Building Structures from Classifiers for Passage Reranking

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

Massimo Nicosia
DISI
University of Trento
38123 Povo (TN), Italy
m.nicosia@disi.unitn.it

Alessandro Moschitti
QCRI
Qatar Foundation
Doha, Qatar
amoschitti@qf.org.qa

ABSTRACT

This paper shows that learning to rank models can be applied to automatically learn complex patterns, such as relational semantic structures occurring in questions and their answer passages. This is achieved by providing the learning algorithm with a tree representation derived from the syntactic trees of questions and passages connected by relational tags, where the latter are again provided by the means of automatic classifiers, i.e., question and focus classifiers and Named Entity Recognizers. This way effective structural relational patterns are implicitly encoded in the representation and can be automatically utilized by powerful machine learning models such as kernel methods.

We conduct an extensive experimental evaluation of our models on well-known benchmarks from the question answer (QA) track of TREC challenges. The comparison with state-of-the-art systems and BM25 show a relative improvement in MAP of more than 14% and 45%, respectively. Further comparison on the task restricted to the answer sentence reranking shows an improvement in MAP of more than 8% over the state of the art.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: [Language parsing and understanding, Text analysis]

General Terms

Algorithms, Experimentation

Keywords

Question Answering; Learning to Rank; Kernel Methods; Structural Kernels

1. INTRODUCTION

Automated Question Answering (QA) is a complex task that often requires manual definition of rules and syntactic patterns to detect the relations between a question and its candidate answers in text fragments. Simple heuristics just refer to computing a textual similarity between the question and one of its answer passages but the most accurate method is to manually design specific rules that are triggered when patterns in the question and in the passage are found. Such rules are based on syntactic and semantic patterns. For example, given a question[1]:

> *What is Mark Twain's real name?*

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

> *Samuel Langhorne Clemens, better known as Mark Twain.*

the QA engineer usually applies a syntactic parser to obtain the parse trees of the above two sentences, e.g., like those in the top of Fig. 3. Then she/he derives rules like:

> **if** *the pattern* "`What is` NP_2's ADJ `name`" *is in the question* **and** *the pattern* "NP_1 `better known as` NP_2" *is in the answer passage* **then** *associate the passage with a high score*[2],

where the NPs are noun phrases and ADJ is an adjectival phrase recognized by the syntactic parser.

Previous work, e.g., carried out in TREC[3] [46, 47, 48], has shown that such an approach can lead to the design of accurate systems. However, it suffers from two major drawbacks: (i) being based on heuristics it does not provide a definitive methodology, since natural language is too complex to be characterized by a finite set of rules; and (ii) given the latter claim, new domains and languages require the definition of new specific rules, which in turn require a large engineering effort.

An alternative to manual rule definition is the use of machine learning, which often shifts the problem to the easier task of feature engineering. This is very convenient for simple text categorization problems, such as document topic classification, where simple *bag-of-words* models have been shown very effective, e.g., [21]. Unfortunately, when the learning task is more complex such as in QA, features have to encode the combination of syntactic and semantic properties, which basically assume the shape of high-level rules: these are essential to achieve state-of-the-art accuracy. For example, the famous IBM Watson system [14] also uses a

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505688.

[1]We use this question/answer passage (q/a) pair from TREC QA as a running example in the rest of the paper.
[2]If the point-wise answer is needed rather than the entire passage, the rule could end with: **returns** NP_1
[3]`http://trec.nist.gov/data/qamain.html`

learning to rank algorithm fed with hundreds of features. For the extraction of some of the latter, articulated rules are required, which are very similar to those constituting typical manually engineered QA systems.

In this paper, we show that learning to rank models can be applied to automatically learn structural patterns. These are relational structures occurring in question and answer passages and are based on the semantic information automatically derived by additional automatic classification modules. In particular, we first derive a representation of the question and answer passage (q/a) pair, where we follow our approach in [39] by engineering a pair of shallow syntactic trees connected with relational nodes (i.e., those matching the same words in the question and in the answer passages).

Secondly, we include a large sample of basic and advanced features that represent a strong baseline model for answer passage reranking. Additionally, we explore various methods in addressing the "lexical gap" problem: words in a question and its candidate answer can be semantically similar but having different surface forms. To establish relational links between a question and an answer, we exploit WordNet hierarchy [25], LDA topic models [9, 20], SuperSense tagging [10] and word alignments from translation models, e.g., [38, 43]. Their failure in improving our structural representation provides a strong indication that word generalization is not sufficient to improve on strong statistical-based retrieval systems, and more principled linking strategy is required.

To enable a more principled linking strategy, we model and implement question and focus classifiers based on kernel methods. Then, we use the output of such classifiers together with a named entity recognizer (NER) to establish relational links [40]. The focus classifier determines the constituent of the question to be linked to the named entities (NEs) of the answer passage. The target NEs are selected based on the compatibility of their category and the category of the question, e.g., an NE of type PERSON is compatible with a category of a question asking for a HUMAN.

Next, we provide extensive experiments on TREC QA combining our models with traditional feature vectors and the improved relational structures. The results show that our methods greatly improve over previous state-of-the-art QA systems for answer sentence selection. In particular, differently from previous work [39], our models can effectively use NERs and the output of different automatic modules. These improve the system in [39] by more than 14% in MAP and also provide promising directions for fully exploiting semantic resources.

We show that highly discriminative features can be, in fact, automatically extracted and learned by using our kernel learning framework. Our approach does not require manual feature engineering to represent input structures. We do not fully rely on traditional similarity features that encode the degree of similarity between a question and its answer. We treat the input q/a pairs directly encoding them into linguistic trees. More powerful features can be encoded by injecting additional semantic information directly into the tree via special tags and additional tree nodes. We believe such way of engineering features is more intuitive and requires less effort, since the final patterns are automatically extracted by expressive tree kernels.

