A Study of using Syntactic and Semantic Structures for Concept Segmentation and Labeling

Iman Saleh[*], Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, Preslav Nakov
ALT Research Group
Qatar Computing Research Institute
{sjoty,lmarquez,amoschitti,pnakov}@qf.org.qa

Scott Cyphers, Jim Glass
MIT CSAIL
Cambridge, Massachusetts 02139
USA
{cyphers,glass}@mit.edu

Abstract

This paper presents an empirical study on using syntactic and semantic information for Concept Segmentation and Labeling (CSL), a well-known component in spoken language understanding. Our approach is based on reranking N-best outputs from a state-of-the-art CSL parser. We perform extensive experimentation by comparing different tree-based kernels with a variety of representations of the available linguistic information, including semantic concepts, words, POS tags, shallow and full syntax, and discourse trees. The results show that the structured representation with the semantic concepts yields significant improvement over the base CSL parser, much larger compared to learning with an explicit feature vector representation. We also show that shallow syntax helps improve the results and that discourse relations can be partially beneficial.

1 Introduction

Spoken Language Understanding aims to interpret user utterances and to convert them to logical forms, or, equivalently, database queries, which can then be used to satisfy the user's information needs. This process is known as Concept Segmentation and Labeling (CSL): it maps utterances into meaning representations based on semantic constituents. The latter are basically sequences of semantic entities, often referred to as concepts, attributes or semantic tags. Traditionally, grammar-based methods have been used for CSL, but more recently machine learning approaches to semantic structure computation have been shown to yield higher accuracy. However, most previous work did not exploit syntactic/semantic structures of the utterances, and the state-of-the-art is represented by conditional models for sequence labeling, such as Conditional Random Fields (Lafferty et al., 2001) trained with simple morphological and lexical features. In our study, we measure the impact of syntactic and discourse structures by also combining them with innovative features. In the following subsections, we present the application context for our CSL task and then we outline the challenges and the findings of our research.

1.1 Semantic parsing for the "restaurant" domain

We experiment with the dataset of McGraw et al. (2012), containing spoken and typed questions about restaurants, which are to be answered using a database of free text such as reviews, categorical data such as names and locations, and semi-categorical data such as user-reported cuisines and amenities.

Semantic parsing, in the form of sequential segmentation and labeling, makes it easy to convert spoken and typed questions such as "cheap lebanese restaurants in doha with take out" into database queries. First, a language-specific semantic parser tokenizes, segments and labels the question:

[$_{Price}$ cheap] [$_{Cuisine}$ lebanese] [$_{Other}$ restaurants in] [$_{City}$ doha] [$_{Other}$ with] [$_{Amenity}$ take out]

Then, label-specific normalizers are applied to the segments, with the option to possibly relabel mislabeled segments; at this point, discourse history may be incorporated as well.

[$_{Price}$ low] [$_{Cuisine}$ lebanese] [$_{City}$ doha] [$_{Amenity}$ carry out]

[*] Iman Saleh (iman.saleh@fci-cu.edu.eg) is affiliated to Faculty of Computers and Information, Cairo University.

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 193–202, Dublin, Ireland, August 23-29 2014.

Finally, a database query is formed from the list of labels and values, and is then executed against the database, e.g., MongoDB; a backoff mechanism may be used if the query does not succeed.

```
{$and [{cuisine:"lebanese"}, {city:"doha"},  {price:"low"},  {amenity:"carry out"}]}
```

1.2 Related work on CSL

Pieraccini et al. (1991) used Hidden Markov Models (HMMs) for CSL, where the observations were word sequences and the hidden states were meaning units, i.e, *concepts*. In subsequent work (Rubinstein and Hastie, 1997; Santafé et al., 2007; Raymond and Riccardi, 2007; De Mori et al., 2008), other generative models were applied, which model the joint probability of a word sequence and a concept sequence, as well as discriminative models, which directly model a conditional probability over the concepts in the input text.

Seneff (1989) and Miller et al. (1994) used stochastic grammars for CSL. In particular, they applied stochastic Finite State Transducers (FST) for recognizing constituent annotations. FSTs describe local syntactic structures with a sequence of words, e.g., noun phrases or even constituents. Papineni et al. (1998) proposed and evaluated exponential models, but, nowadays, Conditional Random Fields (Lafferty et al., 2001) are considered to be the state-of-the-art. More recently, Wang et al. (2009) illustrated an approach for CSL that is specific to query understanding for web applications. A general survey of CSL approaches can be found in (De Mori et al., 2008). CSL is also connected to a large body of work on shallow semantic parsing; see (Gildea and Jurafsky, 2002; Màrquez et al., 2008) for an overview.

