Automatic Feature Engineering for Answer Selection and Extraction

Aliaksei Severyn
DISI, University of Trento
38123 Povo (TN), Italy
severyn@disi.unitn.it

Alessandro Moschitti
Qatar Computing Research Institue
5825 Doha, Qatar
amoschitti@qf.org.qa

Abstract

This paper proposes a framework for automatically engineering features for two important tasks of question answering: answer sentence selection and answer extraction. We represent question and answer sentence pairs with linguistic structures enriched by semantic information, where the latter is produced by automatic classifiers, e.g., question classifier and Named Entity Recognizer. Tree kernels applied to such structures enable a simple way to generate highly discriminative structural features that combine syntactic and semantic information encoded in the input trees. We conduct experiments on a public benchmark from TREC to compare with previous systems for answer sentence selection and answer extraction. The results show that our models greatly improve on the state of the art, e.g., up to 22% on F1 (relative improvement) for answer extraction, while using no additional resources and no manual feature engineering.

1 Introduction

Question Answering (QA) systems are typically built from three main macro-modules: (i) search and retrieval of candidate passages; (ii) reranking or selection of the most promising passages; and (iii) answer extraction. The last two steps are the most interesting from a Natural Language Processing viewpoint since deep linguistic analysis can be carried out as the input is just a limited set of candidates.

Answer sentence selection refers to the task of selecting the sentence containing the correct answer among the different sentence candidates retrieved by a search engine.

Answer extraction is a final step, required for factoid questions, consisting in extracting multiwords constituting the synthetic answer, e.g., *Barack Obama* for a question: *Who is the US president?* The definition of rules for both tasks is conceptually demanding and involves the use of syntactic and semantic properties of the questions and its related answer passages.

For example, given a question from TREC QA[1]:

> *Q: What was Johnny Appleseed's real name?*

and a relevant passage, e.g., retrieved by a search engine:

> *A: Appleseed, whose real name was John Chapman, planted many trees in the early 1800s.*

a rule detecting the semantic links between *Johnny Appleseed's real name* and the correct answer *John Chapman* in the answer sentence has to be engineered. This requires the definition of other rules that associate the question pattern `real_name_?(X)` with `real_name_is(X)` of the answer sentence. Although this can be done by an expert NLP engineer, the effort for achieving the necessary coverage and a reasonable accuracy is not negligible.

An alternative to manual rule definition is the use of machine learning, which often shifts the problem

[1] We use it as our running example in the rest of the paper.

to the easier task of feature engineering. Unfortunately, when the learning task is semantically difficult such as in QA, e.g., features have to encode combinations of syntactic and semantic properties. Thus their extraction modules basically assume the shape of high-level rules, which are, in any case, essential to achieve state-of-the-art accuracy. For example, the great IBM Watson system (Ferrucci et al., 2010) uses a learning to rank algorithm fed with hundreds of features. The extraction of some of the latter requires articulated rules/algorithms, which, in terms of complexity, are very similar to those constituting typical handcrafted QA systems. An immediate consequence is the reduced adaptability to new domains, which requires a substantial re-engineering work.

In this paper, we show that tree kernels (Collins and Duffy, 2002; Moschitti, 2006) can be applied to automatically learn complex structural patterns for both answer sentence selection and answer extraction. Such patterns are syntactic/semantic structures occurring in question and answer passages. To make such information available to the tree kernel functions, we rely on the shallow syntactic trees enriched with semantic information (Severyn et al., 2013b; Severyn et al., 2013a), e.g., Named Entities (NEs) and question focus and category, automatically derived by machine learning modules, e.g., question classifier (QC) or focus classifier (FC).

More in detail, we (i) design a pair of shallow syntactic trees (one for the question and one for the answer sentence); (ii) connect them with relational nodes (i.e., those matching the same words in the question and in the answer passages); (iii) label the tree nodes with semantic information such as question category and focus and NEs; and (iv) use the NE type to establish additional semantic links between the candidate answer, i.e., an NE, and the focus word of the question. Finally, for the task of answer extraction we also connect such semantic information to the answer sentence trees such that we can learn factoid answer patterns.