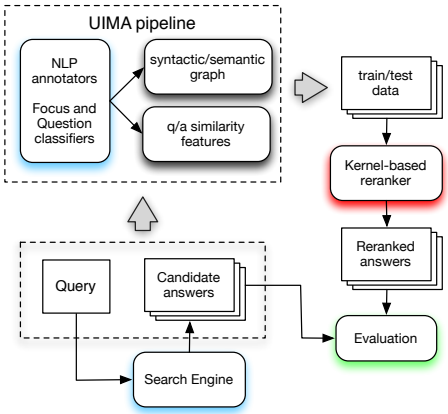


Figure 1: Kernel-based Answer Passage Reranking System

In the reminder of this paper, Section 2 describes our kernel-based reranker, Section 3 illustrates our relational structures along with the classifiers used to generate semantic information. Section 4 reports on the baseline feature vectors used by our rerankers. Section 5 contains answer passage reranking experiments on TREC QA data, while Section 6 compares our models for answer sentence reranking to the current state-of-the-art systems. Section 7 reports on the related work; and finally, Section 8 derives the conclusions.

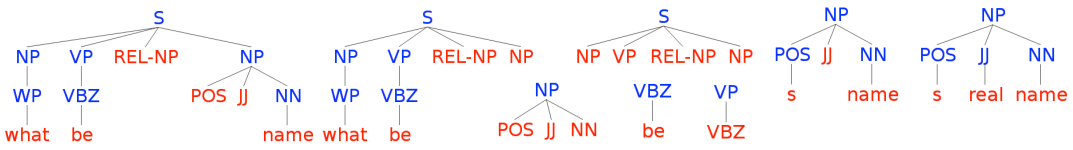
2. LEARNING TO RANK WITH KERNELS

Our QA system is based on a rather simple reranking framework as displayed in Figure 1: given a question q, a search engine retrieves a list of candidate passages ranked by their relevancy. Next, the question together with its candidate answers are processed by a rich natural language processing pipeline that performs basic tokenization, sentence splitting, lemmatization, stopword removal. Various NLP components embedded in the pipeline as UIMA[4] annotators perform more involved linguistic analysis, e.g., part-of-speech tagging, chunking, named entity recognition, constituency and dependency parsing, etc. These annotations are then used to produce structural models (described in Sec. 3), which are further used by a question focus detector and question type classifiers to establish relational links for a given q/a pair. The resulting tree pairs are then used to train a kernel-based reranker, which outputs the model to refine the initial ordering of the retrieved answer passages.

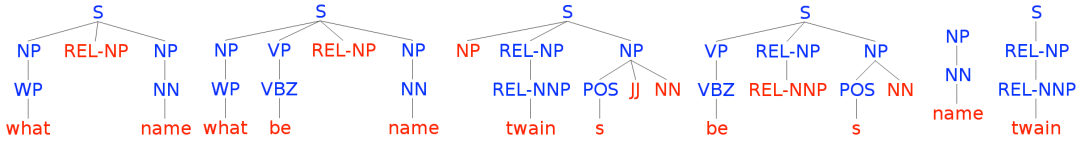
2.1 Tree kernels

As pointed out in the introduction, engineering rules and features is the major bottleneck in the design of a QA system. To tackle this problem, one viable approach is the use of kernel methods (see [27]). A kernel function computes an implicit scalar product between input examples, typically in high dimensional spaces. In our case, they measure similarity between structural objects, e.g., parse trees, in terms of the number of common substructures. Different kernels map objects in different spaces. In this paper, we make use of two types of tree kernels applied to several types of syntactic/semantic structures:

[4] `http://uima.apache.org/`



(a) Some of the fragments generated by STK



(b) Additional fragments only generated by PTK

Figure 2: Some fragments generated by different tree kernels when applied to the question tree of Fig. 3

Syntactic Tree Kernel (STK) also known as a subset tree kernel (SST) [11], which maps objects in the space of all possible tree fragments constrained by the rule that the sibling nodes from their parents cannot be separated. In other words, substructures are composed of atomic building blocks corresponding to nodes along with all of their direct children. These, in case of a syntactic parse tree, are complete production rules of the associated parser grammar. For example, given the parse tree of the question in Fig. 3, some of its syntactic fragments are shown in Fig. 2.(a). It can be noted that each of such fragment can be an important pattern indicating the type of question. For example, the first fragment encodes the first part of the manually designed rule described in the introduction, where `REL` is a relation tag, which is clarified in Sec. 3.

Partial Tree Kernel (PTK) [26] is a function that can be effectively applied to both constituency and dependency parse trees. It generates all possible tree fragments, as above, and sibling nodes can be also separated (so they can be part of different tree fragments). In other words, a fragment is any possible tree path, from whose nodes other tree paths can depart. Consequently, an extremely rich feature space is generated, which results in higher generalization ability. For example, Fig. 2.(b) illustrates some fragments only generated by PTK: such patterns are more general and compact than those provided by STK.

2.2 Preference reranking with kernels

To enable the use of kernels for learning to rank with SVMs, we use preference reranking [22], which reduces the task to binary classification. More specifically, the problem of learning to pick the correct candidate h_i from a candidate set $\{h_1, \ldots, h_k\}$ is reduced to a binary classification problem by creating *pairs*: positive training instances $\langle h_1, h_2\rangle, \ldots, \langle h_1, h_k\rangle$ and negative instances $\langle h_2, h_1\rangle, \ldots, \langle h_k, h_1\rangle$. This set can then be used to train a binary classifier. At classification time the standard one-versus-all binarization method is applied to form all possible pairs of hypotheses. These are ranked according to the number of *classifier votes* they receive: a positive classification of $\langle h_k, h_i\rangle$ gives a vote to h_k whereas a negative one votes for h_i.