Another relevant line of research with a considerable body of work is reranking in NLP. Tree kernels for reranking syntactic parse trees were first proposed in (Collins and Duffy, 2002). Some variants used explicit spaces (Kudo et al., 2005), and feature vector approaches were proposed in (Koo and Collins, 2005). Other reranking work using tree kernels regards predicate argument structures (Moschitti et al., 2006) and named entities (Nguyen and Moschitti, 2012). In (Dinarelli et al., 2011), we rerank CSL hypotheses using structures built on top of concepts, words and features that are simpler than those studied in this paper. The work of Ge and Mooney (2006) and Kate and Mooney (2006) is also similar to ours, as it models the extraction of semantics as a reranking task using string kernels.

1.3 Syntactic and semantic structures for CSL

The related work has highlighted that automatic CSL is mostly based on powerful machine learning algorithms and simple feature representations based on word and tag n-grams. In this paper, we study the impact of more advanced linguistic processing on CSL, such as shallow and full syntactic parsing and discourse structure. We use a reranking approach to select the best hypothesis annotated with concepts derived by a local model, where the hypotheses are represented as trees enriched with semantic concepts similarly to (Dinarelli et al., 2011). These tree-based structures can capture dependencies between sentence constituents and concepts. However, extracting features from them is rather difficult as their number is exponentially large. Thus, we rely on structural kernels (e.g., see (Moschitti, 2006)) for automatically encoding tree fragments, which represent syntactic and semantic dependencies from words and concepts, and we train the reranking functions with Support Vector Machines (e.g., see (Joachims, 1999)). Additionally, we experiment with several types of kernels and newly designed feature vectors.

We test our models on the above-mentioned *Restaurant* domain. The results show that *(i)* the basic CRF model, in fact semi-CRF (see below), is very accurate, achieving more than 83% in F_1-score, which indicates that improving over the semi-CRF approach is very hard; *(ii)* the upper-bound performance of the reranking approach is very high as well, i.e., the correct annotation is generated in the first 100 hypotheses in 98.72% of the cases; *(iii)* our feature vectors show improvement only when all feature groups are used together; otherwise, we only observe marginal improvement; *(iv)* structural kernels yield a 10% relative error reduction from the semi-CRF baseline, which is more than double the feature vector result; *(v)* syntactic information significantly improves on the best model, but only when using shallow syntax; and finally, *(vi)* although, discourse structures provide good improvement over the semi-CRF model, they perform lower than shallow syntax (thus, a valuable use of discourse features is still an open problem that we plan to pursue in future work).

2 CSL reranking

Reranking is based on a list of N annotation hypotheses, which are generated and sorted by probability using local classifiers. Then a reranker, typically a meta-classifier, tries to select the best hypothesis from the list. The reranker can exploit global information, and, specifically, the dependencies between the different concepts that are made available by the local model. We use semi-CRF as our local model since it yields the highest accuracy in CSL (when using a single model), and preference reranking with kernel machines to rerank the N hypotheses generated by the semi-CRF.

2.1 Basic parser using semi-CRF

We use a semi-Markov CRF (Sarawagi and Cohen, 2004), or semi-CRF, a variation of a linear-chain CRF (Lafferty et al., 2001), to produce the N-best list of labeled segment hypotheses that serve as the input to reranking. In a linear-chain CRF, with a sequence of tokens x and labels y, we approximate $p(y|x)$ as a product of factors of the form $p(y_i|y_{i-1}, x)$, which corresponds to features of the form $f_j(y_{i-1}, y_i, i, x)$, where i iterates over the token/label positions. This supports a Viterbi search for the approximate N best values of y. With M label values, if for each label y_m we know the best N sequences of labels $y_1, y_2, \ldots, y_{i-1} = y_m$, then we can use $p(y_i|y_{i-1}, x)$ to get the probability for extending each path by each possible label $y_i = y'_m$. Then for each label y'_m, we will have MN paths and scores, one from each of the paths of length $i-1$ ending with y_m. For each y'_m, we pick the N best extended paths.

With semi-CRF, we want a labeled segmentation s rather than a sequence of labels. Each segment $s_i = (y_i, t_i, u_i)$ has a label y_i as well as a starting and ending token position for the segment, t_i and u_i respectively, where $u_i + 1 = t_{i+1}$. We approximate $p(s|x)$, with factors of the form $p(s_i|s_{i-1}, x)$, which we simplify to $p(y_i, u_i|y_{i-1}, t_i)$, so features take the form $f_j(y_{i-1}, y_i, t_i, u_i)$, i.e., they can use the previous segment's label and the current segment's label and endpoints. The Viterbi search is extended to search for a pair of label and segment end. Whereas for M labels we kept track of MN paths, we must keep track of MLN paths, where L is the maximum segment length.