We show that our models are very effective in producing features for both answer selection and extraction by experimenting with TREC QA corpora and directly comparing with the state of the art, e.g., (Wang et al., 2007; Yao et al., 2013). The results show that our methods greatly improve on both tasks yielding a large improvement in Mean Average Precision for answer selection and in F1 for answer extraction: up to 22% of relative improvement in F1, when small training data is used. Moreover, in contrast to the previous work, our model does not rely on external resources, e.g., WordNet, or complex features in addition to the structural kernel model.

The reminder of this paper is organized as follows, Sec. 2 describes our kernel-based classifiers, Sec. 3 illustrates our question/answer relational structures also enriched with semantic information, Sec. 4 describes our model for answer selection and extraction, Sec. 5 illustrates our comparative experiments on TREC data, Sec. 6 reports on our error analysis, Sec. 7 discusses the related work, and finally, Sec. 8 derives the conclusions.

2 Structural Kernels for classification

This section describes a kernel framework where the input question/answer pairs are handled directly in the form of syntactic/semantic structures.

2.1 Feature vector approach to object pair classification

A conventional approach to represent a question/answer pairs in linear models consists in defining a set of similarity features $\{x_i\}$ and computing the simple scalar product $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_i w_i x_i$, where $\boldsymbol{w}$ is the model weight vector learned on the training data. Hence, the learning problem boils down to estimating individual weights of each of the similarity features x_i. Such features often encode various types of lexical, syntactic and semantic similarities shared between a question and its candidate. Previous work used a rich number of distributional semantic, knowledge-based, translation and paraphrase resources to build explicit feature vector representations. One evident potential downside of using feature vectors is that a great deal of structural information encoded in a given text pair is lost.

2.2 Pair Classification using Structural Kernels

A more versatile approach in terms of the input representation relies on kernels. A typical kernel machine, e.g., SVM, classifies a test input $\boldsymbol{x}$ using the following prediction function: $h(\boldsymbol{x}) = \sum_i \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i)$, where α_i are the model parameters estimated from the training data, y_i are target variables, $\boldsymbol{x}_i$ are support vectors, and $K(\cdot,\cdot)$ is a kernel function. The latter can measure the *similarity* between question and answer pairs.

We define each question/answer pair $\boldsymbol{x}$ as a triple composed of a question tree $\boldsymbol{T}_q$ and answer sentence tree $\boldsymbol{T}_s$ and a similarity feature vector $\boldsymbol{v}$, i.e., $\boldsymbol{x} = \langle \boldsymbol{T}_q, \boldsymbol{T}_s, \boldsymbol{v} \rangle$. Given two triples $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$, we define the following kernel:

$$\begin{aligned} K(\boldsymbol{x}^i, \boldsymbol{x}^j) &= K_{\text{TK}}(\boldsymbol{T}_q^i, \boldsymbol{T}_q^j) \\ &+ K_{\text{TK}}(\boldsymbol{T}_s^i, \boldsymbol{T}_s^j) \\ &+ K_{\text{v}}(\boldsymbol{v}^i, \boldsymbol{v}^j), \end{aligned} \quad (1)$$

where K_{TK} computes a structural kernel, e.g., tree kernel, and K_{v} is a kernel over feature vectors, e.g., linear, polynomial, gaussian, etc. Structural kernels can capture the structural representation of a question/answer pair whereas traditional feature vectors can encode some sort of similarity, e.g., lexical, syntactic, semantic, between a question and its candidate answer.

We prefer to split the kernel computation over a question/answer pair into two terms since tree kernels are very efficient and there are no efficient graph kernels that can encode exhaustively all graph fragments. It should be noted that the tree kernel sum does not capture feature pairs. Theoretically, for such purpose, a kernel product should be used. However, our experiments revealed that using the product is actually worse in practice. In contrast, we solve the lack of feature pairing by annotating the trees with relational tags which are supposed to link the question tree fragments with the related fragments from the answer sentence.

Such relational information is very important to improve the quality of the pair representation as well as the implicitly generated features. In the next section, we show simple structural models that we used in our experiments for question and answer pair classification.

