SPOKEN LANGUAGE UNDERSTANDING WITH KERNELS FOR SYNTACTIC/SEMANTIC STRUCTURES

Alessandro Moschitti, Giuseppe Riccardi, Christian Raymond
Department of Information and Communication Technology
University of Trento
38050 Povo di Trento, Italy
{moschitti,riccardi,christian.raymond}@dit.unit.it

ABSTRACT

Automatic concept segmentation and labeling are the fundamental problems of Spoken Language Understanding in dialog systems. Such tasks are usually approached by using generative or discriminative models based on n-grams. As the uncertainty or ambiguity of the spoken input to dialog system increase, we expect to need dependencies beyond n-gram statistics. In this paper, a general purpose statistical syntactic parser is used to detect syntactic/semantic dependencies between concepts in order to increase the accuracy of sentence segmentation and concept labeling. The main novelty of the approach is the use of new tree kernel functions which encode syntactic/semantic structures in discriminative learning models. We experimented with Support Vector Machines and the above kernels on the standard ATIS dataset. The proposed algorithm automatically parses natural language text with off-the-shelf statistical parser and labels the syntactic (sub)trees with concept labels. The results show that the proposed model is very accurate and competitive with respect to state-of-the-art models when combined with n-gram based models.

Index Terms— Spoken Language Understanding, Natural Language Processing, Kernel Methods

1. INTRODUCTION

In conversational systems, Spoken Language Understanding (SLU) performs the mapping between the speech transcriptions and conceptual structures. To cope with parser robustness issues and conversational speech disfluencies shallow parsing is typically preferred to full-sentence parsing. Thus, the first step in SLU generally relates to the extraction of sequences of semantic units called concepts. The automatic segmentation and classification of concepts can be carried out by applying machine learning approaches to word sequence labeling. For example, in the following Air Travel Information System (ATIS [1]) sentence:

$$\underbrace{list}_{null}\ \underbrace{twa}_{airline\_code}\ \underbrace{flights}_{null}\ \underbrace{from}_{null}\ \underbrace{Washington}_{fromloc.city}\ \underbrace{to}_{null}\ \underbrace{Philadelphia}_{toloc.city}$$

This work was partially funded by the European Commission - LUNA project contract N° 33549.

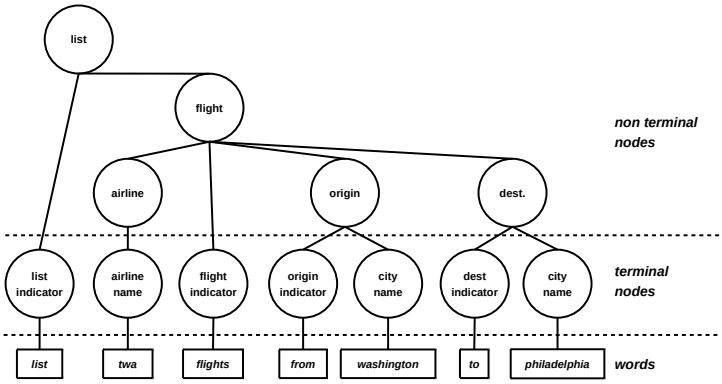

Fig. 1. Tree Structure Meaning Representation

a word sequence output by an automatic speech recognition (ASR) is mapped by a classifier into a concept sequence, where *airlines_code*, *fromLoc.city* and *toLoc.city* are concept categories and *null* indicates that the target word is not relevant for the application domain.

Previous studies show that discriminative models for sequential classification based for example on Support Vector Machines (SVMs) [2] or Conditional Random Fields (CRF) [3, 4], allow for the use of many correlated features which are difficult to include into generative models [6, 7] (see [13] for a discriminative and generative comparison on SLU databases).

Despite the accuracy improvement provided by discriminative approaches, they may result inadequate to parse concept structures of complex application domains. Indeed, only simple feature spaces based on n-grams are used, which tend to neglect long semantic dependencies. As shown in [8], *Tree Structure Meaning Representation* provides a better solution to map words in conceptual structures. In these trees, concepts are nodes whose children are concept components. For example, Figure 1 shows a possible semantic tree representation of the previous ATIS sentence. Such structure encodes long dependencies between concepts, but its automatic production is expensive as it requires the design of a specific parser and the manual annotation of many training examples.