We use token n-gram features relative to the segment boundaries, n-grams within the segment, token regular expression and lexicon features within a segment. Each of these features also includes the labels of the previous and current segment, and the segment length.

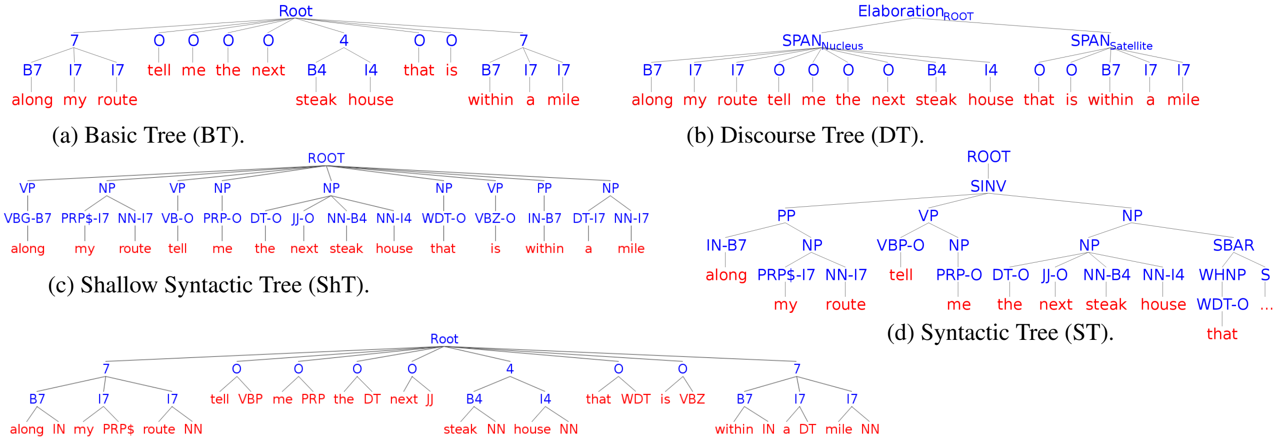
2.2 Preference reranking with kernel machines

Preference reranking (PR) uses a classifier $\mathcal{C}$ of pairs of hypotheses $\langle H_i, H_j \rangle$, which decides if H_i is better than H_j. Given each training question Q, positive and negative examples are generated for training the classifier. We adopt the following approach for example generation: the pairs $\langle H_1, H_i \rangle$ constitute positive examples, where H_1 has the lowest error rate with respect to the gold standard among the hypotheses for Q, and vice versa, $\langle H_i, H_1 \rangle$ are considered as negative examples. At testing time, given a new question Q′, $\mathcal{C}$ classifies all pairs $\langle H_i, H_j \rangle$ generated from the annotation hypotheses of Q′: a positive classification is a vote for H_i, otherwise the vote is for H_j. Also, the classifier score can be used as a weighted vote. H_k are then ranked according to the number (sum) of the (weighted) votes they get.

We build our reranker with kernel machines. The latter, e.g., SVMs, classify an input object o using the following function: $\mathcal{C}(o) = \sum_i \alpha_i y_i K(o, o_i)$, where α_i are model parameters estimated from the training data, o_i are support objects and y_i are the labels of the support objects. $K(\cdot, \cdot)$ is a kernel function, which computes the scalar product between the two objects in an implicit vector space. In the case of the reranker, the objects o are $\langle H_i, H_j \rangle$, and the kernel is defined as follow:

$$K(\langle H_1, H_2 \rangle, \langle H'_1, H'_2 \rangle) = S(H_1, H'_1) + S(H_2, H'_2) - S(H_1, H'_2) - S(H_2, H'_1).$$

Our reranker also includes traditional feature vectors in addition to the trees. Therefore, we define each hypothesis H as a tuple $\langle T, \vec{v} \rangle$ composed of a tree T and a feature vector $\vec{v}$. We then define a structural kernel (similarity) between two hypotheses H and H' as follows: $S(H, H') = S_{\text{TK}}(T, T') + S_{\text{v}}(\vec{v}, \vec{v}')$, where S_{TK} is one of the tree kernel functions defined in Section 3.1, and S_{v} is a kernel over feature vectors (see Section 3.3), e.g., linear, polynomial, gaussian, etc.