2.3 Partial Tree Kernels

The above framework can use any kernel for structural data. We use the Partial Tree Kernel (PTK) (Moschitti, 2006) to compute $K_{\text{TK}}(\cdot,\cdot)$ as it is the most general convolution tree kernel, which at the same time shows rather *good* efficiency. PTK can be effectively applied to both constituency and dependency parse trees. It generalizes the syntactic tree kernel (STK) (Collins and Duffy, 2002), which maps a tree into the space of all possible tree fragments constrained by the rule that sibling nodes cannot be separated. In contrast, the PTK fragments can contain any subset of siblings, i.e., PTK allows for breaking the production rules in syntactic trees. Consequently, PTK generates an extremely rich feature space, which results in higher generalization ability.

3 Relational Structures

This section introduces relational structures designed to encode syntactic and shallow semantic properties of question/answer pairs. We first define a simple to construct shallow syntactic tree representation derived from a shallow parser. Next, we introduce a relational linking scheme based on a plain syntactic matching and further augment it with additional semantic information.

3.1 Shallow syntactic tree

Our shallow tree structure is a two-level syntactic hierarchy built from word lemmas (leaves), part-of-speech tags that organized into chunks identified by a shallow syntactic parser (Fig. 1). We defined a similar structure in (Severyn and Moschitti, 2012) for answer passage reranking, which improved on feature vector baselines.

This simple linguistic representation is suitable for building a rather expressive *answer sentence selection* model. Moreover, the use of a shallow parser is motivated by the need to generate text spans to produce candidate answers required by an *answer extraction* system.

3.2 Tree pairs enriched with relational links

It is important to establish a correspondence between question and answer sentence aligning related concepts from both. We take on a two-level approach, where we first use plain lexical matching to connect common lemmas from the question and its candidate answer sentence. Secondly, we establish semantic links between NEs extracted from the answer sentence and the question focus word, which encodes the expected lexical answer type (LAT). We use the question categories to identify NEs that have

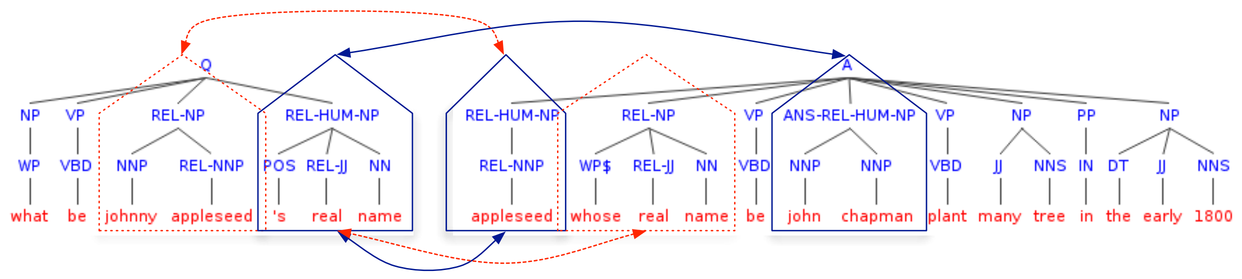

Figure 1: Shallow tree representation of the example q/a pair from Sec. 1. Dashed arrows (red) indicate the tree fragments (red dashed boxes) in the question and its answer sentence linked by the relational `REL` tag, which is established via syntactic match on the word lemmas. Solid arrows (blue) connect a question focus word *name* with the related named entities of type *Person* corresponding to the question category (HUM) via a relational tag `REL-HUM`. Additional `ANS` tag is used to mark chunks containing candidate answer (here the correct answer *John Chapman*).

higher probability to be correct answers following a mapping defined in Table 1.

Next, we briefly introduce our tree kernel-based models for building question focus and category classifiers.

Lexical Answer Type. Question Focus represents a central entity or a property asked by a question (Prager, 2006). It can be used to search for semantically compatible candidate answers, thus greatly reducing the search space (Pinchak, 2006). While several machine learning approaches based on manual features and syntactic structures have been recently explored, e.g. (Quarteroni et al., 2012; Damljanovic et al., 2010; Bunescu and Huang, 2010), we opt for the latter approach where tree kernels handle automatic feature engineering.

To build an automatic Question Focus detector we use a tree kernel approach as follows: we (i) parse each question; (ii) create a set of positive trees by labeling the node exactly covering the focus with FC tag; (iii) build a set of negative trees by labeling any other constituent node with FC; (iii) we train the FC node classifier with tree kernels. At the test time, we try to label each constituent node with FC generating a set of candidate trees. Finally, we select the tree and thus the constituent associated with the highest SVM score.