A vectorial representation of such pairs is the difference between the vectors representing the hypotheses in a pair. However, this assumes that features are explicit and already available whereas we aim at automatically generating implicit patterns with kernel methods. Thus, for keeping implicit the difference between such vectors we use the following preference kernel:

$$P_K(\langle h_1, h_2\rangle, \langle h'_1, h'_2\rangle) = K(h_1, h'_1) + K(h_2, h'_2) - K(h_1, h'_2) - K(h_2, h'_1), \quad (1)$$

where h_i and h'_i refer to two sets of hypotheses associated with two rankings and K is a kernel applied to pairs of hypotheses. It should be noted that we represent the latter as pairs of question and answer passage trees. More formally, given two hypotheses, $h_i = \langle h_i(q), h_i(a)\rangle$ and $h_i = \langle h'_i(q), h'_i(a)\rangle$, whose members are the question and answer passage trees, we define $K(h_i, h'_i)$ as

$$TK(h_i(q), h'_i(q)) + TK(h_i(a), h'_i(a)),$$

where TK can be any tree kernel function, e.g., STK or PTK. Finally, it should be noted that, to add traditional feature vectors to the reranker, it is enough to sum the product $(\vec{x}_{h_1} - \vec{x}_{h_2}) \cdot (\vec{x}_{h'_1} - \vec{x}_{h'_2})$ to the structural kernel P_K, where $\vec{x}_h$ is the feature vector associated with the hypothesis h.

3. STRUCTURAL MODELS OF Q/A PAIRS

In this section we present our structural models aimed at capturing structural similarities between a question and an answer. We use tree structures as our base representation since they provide sufficient flexibility in representation and allow for easier feature extraction than, for example, graph structures. In addition, trees allow for efficient automatic feature engineering.

3.1 Basic structural representations

We first describe our shallow models that we explored in [39]. Next, we propose models to bridge the lexical gap between the words in the question and in the answer passages, using various sources of semantic annotation.

3.1.1 Shallow tree structures.

Our baseline structural model is the shallow tree representation we first proposed in [39]. This is essentially a shallow syntactic structure built from part-of-speech tags grouped into chunks. Each question and its candidate answer passage are encoded into a tree where part-of-speech tags are found at the pre-terminal level and word lemmas at the leaf level. The sequences of part-of-speech (POS) tags are further organized into chunks. To encode the structural relationship

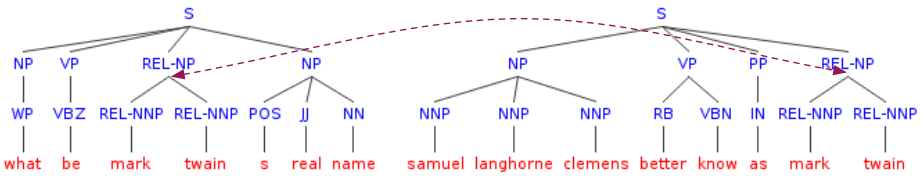


Figure 3: Basic structural representations using a shallow chunk tree for the q/a pair. *Q: What is Mark Twain's real name? A: Samuel Langhorne Clemens, better known as Mark Twain.* Arrows indicate the tree fragments in the question and its answer passage linked by the relational `REL` tag.

for a given q/a pair, a special `REL` tag links the related structures. We adopt a simple strategy to determine such links: lexicals from a question and an answer that have a common lemma get their parents (POS tags) and grandparents, i.e., chunk labels, marked by prepending a `REL` tag. An example of a q/a pair encoded using shallow chunk trees is given in Figure 3 (top). Our empirical evaluation in [39] showed that such representation is superior to more simple bag-of-words and sequences of POS tags models.

3.1.2 *Soft matching for relational linking.*

As shown in [39], the use of a special tag to mark the related fragments in the question and answer tree representations is the key to learn more accurate relational models. However, we just used a naïve hard matching between the word lemmas. To alleviate the word sparsity problem, where semantically similar words have non-matching surface forms, we explore WordNet, topic models, SuperSense tagging and word alignments used in statistical machine translation.

Wordnet. To establish the matching between the structures in the question and a candidate answer we experiment with using synonym groups provided by WordNet. We also experiment with a similarity metric computed between words w_1 and w_2 using the WordNet concept hierarchy that defines hyponym/hypernym relations between the words:

$$sim_{WN}(w_1, w_2) = \frac{min(d(w_1, CP), d(w_2, CP))}{d(CP, R) + min(d(w_1, CP), d(w_2, CP))} \quad (2)$$

where d defines the distance in the hierarchy between two concepts, CP denotes a common parent of two words and R is the root of the WordNet hierarchy. We consider a match between two words if their sim_{WN} is below a specified threshold value. We tried various thresholds and found 0.2 to serve as a meaningful boundary. We did not use any sense disambiguation algorithms and opted for a simple strategy to take the most frequent sense (this seems the most effective approach, according to previous studies, e.g., [44]).

LDA. It has become a popular tool in discovering deeper relationships between q/a pairs, e.g., [20, 9]. It comes from a family of generative probabilistic models, where each document d in the collection D is viewed as a mixture of a fixed number of topics $z \in \mathcal{Z}$. Each topic z represents a distribution over the unique words $w \in V$ in the vocabulary V. More specifically, given a training corpus, an LDA model infers two distributions: $\phi_z^{(w)}$, the probability of a word w being assigned to a topic z, and $\theta^{(d)}$ which defines a distribution of topics for a document d.

Different from previous applications of LDA, where it is primarily used to compute various similarity scores for a given q/a pair, we consider using the obtained vector of topic assignments to each word in the document as a way to generalize hard matching on lemmas. Table 1 gives an example of several topics learnt from a large Aquaint corpus from TREC QA. It exemplifies that topics can be used to cluster words into semantically coherent groups. Hence, we explore them to link related fragments in a q/a pair.

Table 1: Top 5 words from 5 randomly picked LDA topics extracted from the Aquaint corpus.

topic1	topic9	topic24	topic51	topic77
bank	music	china	family	baseball
financial	rock	chinese	home	game
government	band	beijing	people	yankees
economic	album	xinhua	day	team
banks	songs	province	years	league

SuperSense matching. We explore an alternative approach to link question and answer words that belong to the same semantic category. For this purpose we use a SuperSense tagger[5] [10] that annotates words in a text with the tagset of 41 Wordnet supersense classes for nouns and verbs, e.g., *act, event, relation, change, person, motion*, etc. Words in a question and answer that have the same tag are used to link the related structures.