Root
7 O O O O 4 O O 7
B7 I7 I7 tell me the next B4 I4 that is B7 I7 I7
along my route steak house within a mile
Elaboration ROOT
SPAN Nucleus
SPAN Satellite
B7 I7 I7 O O O O B4 I4 O O B7 I7 I7
along my route tell me the next steak house that is within a mile
ROOT
VP NP VP NP NP NP VP PP NP
VBG-B7 PRP$-I7 NN-I7 VB-O PRP-O DT-O JJ-O NN-B4 NN-I4 WDT-O VBZ-O IN-B7 DT-I7 NN-I7
along my route tell me the next steak house that is within a mile
ROOT
SINV
PP VP NP
IN-B7 NP VBP-O NP NP SBAR
along PRP$-I7 NN-I7 tell PRP-O DT-O JJ-O NN-B4 NN-I4 WHNP S
my route me the next steak house WDT-O ...
that
Root
7 O O O O 4 O O 7
B7 I7 I7 tell VBP me PRP the DT next JJ B4 I4 that WDT is VBZ B7 I7 I7
along IN my PRP$ route NN steak NN house NN within IN a DT mile NN

(a) Basic Tree (BT).

(b) Discourse Tree (DT).

(c) Shallow Syntactic Tree (ShT).

(d) Syntactic Tree (ST).

(e) BT with POS (BTP).

Figure 1: Syntactic/semantic trees. The numeric semantic tagset is defined in Table 7.

3 Structural kernels for semantic parsing

In this section, we briefly describe the kernels we use in $S(H, H')$ for preference reranking. We engineer them by combining three aspects: *(i)* different types of existing tree kernels, *(ii)* new syntactic/semantic structures for representing CSL, and *(iii)* new feature vectors.

3.1 Tree kernels

Structural kernels, e.g., tree and sequence kernels, measure the similarity between two structures in terms of their shared substructures. One interesting aspect is that these kernels correspond to a scalar product in the fragment space, where each substructure is a feature. Therefore, they can be used in the training and testing algorithms of kernel machines (see Section 2.2). Below, we briefly describe different types of kernels we tested in our study, which are made available in the SVM-Light-TK toolkit (Moschitti, 2006).

Subtree Kernel (K0) is one of the simplest tree kernels, as it only generates complete subtrees, i.e., tree fragments that, given any arbitrary starting node, necessarily include all its descendants.

Syntactic Tree Kernel (K1), also known as a subset tree kernel (Collins and Duffy, 2002), maps objects in the space of all possible tree fragments constrained by the rule that the sibling nodes cannot be separated from their parents. In other words, substructures are composed of atomic building blocks corresponding to nodes, along with all of their direct children. In the case of a syntactic parse tree, these are complete production rules for the associated parser grammar.

Syntactic Tree Kernel + BOW (K2) extends ST by allowing leaf nodes to be part of the feature space. The leaves of the trees correspond to words, i.e., we allow bag-of-words (BOW).

Partial Tree Kernel (K3) can be effectively applied to both constituency and dependency parse trees. It generates all possible connected tree fragments, e.g., sibling nodes can be also separated and be part of different tree fragments. In other words, a fragment is any possible tree path from whose nodes other tree paths can depart. Thus, it can generate a very rich feature space.

Sequence Kernel (K4) is the traditional string kernel applied to the words of a sentence. In our case, we apply it to the sequence of concepts.

3.2 Semantic/syntactic structures

As mentioned before, tree kernels allow us to compute structural similarities between two trees without explicitly representing them as feature vectors. For the CSL task, we experimented with a number of tree representations that incorporate different levels of syntactic and semantic information.

To capture the structural dependencies between the semantic tags, we use a basic tree (Figure 1a) where the words of a sentence are tagged with their semantic tags. More specifically, the words in the sentence constitute the leaves of the tree, which are in turn connected to the pre-terminals containing the semantic tags in BIO notation (‘B’=*begin*, ‘I’=*inside*, ‘O’=*outside*). The BIO tags are then generalized in the upper level, and so on. The basic tree does not include any syntactic information.

However, part-of-speech (POS) and phrasal information could be informative for both segmentation and labeling in semantic parsing. To incorporate this information, we use two extensions of the basic tree: one that includes the POS tags of the words (Figure 1e), and another one that includes both POS tags and syntactic chunks (Figure 1c). The POS tags are children of the semantic tags, whereas the chunks (i.e., phrasal information) are included as parents of the semantic tags.

We also experiment with full syntactic trees (Figure 1d) to see the impact of deep syntactic information. The semantic tags are attached to the pre-terminals (i.e., POS tags) in the syntactic tree. We use the Stanford POS tagger and syntactic parser and the Twitter NLP tool[1] for the shallow trees.

A sentence containing multiple clauses exhibits a coherence structure. For instance, in our example, the first clause *"along my route tell me the next steak house"* is *elaborated* by the second clause *"that is within a mile"*. The relations by which clauses in a text are linked are called *coherence* relations (e.g., *Elaboration, Contrast*). Discourse structures capture this coherence structure of text and provide additional semantic information that could be useful for the CSL task (Stede, 2011). To build the discourse structure of a sentence, we use a state-of-the-art discourse parser (Joty et al., 2012) which generates discourse trees in accordance with the Rhetorical Structure Theory of discourse (Mann and Thompson, 1988), as exemplified in Figure 1b. Notice that a text span linked by a coherence relation can be either a *nucleus* (i.e., the core part) or a *satellite* (i.e., a supportive one) depending on how central the claim is.