Question classification. Our question classification model is simpler than before: we use an SVM multi-classifier with tree kernels to automatically extract the question class. To build a multi-class classifier we train a binary SVM for each of the classes and apply a one-vs-all strategy to obtain the predicted class. We use constituency trees as our input representation.

Table 1: Expected Answer Type (EAT) $\rightarrow$ named entity types.

EAT	Named Entity types
HUM	Person
LOCATION	Location
ENTITY	Organization, Person, Misc
DATE	Date, Time, Number
QUANTITY	Number, Percentage
CURRENCY	Money, Number

Our question taxonomy is derived from the UIUIC dataset (Li and Roth, 2002) which defines 6 coarse and 50 fine grain classes. In particular, our set of question categories is formed by adopting 3 coarse classes: HUM (human), LOC (location), ENTY (entities) and replacing the NUM (numeric) coarse class with 3 fine-grain classes: CURRENCY, DATE, QUANTITY[2]. This set of question categories is sufficient to capture the coarse semantic answer type of the candidate answers found in TREC. Also using fewer question classes results in a more accurate multi-class classifier.

Semantic tagging. Question focus word specifies the lexical answer type capturing the target information need posed by a question, but to make this piece of information effective, the focus word needs to be linked to the target candidate answer. The focus word can be lexically matched with words present in

[2]This class is composed by including all the fine-grain classes from NUMERIC coarse class except for CURRENCY and DATE.

the answer sentence, or the match can be established using semantic information. Clearly, the latter approach is more appealing since it helps to alleviate the lexical gap problem, i.e., it improves the coverage of the näive string matching of words between a question and its answer.

Hence, we propose to exploit a question focus along with the related named entities (according to the mapping from Table 1) of the answer sentence to establish relational links between the tree fragments. In particular, once the question focus and question category are determined, we link the focus word w_{focus} in the question, with all the named entities whose type matches the question class (Table 1). We perform tagging at the chunk level and use a relational tag typed with a question class, e.g., `REL-HUM`. Fig. 1 shows an example q/a pair where the typed relational tag is used in the shallow syntactic tree representation to link the chunk containing the question focus *name* with the named entities of the corresponding type *Person*, i.e., *Appleseed* and *John Chapman*.

4 Answer Sentence Selection and Answer Keyword Extraction

This section describes our approach to (i) answer sentence selection used to select the most promising answer sentences; and (ii) answer extraction which returns the answer keyword (for factoid questions).

4.1 Answer Sentence Selection

We cast the task of answer sentence selection as a classification problem. Considering a supervised learning scenario, we are given a set of questions $\{q_i\}_{i=1}^N$ where each question q_i is associated with a list of candidate answer sentences $\{(r_i, s_i)\}_{i=1}^N$, with $r_i \in \{-1, +1\}$ indicating if a given candidate answer sentence s_i contains a correct answer $(+1)$ or not (-1). Using this labeled data, our goal is to learn a classifier model to predict if a given pair of a question and an answer sentence is correct or not. We train a binary SVM with tree kernels[3] to train an answer sentence classifier. The prediction scores obtained from a classifier are used to rerank the answer candidates (pointwise reranking), s.t. the sentences that are more likely to contain correct answers will be ranked higher than incorrect candidates. In addition to the structural representation, we augment our model with basic bag-of-word features (unigram and bigrams) computed over lemmas.

[3]disi.unitn.it/moschitti/Tree-Kernel.htm

4.2 Answer Sentence Extraction

The goal of answer extraction is to extract a text span from a given candidate answer sentence. Such span represents a correct answer phrase for a given question. Different from previous work that casts the answer extraction task as a tagging problem and apply a CRF to learn an answer phrase tagger (Yao et al., 2013), we take on a simpler approach using a kernel-based classifier.

In particular, we rely on the shallow tree representation, where text spans identified by a shallow syntactic parser serve as a source of candidate answers. Algorithm 1 specifies the steps to generate training data for our classifier. In particular, for each example representing a triple $\langle a, T_q, T_s \rangle$ composed of the answer a, the question and the answer sentence trees, we generate a set of training examples E with every candidate chunk marked with an `ANS` tag (one at a time). To reduce the number of generated examples for each answer sentence, we only consider `NP` chunks, since other types of chunks, e.g., `VP`, `ADJP`, typically do not contain factoid answers. Finally, an original untagged tree is used to generate a positive example (line 8), when the answer sentence contains a correct answer, and a negative example (line 10), when it does not contain a correct answer.