Word alignment. Recently, the utility of translation models to alleviate the lexical gap between questions and answers has been explored by many QA systems, e.g., [43, 38, 45]. The typical approach is to learn question-to-answer and answer-to-question transformations using a translation model. The translation model is finally used to compute the similarity score relating a given q/a pair, which is then integrated as a similarity feature into the learning to rank model [45]. Different from the previous approaches, we explore the utility of a translation model to align words in a given q/a pair for relational linking. To obtain the alignments, we use METEOR[6] monolingual word aligner [13] that includes flexible word and phrase matching using exact, synonym and paraphrase matches. Similar to the string matching strategy, we mark the structures spanning the aligned words with a relational tag.

3.2 Refining relational tag using question focus and question category

In this section we briefly describe our alternative strategy to establish relational links first proposed in [40]. We use question category to link the focus word of a question with the named entities extracted from the candidate answer. For this purpose, we briefly introduce our models for building a question focus detector and question category classifier.

3.2.1 *Question focus detection*

[5] `http://sourceforge.net/projects/supersensetag/`
[6] `http://www.cs.cmu.edu/~alavie/METEOR`

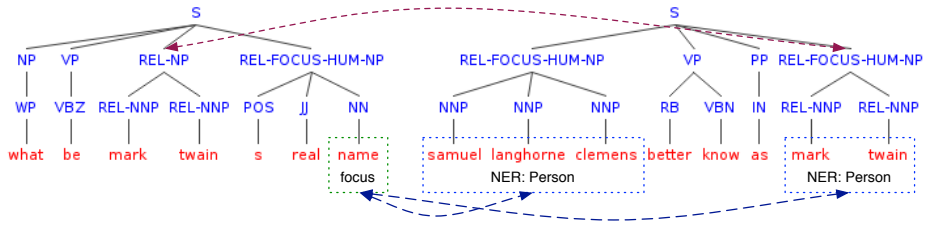


Figure 4: Using a typed relation tag `REL-FOCUS-HUM` to link a question focus word *name* with the named entities, whose type corresponds to the question category (HUM).

The question focus is typically a simple noun representing the entity or property being sought by the question [34]. It can be used to search for semantically compatible candidate answers in document passages, thus greatly reducing the search space [33]. While several machine learning approaches based on manual features and syntactic structures have been recently explored, e.g., [35, 12, 8], we opt for the latter and rely on the power of tree kernels to handle automatic feature engineering.

3.2.2 *Question classification*

Question classification is the task of assigning a question to one of the pre-specified categories. We use the coarse-grain classes described in [23]: six non overlapping classes: Abbreviations (ABBR), Descriptions (DESC, e.g., definitions or explanations), Entity (ENTY, e.g., animal, body or color), Human (HUM, e.g., group or individual), Location (LOC, e.g., cities or countries) and Numeric (NUM, e.g., amounts or dates). These categories can be used to determine the Expected Answer Type for a given question and find the appropriate entities found in the candidate answers. Imposing such constraints on the potential answer keys greatly reduces the search space.

3.2.3 *Linking focus word with named entities using question class*

Question focus captures the target information need posed by the question but, to make this piece of information effective, the focus words need to be linked to the target candidate answer. They can be lexically matched with words present in an answer, or the match can be established using semantic information. One method exploiting the latter approach involves question classification.

Once the question focus and class are determined, we propose to link the focus word w_{focus} in the question, with all the named entities whose type matches the question class. Table 2 provides the correspondence between the question classes and named entity types. We perform tagging at the chunk level and use two types of the relational tag: plain `REL-FOCUS` and a tag typed with the question class, e.g., `REL-FOCUS-HUM`. Fig. 4 shows an example q/a pair where the typed relational tag is used to link the chunk containing the question focus word *name* with the named entities of the corresponding type *Person* (according to the mapping defined in Table 2).

4. FEATURE VECTOR REPRESENTATION

While the primary focus of our study is on the structural representations and relations between the q/a pairs we also include some basic and more advanced features widely applied in QA. We use several similarity functions between q and a, computed over various input representations to form a feature vector. These feature vectors are used along with the structural models.

Table 2: Question classes → named entity types.

Question Category	Named Entity types
HUM	Person
LOC	Location
NUM	Date, Time, Money, Percentage
ENTY	Organization, Person

N-gram overlap features. We compute a cosine similarity over question and answer: $sim_{COS}(q, a)$, where the input vectors are composed of: (i) word lemmas, (ii) bi-grams, (iii) part-of-speech tags, (iv) topics, and (v) dependency triplets. For the latter, we simply hash the string value of the predicate defining the triple together with its argument, e.g., *poss(name, twain)*. We also generalize the arguments of the predicates by using topics instead of words.

Tree kernel similarity. For the structural representations we also define a similarity based on the PTK score: $sim_{PTK}(q, a) = PTK(q, a)$, where the input trees can be raw constituency trees and shallow chunk trees used in structural representations. Note that this similarity is computed between the members of a q/a pair, thus, it is very different from the one defined by Eq. 1. We also compute these features over the trees where the lexicals are replaced with their associated topics.

NER relatedness. We also compute a feature that represents a match between a question category and the related named entity types extracted from the candidate answer. We simply count the proportion of named entities in the answer that correspond to the question type returned by the question classifier.