3.3 New features

In order to compare to the structured representation, we also devoted significant effort towards engineering a set of features to be used in a flat feature-vector representation; they can be used in isolation or in combination with the kernel-based approach (as a composite kernel using a linear combination):

CRF-based: these include the basic features used to train the initial semi-CRF model (cf. Section 2.1).

n**-gram based**: we collected 3- and 4-grams of the output label sequence at the level of concepts, with artificial tags inserted to identify the start ('S') and end ('E') of the sequence.[2]

Probability-based: two features computing the probability of the label sequence as an average of the probabilities at the word level $p(l_i|w_i)$ (i.e., assuming independence between words). The *unigram* probabilities are estimated by frequency counts using maximum likelihood in two ways: *(i)* from the complete 100-best list of hypotheses; *(ii)* from the training set (according to the gold standard annotation).

DB-based: a single feature encoding the number of results returned from the database when constructing a query using the conjunction of all semantic segments in the hypothesis. Three possible values are considered by using a threshold t: 0 (if the query result is void), 1 (if the number of results is in $[1, t]$), and 2 (if the number of results is greater than t). In our case, t is empirically set to 10,000.

4 Experiments

The experiments aim at investigating which structures, and thus which linguistic models and combination with other models, are the most appropriate for our reranker. We first calculate the *oracle* accuracy in order to compute an upper bound of the reranker. Then we present experiments with the feature vectors, tree kernels, and representations of linguistic information introduced in the previous sections.

4.1 Experimental setup

In our experiments, we use questions annotated with semantic tags in the restaurant domain,[3] which were collected by McGraw et al. (2012) through crowdsourcing on Amazon Mechanical Turk.[4] We split the dataset into training, development and test sets. Table 1 shows statistics about the dataset and about the size of the parts we used for training, development and testing (see the semi-CRF line).

We subsequently split the training data randomly into ten folds. We generated the N-best lists on the training set in a cross-validation fashion, i.e., iteratively training on nine folds and annotating the remaining fold. We computed the 100-best hypotheses for each example.

[1] Available from *http://nlp.stanford.edu/software/index.shtml* and *https://github.com/aritter/twitter_nlp*, respectively.

[2] For instance, if the output sequence is *Other-Rating-Other-Amenity* the 3-gram patterns would be: S-*Other-Rating, Other-Rating-Other, Rating-Other-Amenity*, and *Other-Amenity*-E.

[3] *http://www.sls.csail.mit.edu/downloads/restaurant*

[4] We could not use the datasets used by Dinarelli et al. (2011), because they use French and Italian corpora for which there are no reliable syntactic and discourse parsers.

	Train	**Devel.**	**Test**	**Total**
semi-CRF	6,922	739	1,521	9,182
Reranker	28,482	3,695	7,605	39,782

Table 1: Number of instances and pairs used to train the semi-CRF and rerankers, respectively.

N	1	2	**5**	10	100
F_1	83.03	87.76	**92.63**	95.23	98.72

Table 2: Oracle F_1-score for N-best lists of different lengths.

We used the development set to experiment and tune the hyper-parameters of the reranking model. The results on the development set presented in Section 4.2 were obtained by semi-CRF and reranking models learned on the training set. The results on the test set were obtained by models trained on the training plus development sets. Similarly, the N-best lists for the development and test sets were generated using a single semi-CRF model trained on the training set and the training+development sets, respectively.

Each generated hypothesis is represented using a semantic tree and a feature vector (explained in Section 3) and two extra features accounting for *(i)* the semi-CRF probability of the hypothesis, and *(ii)* the hypothesis reciprocal rank in the N-best list. SVM-Light-TK[5] is used to train the reranker with a combination of tree kernels and feature vectors (Moschitti, 2006; Joachims, 1999). Although we tried several parameters on the validation set, we observed that the default values yielded the highest results. Thus, we used the default c (trade-off) and tree kernel parameters and a linear kernel for the feature vectors. Table 1 shows the sizes of the train, the development and the test sets used for the semi-CRF as well as the number of pairs generated for the reranker. As a baseline, we picked the best-scored hypothesis in the list, according to the semi-CRF tagger. The evaluation measure used in all the experiments is the harmonic mean of precision and recall, i.e., the F_1-score (van Rijsbergen, 1979), computed at the token level and micro-averaged over the different semantic types.[6] We used paired t-test to measure the statistical significance of the improvements: we split the test set into 31 equally-sized samples and performed t-tests based on the F_1-scores of different models on the resulting samples.