At the classification time, given a question and a candidate answer sentence, all NP nodes of the sentence are marked with `ANS` (one at a time) as the possible answer, generating a set of tree candidates. Then, such trees are classified (using the kernel from Eq. 1) and the one with the highest score is selected. If no tree is classified as positive example we do not extract any answer.

5 Experiments

We provide the results on two related yet different tasks: answer sentence selection and answer extraction. The goal of the former is to learn a model scoring correct question and answer sentence pairs to bring in the top positions sentences containing the correct answers. Answer extraction derives the cor-

Algorithm 1 Generate training data for answer extraction

```
1: for all $\langle a, T_q, T_s \rangle \in \boldsymbol{D}$ do
2:     $E \leftarrow \emptyset$
3:     for all $chunk \in extract\_chunks(T_s)$ do
4:         if not $chunk ==$ NP then
5:             continue
6:         $T'_s \leftarrow tagAnswerChunk(T_s, chunk)$
7:         if $contains\_answer(a, chunk)$ then
8:             label $\leftarrow +1$
9:         else
10:            label $\leftarrow -1$
11:        $e \leftarrow build\_example(T_q, T'_s, label)$
12:        $E \leftarrow E \cup \{e\}$
13: return $E$
```

rect answer keywords, i.e., a text span such as multiwords or constituents, from a given sentence.

5.1 Semantic Annotation

We briefly describe the experiments of training automatic question category and focus classifiers, which are more extensively described in (Severyn et al., 2013b).

Question Focus detection. We used three datasets for training and evaluating the performance of our focus detector: SeCo-600 (Quarteroni et al., 2012), Mooney GeoQuery (Damljanovic et al., 2010) and the dataset from (Bunescu and Huang, 2010). The SeCo dataset contains 600 questions. The Mooney GeoQuery contains 250 question targeted at geographical information in the U.S. The first two datasets are very domain specific, while the dataset from (Bunescu and Huang, 2010) is more generic containing the first 2,000 questions from the answer type dataset from Li and Roth annotated with focus words. We removed questions with implicit and multiple focuses.

Question Classification. We used the UIUIC dataset (Li and Roth, 2002) which contains 5,952 factoid questions [4] to train a multi-class question classifier.

Table 2 summarizes the results of question focus and category classification.

[4]We excluded questions from TREC to ensure there is no overlap with the data used for testing models trained on TREC QA.

Table 2: Accuracy (%) of focus (FC) and question classifiers (QC) using PTK.

TASK	SET	PTK
FC	MOONEY	80.5
	SECO-600	90.0
	BUNESCU	96.9
QC	UIUIC	85.9
	TREC 11-12	78.1

5.2 Answer Sentence Selection

We used the train and test data from (Wang et al., 2007) to enable direct comparison with previous work on answer sentence selection. The training data is composed by questions drawn from TREC 8-12 while questions from TREC 13 are used for testing. The data provided for training comes as two sets: a small set of 94 questions (TRAIN) that were manually curated for errors[5] and 1,229 questions from the entire TREC 8-12 that contain at least one correct answer sentence (ALL). The latter set represents a more noisy setting, since many answer sentences are marked erroneously as correct as they simply match a regular expression. Table 3 summarizes the data used for training and testing.

Table 4 compares our kernel-based structural model with the previous state-of-the-art systems for answer sentence selection. In particular, we compare with four most recent state of the art answer sentence reranker models (Wang et al., 2007; Heilman and Smith, 2010; Wang and Manning, 2010; Yao et al., 2013), which report their performance on the same questions and candidate sets from TREC 13 as provided by (Wang et al., 2007).

Our simple shallow tree representation (Severyn and Moschitti, 2012) delivers state-of-the-art accuracy largely improving on previous work. Finally, augmenting the structure with semantic linking (Severyn et al., 2013b) yields additional improvement in MAP and MRR. This suggests the utility of using supervised components, e.g., question focus and question category classifiers coupled with NERs, to establish semantic mapping between words in a q/a pair.