LDA. The similarity between a question **q** and a candidate answer **a** can be captured by the similarity between their topic distributions $\theta^{(q)}$ and $\theta^{(a)}$. For this purpose, we use the symmetrized KL divergence:

$$sim_{KL}(q, a) = \frac{1}{2}[KL(\theta^{(q)}||\theta^{(c)}) + KL(\theta^{(c)}||\theta^{(q)})]$$

LDA provides yet another way to compute the similarity between two documents using conditional probabilities of one document given the topic distribution of the other. For a question **q** and its candidate answer **a** it can be estimated as follows:

$$P(q|a) = \prod_{w \in q} P(w|a) = \prod_{w \in q} \sum_{t \in T} P(w|z = t)P(z = t|a)$$

where the last term $P(z = t|a)$ is simply the probability of a topic $z = t$ under the topic distribution of an answer **a**. Hence, we define two similarities : $sim^{LDA_1}(q, a) = P(q|a)$ and $sim^{LDA_2}(q, a) = P(a|q)$, which compute the probability

of the question being generated from the topic distribution of the answer and vice versa. Differently from the features derived from translation-based language models [53], which extract knowledge from q/a pairs, topic models use the distribution of words over the entire collection.

The total number of our basic features is 24. Although far from being complete, our features represent a good sample of typical features used in many QA systems to rank candidate answers. Nevertheless, in our study feature vectors serve a complementary purpose, while the main focus is to study the virtue of structural representations for reranking. The effect of a more extensive number of features computed for the q/a pairs has been studied elsewhere, e.g., [45].

5. EXPERIMENTS

In these experiments, we evaluate our kernel models exploiting pairwise structural relationships on the answer passage re-ranking task, where the data is derived from the factoid open-domain QA corpus of TREC QA.

5.1 Setup

SVM re-ranker. To train our models, we use SVM-light-TK[7], which enables the use of structural kernels [26] in SVM-light [22]. We use default parameters as described in [39]. We choose PTK as the re-ranker kernel to establish pairwise similarities between tree structures, since PTK is the most general kernel able to generate a vast number of tree fragments. It is also particularly suitable for tree representations using dependency structures. For explicit feature vectors we use polynomial kernel of degree 3, as it has better discriminative power w.r.t. linear kernel.

LDA model. To train an LDA model we use an implementation of a parallelized Gibbs sampler from MALLET[8] library. We fix the number of iterations of the Gibbs sampler to 1000 and fix the other parameters as their default values. Since the LDA implementation performs an automatic tuning of prior parameters for the document-topic and word-topic distributions, we fixed them at their default values. As an input training data we perform basic stopword removal and lemmatization. At the inference time on the unseen test documents we set the number of iterations of the Gibbs sampler to 300.

Metrics. To measure the impact of the re-ranker on the output of our QA system, we use metrics most commonly used to assess the accuracy of QA systems: Precision at rank 1 (P@1), Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP). P@1 is the percentage of questions with a correct answer ranked at the first position, MRR is computed as follows: $MRR = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{rank(q)}$, where *rank(q)* is the position of the first correct answer in the candidate list. For a set of queries Q MAP is the mean over the average precision scores for each query: $\frac{1}{Q}\sum_{q=1}^{Q} AveP(q)$.

Pipeline. We built the entire processing pipeline on top of the UIMA Framework, which provides a convenient abstraction for developing NLP annotators for analyzing unstructured content.

We have included many off-the-shelf NLP tools wrapping them as UIMA annotators to perform sentence detection, tokenization, Named Entity Recognition, parsing, chunking and lemmatization. We included other tools such as the Stanford Parser, the Berkeley parser, the ClearTK dependency parser, the Illinois Chunker, the RitaWordnet Lemmatizer, Mallet for LDA and the Snowball stemmer. Moreover, we created annotators for building new sentences representations starting from tools' annotations. For example, the component producing a tree representations containing POS tags and chunks outputs the result as a UIMA annotation. Fully integrated into the pipeline, there are also the question focus and question classifiers.

[7] `http://disi.unitn.it/moschitti/Tree-Kernel.htm`

[8] `http://mallet.cs.umass.edu`

Table 3: TREC QA. Search Engine baselines.

Model	MAP	MRR	P@1
Lucene	0.16	21.57	13.83
Whoosh	0.20	25.92	17.59
Terrier	0.22	27.91	18.08

5.2 Corpora

Our experiments are carried out on TREC QA data, which is widely used in the evaluation of QA systems. In this task, answer passages containing correct information nuggets, i.e. answer keys, have to be extracted from a given text corpus, typically a large newswire corpus. It should be noted that the passages may contain multiple sentences. This setting is thus rather different from the one we used in Sec. 6. In our experiments, we opted for questions from 2002 and 2003 years (TREC 11-12), which totals to 824 questions. AQUAINT newswire corpus is used for searching the supporting answers. To train Question Focus and Category classifiers we follow the same setup as described in [40].

5.3 Search engines

For the questions from TREC 11-12 the supporting corpus is AQUAINT which represents a large collection of newswire text (3Gb) containing about 1 million documents. We perform indexing at the paragraph level by splitting each document into a set of paragraphs which are then added to the search index. The resulting index contains about 12 million items. For the Answerbag we index the entire 180k answers. For both TREC and Answerbag we retrieve a list of 50 candidate answers for each question.

We tested 3 open-source search-engines: Lucene, Whoosh and Terrier. The results are given in Table 3. Lucene implements the most basic retrieval model based on the cosine similarity and represents the weakest baseline, while Whoosh and Terrier, which use a more accurate BM25 scoring model, demonstrate better performance. Hence, in our further experiments we use BM25 implemented by Terrier as our baseline model.

5.4 Relational learning on TREC

To evaluate the performance of our retrieval model, we evaluate the following models previously introduced in Sec. 3:
BM25 - initial ranking obtained by the BM25 search engine model.
CH - our shallow chunk tree model in proposed [39].
V - reranker model using the set of features defined in Sec. 4.
CH+V - a combination of tree structures encoding q/a pairs with similarity scores stored in the feature vector.
F - relational linking of the question focus word and named entities of the corresponding type using Focus and Question classifiers.

Table 4: Answer passage reranking on TREC QA. † shows significant improvement (using a two tailed t-test with p < 0.05) w.r.t. to our baseline model CH [39], while ‡ compares +F and +TFC w.r.t. CH+V. Model +TF does not provide significant improvement over +F.