4.2 Results

Oracle accuracy. Table 2 shows the oracle F_1-score for N-best lists of different lengths, i.e., which can be achieved by picking the best candidate of the N-best list for various values of N. We can see that going to 5-best increases the oracle F_1-score by almost ten points absolute. Going down to 10-best only adds 2.5 extra F_1 points absolute, and a 100-best list adds 3.5 F_1 points more to yield a respectable F_1-score of 98.72. This high result can be explained considering that the size of the complete hypothesis set is smaller than 100 for most questions. Thus, we can conclude that the N-best lists do include many good options and do offer quite a large space for potential improvement. We can further observe that going to 5-best lists offers a good balance between the length of the list and the possibility to improve F_1-score: generally, we do not want too long N-best lists since they slow down computation and also introduce more opportunities to make the wrong choice for a reranker (since there are just more candidates to choose from). In our experiments with larger N, we observed improvements only for 10 and only on the development set; thus, we will focus on 5-best lists in our experiments below.

	K0	**K1**	**K2**	**K3**	**K4**
Dev	84.21	82.92	83.07	**85.07**	83.78
Test	84.08	83.19	83.20	**84.61**	82.93

Table 3: Results for using different tree kernels on the basic tree (BT) representation.

Choosing the best tree kernel. We first select the most appropriate tree kernel to limit the number of experiment variables. Table 3 shows the results of different tree kernels using the basic tree (BT) representation (see Figure 1a). We can observe that for both the development set and the test set, kernel K3 (see Section 3.1) yields the highest F_1-score.

Impact of feature vectors. Table 4 presents the results for the feature vector experiments in terms of F_1-scores and relative error reductions (row RER). The first column shows the baseline, when no reranking is used; the following four columns contain the results when using vectors including different

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

[6] 'Other' is not considered a semantic type, thus 'Other' tokens are not included in the F_1 calculation.

	Baseline	n-grams	CRF features	Count	DB	ProbBased	AllFeat
Dev	83.86	83.79	**83.96**	83.80	83.86	83.87	**84.49**
	RER	-0.4	**0.6**	-0.4	0.0	0.0	**3.9**
Test	83.03	82.90	**83.44**	82.90	83.01	83.09	**83.86**
	RER	-0.7	**2.4**	-0.7	-0.1	0.3	**4.8**

Table 4: Feature vector experiments: F_1 score and relative error reduction (in %).

							Combining AllFeat and		
	Baseline	BT	BTP	ShT	ST	AllFeat	+BT +ShT	+ShT	+BT
Dev	83.86	85.07	**85.41**	**85.06**	84.30	84.49	85.57	**85.58**	85.33
	RER	7.5	**9.6**	**7.4**	2.8	3.9	10.6	**10.7**	9.1
Test	83.03	84.61	**84.63**	**84.07**	83.81	83.86	84.67	**84.79**	84.76
	RER	9.3	**9.4**	**6.1**	4.5	4.8	9.6	**10.2**	10.2
	p.v.	0.00049	0.0002	0.012	0.032	0.00018	0.00028	0.00004	0.000023

Table 5: Tree kernel experiments: F_1-score, relative error reduction (in %), and p-values.

kinds of features: *(i)* n-gram features, *(ii)* all features used by the semi-CRF, *(iii)* count features, and *(iv)* database (DB) features. In each case, we include two additional features: the semi-CRF score (i.e., the probability) and the reciprocal rank of the hypothesis in the N-best list. Among *(i)–(iv)*, only the semi-CRF features seem to help; the rest either show no improvements or degrade the performance. However, putting all these features together (AllFeat) yields sizable gains in terms of F_1-score and a relative error reduction of 4-5%; the improvement is statistically significant, and it is slightly larger on the test dataset compared to the development dataset.

Impact of structural kernels and combinations. Table 5 shows the results when experimenting with various tree structures (see columns 2-5): *(i)* the basic tree (BT), *(ii)* the basic tree augmented with part-of-speech information (BTP), *(iii)* shallow syntactic tree (ShT), and *(iv)* syntactic tree (ST). We can see that the basic tree works rather well, yielding +1.6 F_1-score on the test dataset, but adding POS information can help a bit more, especially for the tuning dataset. Interestingly, the syntactic tree kernels, ShT and ST, perform worse than BT and BTP, especially on the test dataset. The last three columns in the table show the results when we combine the AllFeat feature vector (see Table 4) with BT and ShT. We can see that combining AllFeat with ShT works better, on both development and test sets, than combining it with BT or with both ShT and BT. Also note the big jump in performance from AllFeat to AllFeat+ShT. Overall, we can conclude that shallow syntax has a lot to offer over AllFeat, and it is preferable over BT in the combination with AllFeat. The improvements reported in Tables 5 and 6 are statistically significant when compared to the semi-CRF baseline as shown by the p.v. (value) row. Moreover, the improvement of AllFeat + ShT over BT is also statistically significant (p.v.<0.05).