In this paper, we propose a general purpose statistical syn-

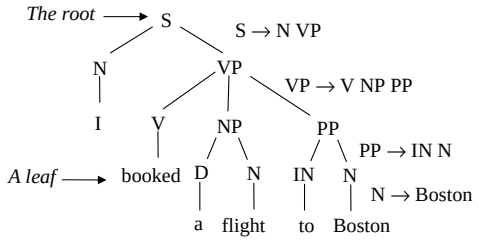

Fig. 2. A syntactic parse tree.

tactic parser to automatically generate semantic trees. The basic idea relates to enriching syntactic trees with conceptual categories. This is done by first annotating the leaves and then percolating the semantic information toward the root by means of different strategies. The resulting trees is affected by noise due to speech and parsing errors and to the use of an approximated algorithm to add semantic information. A solution to deal with such noisy representation is the use of tree kernel functions. The kernel functions encode trees in the learning algorithm by means of all possible subtrees in the feature space. The main advantages are: (a) flexible approach to represent structures and (b) robustness to noise since errors in parse trees produce a portion of irrelevant features (tree subparts), which are *ignored* by maximum margin approach like Support Vector Machines.

The paper is organized as follows. Section 2 presents the diverse syntactic semantic structures, the algorithms to automatically obtaining them and the tree kernels for their representation. Section 3 reports the experiments with our models combined with state-of-art approaches and Section 4 concludes the paper.

2. SYNTACTIC SEMANTIC STRUCTURES FOR CONCEPT CLASSIFICATION

The design of satisfactory NLU systems requires that the meaning representation in the learning algorithm is both precise and appropriate. As pointed out in [8], it should be expressive, annotable, trainable and computationally tractable. Tree structured meaning representations have the above properties and can be fully aligned to the words of a sentence but they require high annotation effort.

To solve this problem, we use an off-the-shelf statistical syntactic parser and enrich the produced trees with semantic information. This is achieved by marking the nodes corresponding to the concepts and percolating the semantic labels up towards the parse tree root.

2.1. Syntactic Parse Trees

In our study, we consider syntactic parse trees, consequently, each node with its children is associated with a grammar production rule, where the symbol at left-hand side corresponds to the parent node and the symbols at right-hand side are associated with its children.