[5]In TREC correct answers are identified by regex matching using the provided answer pattern files

Table 3: Summary of TREC data for answer extraction used in (Yao et al., 2013).

data	questions	candidates	correct
TRAIN	94	4718	348
ALL	1229	53417	6410
TEST	89	1517	284

Table 4: Answer sentence reranking on TREC 13.

System	MAP	MRR
Wang et al. (2007)	0.6029	0.6852
Heilman & Smith (2010)	0.6091	0.6917
Wang & Manning (2010)	0.5951	0.6951
Yao et al. (2013)	0.6319	0.7270
+ WN	0.6371	0.7301
shallow tree (S&M, 2012)	0.6485	0.7244
+ semantic tagging	**0.6781**	**0.7358**

It is worth noting that our kernel-based classifier is conceptually simpler than approaches in the previous work, as it relies on the structural kernels, e.g., PTK, to automatically extract salient syntactic patterns relating questions and answers. Our model only includes the most basic feature vector (uni- and bi-grams) and does not rely on external sources such as WordNet.

5.3 Answer Extraction

Our experiments on answer extraction replicate the setting of (Yao et al., 2013), which is the most recent work on answer extraction reporting state-of-the-art results.

Table 5 reports the accuracy of our model in recovering correct answers from a set of candidate answer sentences for a given question. Here the focus is on the ability of an answer extraction system to recuperate as many correct answers as possible from each answer sentence candidate. The set of extracted candidate answers can then be used to select a single best answer, which is the final output of the QA system for factoid questions. Recall (R) encodes the percentage of correct answer sentences for which the system correctly extracts an answer (for TREC 13 there are a total of 284 correct answer sentences), while Precision (P) reflects how many answers extracted by the system are actually correct. Clearly, having a high recall system, allows for correctly answering more questions. On the other hand, a high precision system would attempt to answer less questions (extracting no answers at all) but get them right.

We compare our results to a CRF model of (Yao et al., 2013) augmented with WordNet features (without forced voting) [6]. Unlike the CRF model which obtains higher values of precision, our system acts as a high recall system able to recover most of the answers from the correct answer sentences. Having higher recall is favorable to high precision in answer extraction since producing more correct answers can help in the final voting scheme to come up with a single best answer. To solve the low recall problem of their CRF model, Yao et al. (2013) apply fairly complex outlier resolution techniques to force answer predictions, thus aiming at increasing the number of extracted answers.

To further boost the number of answers produced by our system we exclude negative examples (answer sentences not containing the correct answer) from training, which slightly increases the number of pairs with correctly recovered answers. Nevertheless, it has a substantial effect on the number of questions that can be answered correctly (assuming perfect single best answer selection). Clearly, our system is able to recover a large number of answers from the correct answer sentences, while low precision, i.e., extracting answer candidates from sentences that do not contain a correct answer, can be overcome by further applying various best answer selection strategies, which we explore in the next section.

5.4 Best Answer Selection

Since the final step of the answer extraction module is to select for each question a single best answer from a set of extracted candidate answers, an answer selection scheme is required.

We adopt a simple majority voting strategy, where we aggregate the extracted answers produced by our answer extraction model. Answers sharing similar lemmas (excluding stop words) are grouped together. The prediction scores obtained by the an-

[6] We could not replicate the results obtained in (Yao et al., 2013) with the forced voting strategy. Thus such result is not included in Table 5.

Table 5: Results on answer extraction. P/R - precision and recall; pairs - number of QA pairs with a correctly extracted answer, q - number of questions with at least one correct answer extracted, F1 sets an upper bound on the performance assuming the selected best answer among extracted candidates is always correct. *-marks the setting where we exclude incorrect question answer pairs from training.

set	P	R	pairs	q	F1
Yao et al. (2013)	25.7	23.4	73	33	-
+ WN	26.7	24.3	76	35	-
TRAIN	29.6	64.4	183	58	65.2
TRAIN*	15.7	71.8	204	66	74.1
Yao et al. (2013)	35.2	35.1	100	38	-
+ WN	34.5	34.7	98	38	-
ALL	29.4	74.6	212	69	77.5
ALL*	15.8	76.7	218	73	82.0

Table 6: Results on finding the best answer with voting.

system	set	P	R	F1
Yao et al. (2013)	TRAIN	55.7	43.8	49.1
+ forced		54.5	53.9	54.2
+ WN		55.2	53.9	54.5
this work		**66.2**	**66.2**	**66.2**
Yao et al. (2013)	ALL	67.2	50.6	57.7
+ forced		60.9	59.6	60.2
+ WN		63.6	62.9	63.3
this work		**70.8**	**70.8**	**70.8**

swer extraction classifier are used as votes to decide on the final rank to select the best single answer.