			Combining AllFeat and		
	Baseline	DS	+DS	+DS +BT	+DS +ShT
Dev	83.86	84.61	85.14	85.43	**85.46**
	RER	4.7	7.9	9.7	**9.9**
Test	83.03	84.38	84.55	84.63	**84.67**
	RER	7.9	8.9	9.4	**9.6**
	p.v.	0.0005	0.0001	0.00066	0.00015

Table 6: Experiments with discourse kernels: F_1 score, relative error reduction (in %), and p-values.

Discourse structure. Finally, Table 6 shows the results for the discourse tree kernel (DS), which we designed and experimented with for the first time in this paper. We see that DS yields sizable improvements over the baseline. We also see that further gains can be achieved by combining DS with AllFeat, and also with BT and ShT, the best combination being AllFeat+DS+ShT (see last column). However, comparing to Table 5, we see that it is better to use just AllFeat+ShT and leave DS out. We would like to note though that the discourse parser produced non-trivial trees for only 30% of the hypotheses (due to the short, simple nature of the questions); in the remaining cases, it probably hurt rather than helped. We conclude that discourse structure has clear potential, but how to make best use of it, especially in the case of short simple questions, remains an open question that deserves further investigation.

Tag ID		Other	Rating	Restaurant	Amenity	Cuisine	Dish	Hours	Location	Price
0	Other	8260	35	43	110	15	19	55	113	9
1	Rating	29	266	0	**14**	3	6	0	0	8
2	Restaurant	72	6	657	**20**	19	15	0	5	0
3	Amenity	117	9	10	841	**27**	**27**	7	12	7
4	Cuisine	36	2	12	**26**	543	**44**	3	1	0
5	Dish	23	0	4	**20**	**33**	324	1	4	0
6	Hours	61	0	1	2	6	1	426	9	1
7	Location	104	1	14	**20**	2	1	1	1457	0
8	Price	22	1	0	7	0	2	0	1	204

Table 7: Confusion matrix for the output of the best performing system.

4.3 Error analysis and discussion

Table 7 shows the confusion matrix for our best-performing model *AllFeat+ShT* (rows = gold standard tags; columns = system predicted tags). Given the good results of the semantic parser, the numbers in the diagonal are clearly dominating the weight of the matrix. The largest errors correspond to missed (first column) and over-generated (first row) entity tokens. Among the proper confusions between semantic types, *Dish* and *Cuisine* tend to mislead each other most. This is due to the fact that these two tags are semantically similar, thus making them hard to distinguish. We can also notice that it is difficult to identify *Amenity* correctly, and the model mistakenly tags many other tags as *Amenity*. We looked into some examples to further investigate the errors. Our findings are as follow:

Inaccuracies and inconsistencies in human annotations. Since the annotations were done in Mechanical Turk, they have many inaccuracies and inconsistencies. For example, the word *good* with exactly the same sense was tagged as both *Other* and *Rating* by the Turkers in the following examples:

Gold: [$_{Other}$ any good] [$_{Price}$ cheap] [$_{Cuisine}$ german] [$_{Other}$ restaurants] [$_{Location}$ nearby]
Model: [$_{Other}$ any] [$_{Rating}$ good] [$_{Price}$ cheap] [$_{Cuisine}$ german] [$_{Other}$ restaurants] [$_{Location}$ nearby]

Gold: [$_{Other}$ any place] [$_{Location}$ along the road] [$_{Other}$ has a] [$_{Rating}$ good] [$_{Dish}$ beer] [$_{Other}$ selection that also serves] ...

Requires lexical semantics and more coverage. In some cases our model fails to generalize well. For instance, it fails to correctly tag *establishments* and *tameles* for the following examples. This suggests that we need to consider other forms of semantic information, e.g., distributional and compositional semantics computed from large corpora and/or using Web resources such as Wikipedia.

Gold: [$_{Other}$ any] [$_{Location}$ dancing establishments] [$_{Other}$ with] [$_{Price}$ reasonable] [$_{Other}$ pricing]
Model: [$_{Other}$ any] [$_{Amenity}$ dancing] [$_{Other}$ establishments] [$_{Other}$ with] [$_{Price}$ reasonable] [$_{Other}$ pricing]

Gold: [$_{Other}$ any] [$_{Cuisine}$ mexican] [$_{Other}$ places have a] [$_{Dish}$ tameles] [$_{Amenity}$ special today]
Model: [$_{Other}$ any] [$_{Cuisine}$ mexican] [$_{Other}$ places have a] [$_{Amenity}$ tameles] [$_{Other}$ special] [$_{Hours}$ today]

5 Conclusions

We have presented a study on the usage of syntactic and semantic structured information for improved Concept Segmentation and Labeling (CSL). Our approach is based on reranking a set of N-best sequences generated by a state-of-the-art semi-CRF model for CSL. The syntactic and semantic information was encoded in tree-based structures, which we used to train a reranker with kernel-based Support Vector Machines. We empirically compared several variants of syntactic/semantic structured representations and kernels, including also a vector of manually engineered features.

