A., & Mubarak, H. (2014). Using stem-templates to improve Arabic POS and gender/number tagging. In *Proceedings of LREC 2014* (pp. 2926–2931). http://www.lrec-conf.org/proceedings/lrec2014/pdf/335_912Paper.pdf.

Darwish, K., Mubarak, H., et al. (2016). Farasa: A new fast and accurate Arabic word segmenter. In N. C. C. Chair, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, & B. Maegaard (Eds.), *Proceedings of the tenth international conference on language resources and evaluation, LREC* (pp. 1070–1074). Paris, France: European Language Resources Association (ELRA).

Darwish, K., Mubarak, H., Abdelali, A., & Eldesouki, M. (2017). Arabic pos tagging: Dont abandon feature engineering just yet. Proceedings of workshop on Arabic natural language processing, WANLP 2017 (co-located with EACL 2017), (p. 130).

dos Santos, C., Barbosa, L., Bogdanova, D., & Zadrozny, B. (2015). Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of ACL-HLT 2015* (pp. 694–699).

Duan, H., Cao, Y., Lin, C.-Y., & Yu, Y. (2008). Searching questions by identifying question topic and question focus, In Proceedings of ACL-HLT 2008, 156–164

El Adlouni, Y., Lahbari, I., Rodríguez, H., Meknassi, M., & El Alaoui, S. O. (2016). UPC-USMBA at SemEval-2016 Task 3: UPC-USMBA participation in SemEval 2016 Task 3, subtask D: CQA for Arabic. In *Proceedings of SemEval 2016*.

Ezzeldin, A., & Shaheen, M. (2012). A survey of Arabic question answering: Challenges, tasks, approaches, tools, and future trends. In *Proceedings of the 13th international arab conference on information technology, ACIT 2012* (pp. 1–8).

Filice, S., Castellucci, G., Croce, D., Da San Martino, G., Moschitti, A., & Basili, R. (2015). KeLP: A kernel-based learning platform in Java. In *Proceedings of the workshop on machine learning open source software: Open ecosystems*. Lille, France: International Conference of Machine Learning.

Filice, S., Croce, D., Moschitti, A., & Basili, R. (2016). KeLP at SemEval-2016 Task 3: Learning semantic relations between questions and answers. In *Proceedings of SemEval 2016* (pp. 1116–1123). http://www.aclweb.org/anthology/S16-1172.

Franco-Salvador, M., Kar, S., Solorio, T., & Rosso, P. (2016). Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. Proceedings of international workshop on semantic evaluation, SemEval, 16, 814–821.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of international conference on acoustics, speech and signal processing, ICASSP* (pp. 6645–6649).

Green, S., & Manning, C. D. (2010). Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd international conference on computational linguistics, COLING 2010* (pp. 394–402). Stroudsburg, PA, USA: Association for Computational Linguistics.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences computer science and computational biology*. Cambridge University Press.

Hall, D. L. W., Durrett, G., & Klein, D. (2014). Less grammar, more features. In *Proceedings of the 52nd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing* (pp. 228–237).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Hou, Y., Tan, C., Wang, X., Zhang, Y., Xu, J., & Chen, Q. (2015). HITSZ-ICRC: Exploiting classification approach for answer selection in community question answering. In *Proceedings of SemEval 2015* (pp. 196–202).

Hsu, W., Zhang, Y., & Glass, J. R. (2016). Recurrent neural network encoder with attention for community question answering. *CoRR*. abs/1603.07044.

Jaccard, P. (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.

Jeon, J., Croft, W. B., & Lee, J. H. (2005). Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on information and knowledge management* (pp. 84–90). Bremen, Germany.

Jeon, J., Croft, W. B., Lee, J. H., & Park, S. (2006). A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th ACM SIGIR conference on research and development in information retrieval, SIGIR 2006* (p. 228). New York, New York, USA: ACM Press.

Ji, Z., Xu, F., Wang, B., & He, B. (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* (pp. 2471–2474). ACM.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods* (pp. 169–184). Cambridge, MA, USA: MIT Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of conference on knowledge discovery and data mining, KDD* (pp. 133–142).

Joachims, T. (2006). Training linear SVMs in linear time. In *Proceedings of KDD 2006* (pp. 217–226).

Joty, S., Barrón-Cedeño, A., Da San Martino, G., Filice, S., Màrquez, L., Moschitti, A., et al. (2015). Global thread-level inference for comment classification in community question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 573–578). Lisbon, Portugal: Association for Computational Linguistics.

Kanaan, G., Hammouri, A., Al-Shalabi, R., & Swalha, M. (2009). A new question answering system for the Arabic language. *American Journal of Applied Sciences, 6*(4), 797–805.

Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology – volume 1, NAACL 2003* (pp. 48–54). Stroudsburg, PA: Association for Computational Linguistics. doi:10.3115/1073445.1073462.

Lyon, C., Malcolm, J., & Dickerson, B. (2001). *Detecting short passages of similar text in large document collections* (pp. 118–125). Pittsburgh, PA.

Maamouri, M., Bies, A., Buckwalter, T., & Mekki, W. (2004). The penn Arabic treebank: building a large-scale annotated Arabic corpus. In *Proceedings of nemlar conference on arabic language resources and tools: 27* (pp. 466–467).

Magooda, A., Gomaa, A., Mahgoub, A., Ahmed, H., Rashwan, M., Raafat, H., et al. (2016). RDI at SemEval-2016 Task 3: RDI unsupervised framework for text ranking,. In *Proceedings of SemEval 2016* (pp. 822–827).