Table 6 shows the results after the majority voting is applied to select a single best answer for each candidate. A rather naïve majority voting scheme already produces satisfactory outcome demonstrating better results than the previous work. Our voting scheme is similar to the one used by (Yao et al., 2013), yet it is much simpler since we do not perform any additional hand tuning to account for the weight of the "forced" votes or take any additional steps to catch additional answers using outlier detection techniques applied in the previous work.

6 Discussion and Error Analysis

There are several sources of errors affecting the final performance of our answer extraction system: (i) chunking, (ii) named entity recognition and semantic linking, (iii) answer extraction, (iv) single best answer selection.

Chunking. Our system uses text spans identified by a chunker to extract answer candidates, which makes it impossible to extract answers that lie outside the chunk boundaries. Nevertheless, we found this to be a minor concern since for 279 out of total 284 candidate sentences from TREC 13 the answers are recoverable within the chunk spans.

Semantic linking. Our structural model relies heavily on the ability of NER to identify the relevant entities in the candidate sentence that can be further linked to the focus word of the question. While our answer extraction model is working on all the NP chunks, the semantic tags from NER serve as a strong cue for the classifier that a given chunk has a high probability of containing an answer. Typical off-the-shelf NER taggers have good precision and low recall, s.t. many entities as potential answers are missed. In this respect, a high recall entity linking system, e.g., linking to wikipedia entities (Ratinov et al., 2011), is required to boost the quality of candidates considered for answer extraction. Finally, improving the accuracy of question and focus classifiers would allow for having more accurate input representations fed to the learning algorithm.

Answer Extraction. Our answer extraction model acts as a high recall system, while it suffers from low precision in extracting answers for many incorrect sentences. Improving the precision without sacrificing the recall would ease the successive task of best answer selection, since having less incorrect answer candidates would result in a better final performance. Introducing additional constraints in the form of semantic tags to allow for better selection of answer candidates could also improve our system.

Best Answer Selection. We apply a naïve majority voting scheme to select a single best answer from a set of extracted answer candidates. This step has a dramatic impact on the final performance of the answer extraction system resulting in a large drop of recall, i.e., from 82.0 to 70.8 before and after voting respectively. Hence, a more involved model, i.e., performing joint answer sentence re-ranking and answer extraction, is required to yield a better performance.

7 Related Work

Tree kernel methods have found many applications for the task of answer reranking which are reported in (Moschitti, 2008; Moschitti, 2009; Moschitti and Quarteroni, 2008; Severyn and Moschitti, 2012). However, their methods lack the use of important relational information between a question and a candidate answer, which is essential to learn accurate relational patterns. In this respect, a solution based on enumerating relational links was given in (Zanzotto and Moschitti, 2006; Zanzotto et al., 2009) for the textual entailment task but it is computationally too expensive for the large dataset of QA. A few solutions to overcome computational issues were suggested in (Zanzotto et al., 2010).

In contrast, this paper relies on structures directly encoding the output of question and focus classifiers to connect focus word and good candidate answer keywords (represented by NEs) of the answer passage. This provides more effective relational information, which allows our model to significantly improve on previous rerankers. Additionally, previous work on kernel-based approaches does not target answer extraction.

One of the best models for answer sentence selection has been proposed in (Wang et al., 2007). They use the paradigm of quasi-synchronous grammar to model relations between a question and a candidate answer with syntactic transformations. (Heilman and Smith, 2010) develop an improved Tree Edit Distance (TED) model for learning tree transformations in a q/a pair. They search for a good sequence of tree edit operations using complex and computationally expensive Tree Kernel-based heuristic. (Wang and Manning, 2010) develop a probabilistic model to learn tree-edit operations on dependency parse trees. They cast the problem into the framework of structured output learning with latent variables. The model of (Yao et al., 2013) has reported an improvement over the Wang's et al. (2007) system. It applies linear chain CRFs with features derived from TED and WordNet to automatically learn associations between questions and candidate answers.