The first and foremost conclusion from our study is that structural kernels yield significant improvement over the strong baseline system, with a relative error reduction of $\sim 10\%$. This more than doubles the improvement when using the explicit feature vector. Second, we observed that shallow syntactic information also improves results significantly over the best model. Unfortunately, the results obtained using full syntax and discourse trees are not so clear. This is probably explained by the fact that user queries are rather short and linguistically not very complex. We also observed that the upper bound performance for the reranker still leaves large room for improvement. Thus, it remains to be seen whether some alternative kernel representations can be devised to make better use of discourse and other syntactic/semantic information. Also, we think that some innovative features based on analyzing the results obtained from our database (or the Web) when querying with the segments represented in each hypotheses have the potential to improve the results. All these concerns will be addressed in future work.

Acknowledgments

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

References

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 263–270, Philadelphia, PA, USA.

Renato De Mori, Dilek Hakkani-Tür, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken language understanding: a survey. *IEEE Signal Processing Magazine*, 25:50–58.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2011. Discriminative reranking for spoken language understanding. *IEEE Transactions on Audio, Speech and Language Processing*, 20(2):526–539.

Ruifang Ge and Raymond Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, COLING-ACL'06, pages 263–270, Sydney, Australia.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Thorsten Joachims. 1999. Advances in kernel methods. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Making Large-scale Support Vector Machine Learning Practical*, pages 169–184, Cambridge, MA, USA. MIT Press.

Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 904–915, Jeju Island, Korea.

Rohit Kate and Raymond Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, COLING-ACL '06, pages 913–920, Sydney, Australia.

Terry Koo and Michael Collins. 2005. Hidden-variable models for discriminative reranking. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP '05, pages 507–514, Vancouver, British Columbia, Canada.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 189–196, Ann Arbor, MI, USA.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 282–289, Williamstown, MA, USA.

William Mann and Sandra Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Lluís Màrquez, Xavier Carreras, Kenneth Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

Ian McGraw, Scott Cyphers, Panupong Pasupat, Jingjing Liu, and Jim Glass. 2012. Automating crowd-supervised learning for spoken language systems. In *Proceedings of 13th Annual Conference of the International Speech Communication Association*, INTERSPEECH '12, Portland, OR, USA.

Scott Miller, Richard Schwartz, Robert Bobrow, and Robert Ingria. 1994. Statistical language processing using hidden understanding models. In *Proceedings of the workshop on Human Language Technology*, HLT '94, pages 278–282, Morristown, NJ, USA.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel joint inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 61–68, New York City, June.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 318–329, Berlin, Heidelberg. Springer-Verlag.

Truc-Vien T. Nguyen and Alessandro Moschitti. 2012. Structural reranking models for named entity recognition. *Intelligenza Artificiale*, 6(2):177–190.

Kishore Papineni, Salim Roukos, and Todd Ward. 1998. Maximum likelihood and discriminative training of direct translation models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 189–192, Seattle, WA, USA.

Roberto Pieraccini, Esther Levin, and Chin-Hui Lee. 1991. Stochastic representation of conceptual structure in the ATIS task. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 121–124, Pacific Grove, CA, USA.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proceedings of 8th Annual Conference of the International Speech Communication Association*, INTERSPEECH '07, pages 1605–1608, Antwerp, Belgium.

Yigal Dan Rubinstein and Trevor Hastie. 1997. Discriminative vs informative learning. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD '97, pages 49–53, Newport Beach, CA, USA.

Guzmán Santafé, Jose Lozano, and Pedro Larrañaga. 2007. Discriminative vs. generative learning of Bayesian network classifiers. *Lecture Notes in Computer Science*, 4724:453–464.

Sunita Sarawagi and William Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems*, NIPS '04, pages 1185–1192, Vancouver, British Columbia, Canada.

Stephanie Seneff. 1989. TINA: A probabilistic syntactic parser for speech understanding systems. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '89, pages 168–178, Philadelphia, PA, USA.

Manfred Stede. 2011. *Discourse Processing*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers.

Cornelis Joost van Rijsbergen. 1979. *Information Retrieval*. Butterworth.

Ye-Yi Wang, Raphael Hoffmann, Xiao Li, and Jakub Szymanski. 2009. Semi-supervised learning of semantic classes for query understanding: from the web and for the web. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 37–46, New York, NY, USA.