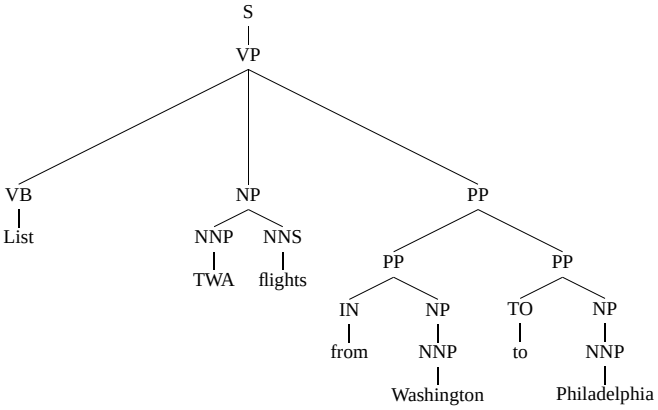

Fig. 3. Charniak Parse Tree

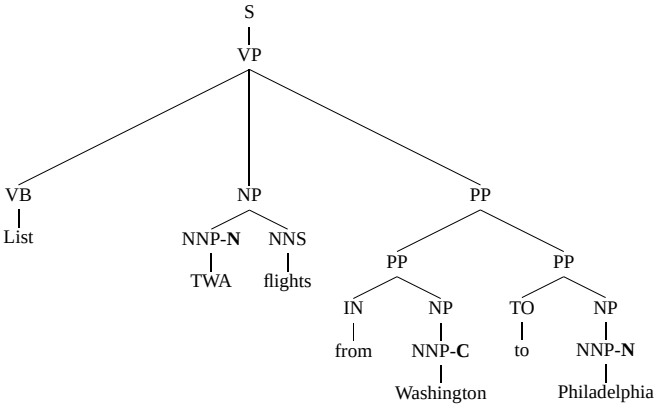

Fig. 4. Syntactic Semantic Tree

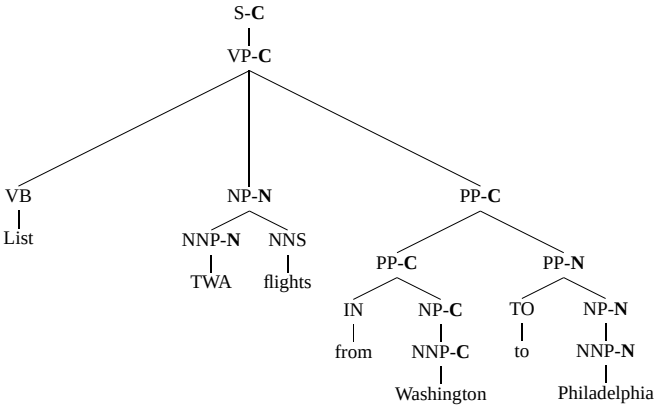

Fig. 5. Syntactic Semantic Tree with percolating concepts

The terminal symbols of the grammar are always associated with the leaves of the tree. For example, Figure 2 illustrates the syntactic parse of the sentence `"I booked a flight to Boston"`.

Such tree expresses a global syntactic information that characterizes no specific concept. To solve this problem, we *mark* the target node in the tree as suggested in [9]. In this way, we focus on a specific concept, adding, at the same time, its semantic information to the tree.

2.2. Adding Semantic Information to Parse Trees

Figure 3 shows the parse tree of the ATIS sentence discusses in the Introduction section. To characterize the departing city (*fromloc.city*), *i.e.* Washington, we could mark its parent node with the tag `C` (standing for concept) as shown in Figure 4.

However, the other concepts also bring semantic information therefore, we also mark *TWA* and *Philadelphia* with another tag, *i.e.* `N` (standing for not the target concept), to distinguish the different semantic roles played by them (see Figure 4).

Once all the concepts have been marked, to obtain a true Meaning Representation, the conceptual information should be propagated to the higher nodes of the trees. In [8], the parent node provides a generalization of the child concepts. For example, in Figure 1, the two concepts *Origin Indicator* and *City Name* are generalized in the *Origin* concept. We obtain a similar effect by percolating the conceptual information from the child to the father node. As shown in Figure 5, the `PP` dominating *from* and *Washington* receives the semantic label from the latter.

By iterating the label propagation and giving the precedence to the target concept label, *i.e.* `C`, we obtain the tree of the previous figure. It is worth nothing that such semantic tree is not identical to the manually-derived semantic tree representation since a general purpose parser groups together constituents according to a general language grammar. For example, *from Washington to Philadelphia* are grouped together whereas semantic consideration specific to the ATIS domain would have suggested to consider *TWA flights from Washington to Philadelphia* as single semantic unit.

Although the syntactic parser does not produce this intermediate level, the approximation of many tree subparts is remarkably accurate. This is interesting if we consider that tree kernels allow for the extraction of all possible substructures whose relevance can be decide by *powerful* learning algorithms like Support Vector Machines (SVMs).

The next section shows how it is possible to work in a huge features space using the implicit representation provided by tree kernels.

2.3. Tree Kernel Functions

Tree Kernels represent trees in terms of their substructures (fragments) which are mapped into feature vector spaces, *e.g.* $\Re^n$. The kernel function measures the similarity between two trees by counting the number of their common fragments. For example, Figure 6 shows for the parse tree of the sentence `"book a flight"` some of its substructures.

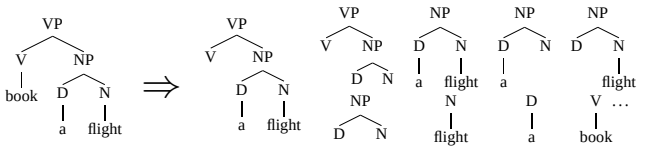

Fig. 6. A tree with some of its substructures.

The main advantage of tree kernels is that, to compute the substructures shared by two trees T_1 and T_2, the whole fragment space is not used. In the following the formal definition is reported [10].

Given the set of fragments $\{f_1, f_2, ..\} = \mathcal{F}$, the indicator function $I_i(n)$ is equal 1 if the target f_i is rooted at node n and 0 otherwise. A tree kernel is then defined as:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2) \qquad (1)$$

where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. This latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes and Δ can be evaluated with the following algorithm:

1. if the productions at n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. if the productions at n_1 and n_2 are the same, and n_1 and n_2 have only leaf children (*i.e.* they are pre-terminals symbols) then $\Delta(n_1, n_2) = 1$;
3. if the productions at n_1 and n_2 are the same, and n_1 and n_2 are not pre-terminals then

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j)) \qquad (2)$$

where $nc(n_1)$ is the number of the children of n_1 and c_n^j is the j-th child of the node n. Note that, since the productions are the same, $nc(n_1) = nc(n_2)$.