MODEL	MAP	MRR	P@1
BM25	0.22	28.02	18.17
V	0.25	31.82	21.61
CH [39]	0.28	35.63	24.88
CH+V	$0.30^{\dagger}$	$37.45^{\dagger}$	$27.91^{\dagger}$
+F	$0.32^{\ddagger}$	$39.48^{\ddagger}$	$29.63^{\ddagger}$
+TF	$0.32^{\ddagger}$	$39.49^{\ddagger}$	$30.00^{\ddagger}$

Table 5: CH+V (LEMMA) representation. Relational tagging with Wordnet (WN) with synonym (SYN) and hierarchy distance (DIST) matching, LDA, SSENSE, and word alignment (WALIGN) matching. None of the linking strategies provide significant improvement over LEMMA matching of CH+V.

MODEL	MAP	MRR	P@1
LEMMA	0.30	37.45	27.91
WN-DIST (0.2)	0.30	37.41	27.20
WN-SYN	0.30	37.54	27.82
LDA50	0.23	29.51	18.29
LDA100	0.28	35.41	24.67
LDA250	0.24	33.01	22.25
LDA500	0.24	32.24	21.34
SSENSE	0.29	36.32	26.77
WALIGN	0.30	37.64	28.04

TF - a typed relational link refined with the question category.

Table 4 reveals that using feature vectors provides a good improvement over BM25 baseline. Most interestingly, the structural representations give a bigger boost in the performance. Combining the structural and feature vector representation in a single model results in further improvement.

5.5 Soft Word Matching

Here, we explore the effect of using word similarity measures derived from WordNet, SuperSense tags, word alignments and LDA topic models on the performance of the reranker. We test WordNet synonym groups (SYN) and a distance (DIST) metric between two concepts to identify semantically close words. For the latter we fix the similarity threshold at 0.2. For the topic model we match words based on their topic assignments obtained when performing inference on the unseen documents. We train a set of LDA models using a fixed number of topics $Z \in \{50, 100, 250, 500\}$. Table 5 shows that using SSENSE or LDA topic matching actually results in lower performance w.r.t. plain string matching used by CH+V. Additionally, using WordNet and word alignment for relational linking does not provide any interesting improvement. Our intuition is that most of the answer candidates have a relatively high word overlap with the question (since search engine retrieves candidates based on metrics derived from word-overlap measures), hence using a plain string match on lemmas results in a rather good coverage of words between question and answer. Differently, using coarser word classes from SuperSense tagger and LDA results in much higher linking but, at the same time, they add a considerable large amount of noise. Thus, the use of larger coverage linking strategies seems to requires a definition of a more principled approach.

We provided a possible solution in [40], where we explore a set of supervised components to semantically link important concepts between questions and answers. We include such strategy in our experiments to build an even stronger baseline. The next section briefly reports the results using such linking models.

5.6 Relational learning using Question Focus and Question Class

In the following set of experiments, we test another strategy for linking structures for a given q/a pair. We use an automatically detected question focus word and a question category obtained from the question classifier to link the focus word with the related named entities in the answer (namely model F). Then, we refine the relational link by typing it with the question category (namely model TF).

Table 4 summarizes the performance of the CH+V model when coupled with F and TF strategies to link structures in a given q/a pair. The structural representations with F yields an interesting improvement, while further refining the relational tag by adding a question category (TF) gives no improvement with CH structure.

6. COMPARISON WITH THE STATE-OF-THE-ART IN SENTENCE RERANKING

In this section, we test our structural CH and CH+F models on a sentence reranking task. For the latter several studies using a common dataset are available, thus it gives us the possibility to compare with several reranking systems on exactly the same test set. First, we briefly describe the experimental setup to replicate the setting of the previous work. Then, we propose an additional set of features to build a strong feature vector baseline model and finally, we compare our models to the previous state-of-the-art systems.

6.1 Setup

We test our models on the manually curated[9] TREC QA dataset[10] from Wang et al. [50], which has been widely used for comparison of answer rerankers in previous work[11].

In their setup, 100 manually judged questions from TREC 8-12 are used for training, while questions from TREC-13 are used for testing. Additionally, Wang et. al [50] provide a "noisy setting" experiment where 2,393 questions from the entire TREC 8-12 collection is used for training. This results in a lower performance of their system, which is explained by the inclusion of erroneous candidate answers treated as

[9]Manual judgement of candidate answer sentences was carried out for the entire TREC 13 set and for the first 100 questions from TREC 8-12. The motivation behind this annotation effort is that TREC provides only the answer patterns to identify if a given passage contains a correct answer key or not. This results in many unrelated candidate answers marked as correct simply because they happen to contain a regex match with the answer key.

[10]`http://cs.stanford.edu/people/mengqiu/data/qg-emnlp07-data.tgz`

[11]Different from the experiments above, where we learn a candidate answer reranker over paragraphs, here the retrieved answer candidates represent a single sentence.

correct by TREC. Nevertheless, we compare with their best results that they obtained using their manually judged training set.

While we use the same test collection, our training data consists of only 824 questions from TREC 2002, 2003 (TREC 11-12), since TREC 8-10 require a separate license. To generate the candidate answer sentences, we extract sentences from the previously retrieved paragraphs (as used in Section 5) that have at least one non-stopword in common with the question. We build examples for our SVM reranker by pairing each correct candidate sentence with at most top 10 incorrect candidates, which results in total 8,730 examples.

6.2 Advanced features

This section describes a set of more involved features to model the similarity of q/a pairs. The features below are largely inspired by the feature sets used by the top performing system [6] in Semantic Textual Similarity (STS) task [2].

Additional word-overlap measures. We include additional word-overlap similarity metrics over lemmas for a given q/a pair by computing the longest common substring (subsequence) measures, and Greedy String Tiling. The longest common substring measure [16] determines the length of the longest string which is also a substring shared by a pair of text fragments. The longest common subsequence measure [4] considers as subsequence also strings that differ from word insertions or deletions. Greedy String Tiling [51] detects similarity of reordered text parts as it is able to identify multiple shared contiguous substrings.