Malhas, R., Torki, M., & Elsayed, T. (2016). QU-IR at SemEval-2016 Task 3: Learning to rank on Arabic community question answering forums with word embedding, in: [81]. In *Proceedings of SemEval 2016* (pp. 866–871).

Mikolov, T., Yih, W.-t., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2013* (pp. 746–751). Atlanta, GA, USA.

Mohtarami, M., Belinkov, Y., Hsu, W.-N., Zhang, Y., Lei, T., Bar, K., et al. (2016). SLS at SemEval-2016 Task 3: Neural-based approaches for ranking in community question answering. In *Proceedings of SemEval 2016* (pp. 828–835). http://www.aclweb.org/anthology/S16-1128.

Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European conference on machine learning, ECML 2006* (pp. 318–329). Berlin, Germany: Springer-Verlag Berlin Heidelberg.

Nakov, P., Màrquez, L., Magdy, W., Moschitti, A., Glass, J., & Randeree, B. (2015). SemEval-2015 Task 3: Answer selection in community question answering. In *Proceedings of SemEval 2015*.

Nakov, P., Màrquez, L., Moschitti, A., Magdy, W., Mubarak, H., Freihat, a., et al. (2016). SemEval-2016 Task 3: Community question answering. In *Proceedings of SemEval 2016* (pp. 525–545). http://www.aclweb.org/anthology/773S16-1083.

Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence, 193*, 217–250.

Nicosia, M., Filice, S., Barrón-Cedeño, A., Saleh, I., Mubarak, H., Gao, W., et al. (2015). QCRI: Answer Selection for community question answering – Experiments for Arabic and English. In *Proceedings of SemEval 2015* (pp. 203–209).

Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., et al. (2014). MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of LREC 2014* (pp. 1094–1101).

Potthast, M., Barrón-Cedeño, A., Stein, B., & Rosso, P. (2011). Cross-language plagiarism detection. *Language Resources and Evaluation, 45*(1), 45–62. http://dx.doi.org/10.1007/s10579-009-9114-z.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiskỳ, T., & Blunsom, P. (2016). Reasoning about entailment with neural attention. *International conference on learning representations*.

Romeo, S., Da San Martino, G., Barrón-Cedeño, A., Moschitti, A., Belinkov, Y., Hsu, W.-N., et al. (2016). Neural attention for learning to rank questions in community question answering. In *Proceedings of COLING 2016* (pp. 1734–1745).

Rosso, P., Lyhyaoui, A., Peñarrubia, J., Montes y Gómez, M., Benajiba, Y., & Raissouni, N. (2005). Arabic–English question answering. In *Proceedings of information communication technologies international symposium, ICTIS, Tetuan, Morocco*.

Salem, Z., Sadek, J., Chakkour, F., & Haskkour, N. (2010). *Automatically finding answers to "Why" and "How to" questions for Arabic language* (pp. 586–593). Springer.

Schneider, N., Mohit, B., Oflazer, K., & Smith, N. A. (2012). Coarse lexical semantic annotation with supersenses: An Arabic case study. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers – volume 2, ACL 2012* (pp. 253–258). Stroudsburg, PA, USA: Association for Computational Linguistics.

Seddah, D., Tsarfaty, R., Kübler, S., Candito, M., Choi, J., Farkas, R., et al. (2013). Overview of the SPMRL 2013 shared task: Cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of SPMRL '13* (pp. 146–182).

Severyn, A., & Moschitti, A. (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 2012* (pp. 741–750). Portland, OR. doi:10.1145/2348283.2348383.

Silberztein, M. (2005). NooJ: A linguistic annotation system for corpus processing. In *Proceedings of HLT/EMNLP on interactive demonstrations* (pp. 10–11). Association for Computational Linguistics.

Strzalkowski, T., & Harabagiu, S. (Eds.). (2008). *Advances in open domain question answering*. Netherlands: Springer.

Surdeanu, M., Ciaramita, M., & Zaragoza, H. (2008). Learning to rank answers on large online QA collections. In *Proceedings of ACL-HLT 2008* (pp. 719–727).

Tran, Q. H., Tran, V., Vu, T., Nguyen, M., & Bao Pham, S. (2015). JAIST: Combining multiple features for answer selection in community question answering. In *Proceedings of SemEval 2015* (pp. 215–219).

Trigui, O., Belguith, H., & Rosso, P. (2010). DefArabicQA: Arabic definition question answering system. In *Proceedings of 7th workshop on language resources and human language technologies for semitic languages, LREC, Valletta, Malta* (pp. 40–45).

Tymoshenko, K., Bonadiman, D., & Moschitti, A. (2016). Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of Association for Computational Linguistics: Human Language Technologies* (pp. 1268–1278).

Wang, K., Ming, Z., & Chua, T.-S. (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* (pp. 187–194). ACM.

Wise, M. (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 1996* (pp. 130–134). New York, NY. doi:10.1145/236452.236525.

Zhang, K., Wu, W., Wu, H., Li, Z., & Zhou, M. (2014). Question retrieval with high quality answers in community question answering. In *Proceedings of the 23rd ACM international conference on information and knowledge management, CIKM 2014* (pp. 371–380).

Zhou, G., Cai, L., Zhao, J., & Liu, K. (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* (pp. 653–662). Association for Computational Linguistics.

Zhou, G., He, T., Zhao, J., & Hu, P. (2015). Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL-HLT 2015* (pp. 250–259).