Different from previous approaches that use tree-edit information derived from syntactic trees, our kernel-based learning approach also use tree structures but with rather different learning methods, i.e., SVMs and structural kernels, to automatically extract salient syntactic patterns relating questions and answers. In (Severyn et al., 2013c), we have shown that such relational structures encoding input text pairs can be directly used within the kernel learning framework to build state-of-the-art models for predicting semantic textual similarity. Furthermore, semantically enriched relational structures, where automatic have been previously explored for answer passage reranking in (Severyn et al., 2013b; Severyn et al., 2013a). This paper demonstrates that this model also works for building a reranker on the sentence level, and extends the previous work by applying the idea of automatic feature engineering with tree kernels to answer extraction.

8 Conclusions

Our paper demonstrates the effectiveness of handling the input structures representing QA pairs directly vs. using explicit feature vector representations, which typically require substantial feature engineering effort. Our approach relies on a kernel-based learning framework, where structural kernels, e.g., tree kernels, are used to handle automatic feature engineering. It is enough to specify the desired type of structures, e.g., shallow, constituency, dependency trees, representing question and its candidate answer sentences and let the kernel learning framework learn to use discriminative tree fragments for the target task.

An important feature of our approach is that it can effectively combine together different types of syntactic and semantic information, also generated by additional automatic classifiers, e.g., focus and question classifiers. We augment the basic structures with additional relational and semantic information by introducing special tag markers into the tree nodes. Using the structures directly in the kernel learning framework makes it easy to integrate additional relational constraints and semantic information directly in the structures.

The comparison with previous work on a public benchmark from TREC suggests that our approach is very promising as we can improve the state of the art in both answer selection and extraction by a large margin (up to 22% of relative improvement in F1 for answer extraction). Our approach makes it relatively easy to integrate other sources of semantic information, among which the use of Linked Open Data can be the most promising to enrich the structural representation of q/a pairs.

To achieve state-of-the-art results in answer sentence selection and answer extraction, it is sufficient to provide our model with a suitable tree structure encoding relevant syntactic information, e.g., using shallow, constituency or dependency formalisms. Moreover, additional semantic and relational information can be easily plugged in by marking tree nodes with special tags. We believe this approach greatly eases the task of tedious feature engineering that will find its applications well beyond QA tasks.

Acknowledgements

This research is partially supported by the EU's 7th Framework Program (FP7/2007-2013) (#288024 LIMOSINE project) and an Open Collaborative Research (OCR) award from IBM Research. The first author is supported by the Google Europe Fellowship 2013 award in Machine Learning.

References

Razvan Bunescu and Yunfeng Huang. 2010. Towards a general model of answer typing: Question focus identification. In *CICLing*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *ACL*.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. 2010. Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In *LREC*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3).

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

A. Moschitti and S. Quarteroni. 2008. Kernels on Linguistic Structures for Answer Extraction. In *ACL*.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *CIKM*.

Alessandro Moschitti. 2009. Syntactic and semantic kernels for short text pair categorization. In *EACL*.

Christopher Pinchak. 2006. A probabilistic answer type model. In *In EACL*.

John M. Prager. 2006. Open-domain question-answering. *Foundations and Trends in Information Retrieval*, 1(2):91–231.

Silvia Quarteroni, Vincenzo Guerrisi, and Pietro La Torre. 2012. Evaluating multi-focus natural language queries over data services. In *LREC*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Building structures from classifiers for passage reranking. In *CIKM*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning adaptable patterns for passage reranking. In *CoNLL*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013c. Learning semantic textual similarity with structural representations. In *ACL*.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *ACL*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitaura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP*.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

F. M. Zanzotto and A. Moschitti. 2006. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *The Joint 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia. Association for Computational Linguistics.

F. M. Zanzotto, M. Pennacchiotti, and A. Moschitti. 2009. A Machine Learning Approach to Recognizing Textual Entailment. *Natural Language Engineering*, Volume 15 Issue 4, October 2009:551–582.

F. M. Zanzotto, L. Dell'Arciprete, and A. Moschitti. 2010. Efficient graph kernels for textual entailment recognition. *FUNDAMENTA INFORMATICAE*, 2010.