Knowledge-based word similarity. We compute Resnik similarity [37] which is based on the WordNet hypernymy hierarchy and on semantic relatedness between concepts. The hierarchy is used to find a path between two concepts. Then, the semantic relatedness is determined by the lowest common concept subsuming both of them. The specificity of the subsuming concept affects the similarity measure: more specific concepts contributes more than generic ones. The aggregation strategy by Mihalcea et al. [24] is applied to scale the measure from pairs of words to sentences.

Explicit Semantic Analysis (ESA) Similarity. ESA [15] maps a document into a vector of concepts extracted from Wikipedia. Thus, the meaning of text fragment is modeled by a set of natural concepts, which are described and defined by humans. Moreover, we used WordNet and Wiktionary as additional concepts.

Lexical Substitution. A supervised word sense disambiguation system [7] finds substitutions for a wide selection of frequent English nouns. Resnik and ESA features are then computed adding the substitutions to the text. This feature enables additional matches aleviating the lexical gap between text fragments.

Translation model. We integrate two similarity score features obtained from the METEOR scorer when treating both a question and a candidate as a translation source.

This feature set adds 19 more features to our basic features from Section 4, which results in total 43 features for our advanced vector-based model (V_{adv}).

6.3 Results

Table 6 compares V_{adv} and CH structural model coupled with F relational linking strategy with the previous state-of-the-art systems[12]. In particular, we compare to four most recent state-of-the-art reranker models [50, 17, 49, 52] that report their performance on the same questions and candidate sets from TREC 13 as provided by [50].

Table 6: Answer sentence reranking on TREC 13. † indicates that improvement delivered by adding F linking to CH+V_{adv} model is significant ($p < 0.05$).

Model	MAP	MRR
Wang et al., 2007 [50]	0.6029	0.6852
Heilman & Smith, 2010 [17]	0.6091	0.6917
Wang & Manning, 2010 [49]	0.5951	0.6951
Yao et al., 2013 [52]	0.6307	0.7477
V_{ADV}	0.5627	0.6294
CH+V_{ADV}	0.6611	0.7419
+F	**0.6829**†	**0.7520**†

First note that, our combined set of basic and advanced features V_{adv} already represents a rather strong baseline model. Furthermore, combining it with CH representations gives state-of-the-art performance providing an improvement over the previous work with a large margin[13]. Finally, augmenting CH representation with F linking strategy yields additional 2 points in MAP and one point in MRR. This strongly indicates on 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.

Our kernel-based learning to rank approach 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. Moreover, the computational complexity of previous approaches limited their application to reranking of answer sentences, while our approach is demonstrated to work well also at the paragraph level.

7. RELATED WORK

Work in QA shows that semantics and syntax are essential to retrieve answers, e.g., [18, 42]. However, most approaches in TREC were based on many complex heuristics and fine manual tuning, which require large effort for system engineering and often made the result not replicable. In contrast, our passage re-ranker is adaptable to any domain and can be also used as front end of more complex QA systems.

Previous studies closely to ours carry out passage reranking by exploiting structural information. In this perspective, a typical approach is to use subject-verb-object relations, e.g., as in [5]. Unfortunately, the large variability of natural language makes such triples rather sparse thus different methods explore soft matching (i.e., lexical similarity) based on answer types and named entity types, e.g., see [3]. Passage reranking using classifiers of question and answer pairs were proposed in [36, 19]. In this context, several approaches focused on reranking the answers to definition/description questions, e.g., [41, 31, 45]. [1] propose a cascading learning to rank approach, where the ranking produced by one ranker is used as input to the next stage.

Regarding kernel methods, the work in [31, 27, 28, 29, 30, 39] were the first to exploit tree kernels for modeling

[12]P@1 metric is omitted since it is not reported in the previous work.

[13]Given the fact that our system was trained on a smaller subset of TREC 8-12, we expect a potential increase in accuracy when trained on the full data.

answer re-ranking. 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 [55, 56] for the textual entailment task but it is computationally too expensive for the large dataset of QA. Some faster versions were provided in [32, 54], which may be worth to try. In contrast, we design our re-ranking models with shallow trees encoding the output of question and focus classifiers connected to the NE information derived from the answer passage. This provides more effective relational information, which allows our model to significantly improve on previous rerankers.

Our relational structures are based on a shallow tree from [39] and we reuse our semantic linking strategy from [40], where question focus is linked to the related named entities (as identified by the question category). Additionally, this paper studies a number of linking strategies to establish connections between related tree fragments from questions and answer passages. Regarding the experimental evaluation, different from our previous work, where experiments were conducted for an answer passage reranking task on a subset of TREC QA data, in this work, we also include experiments for the answer sentence selection task on a public TREC 13 benchmark, such that we can compare to the previous state-of-the-art methods. We show that highly discriminative features can be, in fact, automatically extracted and learned by using our kernel learning framework. Our approach does not require manual feature engineering to encode input structures via similarity features. We treat the input q/a pairs directly encoding them via linguistic trees, and more powerful features can be encoded by injecting additional semantic information directly into the tree nodes.

Regarding previous state of the art in answer sentence rerankers, Wang et al., 2007 [50] use quasi-synchronous grammar to model relations between a question and a candidate answer with the syntactic transformations. Heilman & Smith, 2010 [17] 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 & Manning, 2010 [49] 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 [52] applies linear chain CRFs with features derived from TED to automatically learn associations between questions and candidate answers.

8. CONCLUSIONS

This paper shows a viable research direction in the automatic QA engineering. One of its main characteristics is the use of structural kernel technology to induce rich feature spaces from structural semantic representations of question and answer passage pairs. The same technology is also used to provide adaptable question and focus classifiers, which provides the learning to rank algorithm with important relational information.

We introduce a powerful feature-based model that when combined with the CH model and coupled with focus linking improves the previous state-of-the-art systems with a large margin. Moreover, our approach is conceptually simpler than previous systems, as it relies on the kernel-based learning and simple structures, which can be built using off-the-shelf syntactic parsers and NERs, along with little training data for the question/focus classifiers.

Finally, the inefficacy of using topic models, WordNet, SuperSense and word alignment studied in this paper suggests that information produced by unsupervised methods has still to be carefully considered. Therefore, studying ways to utilize semantic resources at their best is a natural future extension of this paper.

9. ACKNOWLEDGEMENTS

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

10. REFERENCES

[1] A. Agarwal, H. Raghavan, K. Subbian, P. Melville, D. Gondek, and R. Lawrence. Learning to rank for robust question answering. In *CIKM*, 2012.

[2] E. Agirre, D. Cer, M. Diab, and Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM*, 2012.

[3] E. Aktolga, J. Allan, and D. A. Smith. Passage reranking for question answering using syntactic structures and answer types. In *ECIR*, 2011.

[4] L. Allison and T. I. Dix. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.*, 23(6):305–310, Dec. 1986.

[5] G. Attardi, A. Cisternino, F. Formica, M. Simi, and R. Tommasi. Piqasso: Pisa question answering system. In *TREC*, pages 599–607, 2001.

[6] D. Bar, C. Biemann, I. Gurevych, and T. Zesch. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of SemEval*, 2012.

[7] C. Biemann. Creating a system for lexical substitutions from scratch using crowdsourcing. *Lang. Resour. Eval.*, 47(1):97–122, Mar. 2013.

[8] R. Bunescu and Y. Huang. Towards a general model of answer typing: Question focus identification. In *CICLing*, 2010.

[9] A. Celikyilmaz, D. Hakkani-Tur, and G. Tur. Lda based similarity modeling for question answering. In *NAACL HLT Workshop on Semantic Search*, 2010.

[10] M. Ciaramita and Y. Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*, 2006.

[11] M. Collins and N. Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *ACL*, 2002.

[12] D. Damljanovic, M. Agatonovic, and H. Cunningham. Identification of the question focus: Combining syntactic analysis and ontology-based lookup through the user interaction. In *LREC*, 2010.

[13] M. Denkowski and A. Lavie. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP Workshop on Statistical MT*, 2011.

[14] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock,

E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. Building watson: An overview of the deepQA project. *AI Magazine*, 31(3), 2010.

[15] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of IJCAI*, 2007.

[16] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Univ. Press, NY, USA, 1997.

[17] M. Heilman and N. A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*, 2010.

[18] A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. Question answering with lcc chaucer at trec 2006. In *TREC*, 2006.

[19] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM*, 2005.

[20] Z. Ji, F. Xu, B. Wang, and B. He. Question-answer topic model for question retrieval in community question answering. In *CIKM*, 2012.

[21] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *In Proceedings of ECML-98*, pages 137–142, 1998.

[22] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[23] X. Li and D. Roth. Learning question classifiers. In *COLING*, 2002.

[24] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings AAAI'06*, pages 775–780. AAAI Press, 2006.

[25] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[26] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, 2006.

[27] A. Moschitti. Kernel Methods, Syntax and Semantics for Relational Text Categorization. In *Proceeding of CIKM*, Napa Valley, CA, USA, 2008.

[28] A. Moschitti. Syntactic and Semantic Kernels for Short Text Pair Categorization. In *Proceedings EACL*, Athens, Greece 2009.

[29] A. Moschitti and S. Quarteroni. Kernels on Linguistic Structures for Answer Extraction. In ACL, 2008.

[30] A. Moschitti and S. Quarteroni. Linguistic Kernels for Answer Re-ranking in Question Answering Systems. *Information Processing & Management*, 2010.

[31] A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *ACL*, 2007.

[32] A. Moschitti and F. M. Zanzotto. Fast and effective kernels for relational learning from texts. In *Proceedings of ICML*, NY, 2007.

[33] C. Pinchak. A probabilistic answer type model. In *EACL*, 2006.

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

[35] S. Quarteroni, V. Guerrisi, and P. L. Torre. Evaluating multi-focus natural language queries over data services. In *LREC*, 2012.

[36] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. *CoRR*, 2006.

[37] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *In Proceedings of IJCAI*, 1995.

[38] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *ACL*, 2007.

[39] A. Severyn and A. Moschitti. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*, 2012.

[40] A. Severyn, M. Nicosia, and A. Moschitti. Learning adaptable patterns for passage reranking. In *CoNLL*, 2013.

[41] D. Shen and M. Lapata. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, 2007.

[42] S. Small, T. Strzalkowski, T. Liu, S. Ryan, R. Salkin, N. Shimizu, P. Kantor, D. Kelly, and N. Wacholder. HitiQA: Towards analytical question answering. In *COLING*, 2004.

[43] R. Soricut and E. Brill. Automatic question answering using the web: Beyond the factoid. *Inf. Retr.*, 9(2):191–206, 2006.

[44] C. Stokoe, M. P. Oakes, and J. Tait. Word sense disambiguation in information retrieval revisited. In *SIGIR*, 2003.

[45] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online QA collections. In *Proceedings of ACL-HLT*, 2008.

[46] E. M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Proceedings of TREC*, 2001.

[47] E. M. Voorhees. Overview of TREC 2003. In *TREC*.

[48] E. M. Voorhees. Overview of the TREC 2004 question answering track. In *TREC*, 2004.

[49] M. Wang and C. D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answer- ing. In *ACL*, 2010.

[50] M. Wang, N. A. Smith, and T. Mitaura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP*, 2007.

[51] M. J. Wise. Yap3: improved detection of similarities in computer program and other texts. In *Proceedings of SIGCSE '96*, NY, USA, 1996.

[52] P. C. Xuchen Yao, Benjamin Van Durme and C. Callison-Burch. Answer extraction as sequence tagging with tree edit distance. In *NAACL*, 2013.

[53] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR*, 2008.

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

[55] F. M. Zanzotto and A. Moschitti. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *COLING-ACL*, Sydney, Australia, 2006.

[56] F. M. Zanzotto, M. Pennacchiotti, and A. Moschitti. A Machine Learning Approach to Recognizing Textual Entailment. *NLE*, Volume 15 Issue 4, October.