Additionally, we add the decay factor λ by modifying steps (2) and (3) as follows[1]:

2. $\Delta(n_1, n_2) = \lambda$,
3. $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j))$.

The computational complexity of Eq. 1 is $O(|N_{T_1}| \times |N_{T_2}|)$ although the average running time tends to be linear [11].

The above kernel function applied to the semantic trees defined in Section 2 allows us to use them in learning algorithms. The next section reports the experiments with one of the most accurate machine learning model, i.e. SVMs.

[1] To have a similarity score between 0 and 1, we also apply the normalization in the kernel space, *i.e.* $K'(T_1, T_2) = \frac{TK(T_1, T_2)}{\sqrt{TK(T_1, T_1) \times TK(T_2, T_2)}}$.

3. EXPERIMENTS

In this study, we focused on the well known ATIS corpus which allows us to compare with several literature results, *e.g.* [6, 3, 12]. One of its particularities is that all the concepts that appears in the test data appears as well in the training data. Moreover, there is no ambiguity between concepts and not-concepts, *i.e.* the terms collected from training data when are matched in the test data always derive a correct concept. This means that the concept segmentation task is trivial and therefore, we decided to focus in the more interesting problem of concept disambiguation, *e.g.* if a city name refers to an arrival or a departure city.

We experimented with the ATIS concept classification and our semantic tree kernel approach comparing also with state-of-the-art classification models, *e.g.* Conditional Random Fields and Finite State Traducers. The next sections define our experimental setup.

3.1. ATIS dataset

The Air Travel Information System (ATIS) task is designed to provide flight information. The semantic representation used is frame based. The SLU goal is to map the language query into a frame/slot structure, *e.g.*

$$\begin{bmatrix} \text{list flights from boston to Philadelphia} \\ \text{FRAME:} \quad \text{FLIGHT} \\ \qquad\quad \text{FROMLOC.CITY = boston} \\ \qquad\quad \text{TOLOC.CITY = Philadelphia} \end{bmatrix}$$

We used the same dataset as [6]: the training set consists of 4,978 utterances selected from the Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora whilst the ATIS test set contains both the ATIS-3 NOV93 and DEC94 data. Each training utterance is annotated with an abstract semantic annotation derived semi-automatically from the SQL queries provided in ATIS-3. We preprocessed the training set as described in [13].

3.2. Models for Concept Classification

We experimented with concept classification which is a multi-classification approach. The general setting is to learn a binary classifier for each single concept following the schema One-vs-All [14]. In the classification step the concept class that assigns the highest classification score (*e.g.* in case of SVMs this is the instance margin) to the classification instance is selected. This simple strategy has been shown to be one of the most accurate method [14].

Given the above setting, the diverse multiclassification approaches only differ by the use of distinct models for the binary classification step. Hereafter, we define those employed in our experiments.

The first set of approaches that we present are based on SVMs using the kernel function defined by Eq. 1 and the structures described in Section 2 (obtained from Charniak's parser trees). The resulting models are:

- Simple Parse Tree (SPT), in which the sentence parse tree of the target concept is used to encode it in the learning and classification algorithms. This tree alone is not effective as different concepts belong to the same sentence. Thus the classifier only learns which concept most probably belong to a parse trees.
- Concept Marked Tree (CMT), the target classification concept is marked so that the classifier can learn to classify a specific concept. In the test phase, for each sentence, as marked parse trees as the number of the sentence's concepts are generated. Again, in each parse tree, only one concept is marked as a target and each of the n concept classifier will make a decision on it.
- All Concept Marked Tree (ACMT), similar to CMT but also the concepts different from the target receive (a different) label. Therefore, for each target concept, a parse tree similar to the one shown in Figure 4 is generated.
- Percolating Concept Marked Tree (PACMT), the ACMTs are generated and then concept labels are percolated toward the root as described by Figure 5.

To experiment with such models, we used the above trees with the SVM-light-TK software[2] which encodes tree kernel functions in SVM-light [5].

The other models that we selected to compare with are described below.

The first approach is the Stochastic Finite State Transducers (SFST) [7, 15, 6]. SFST based SLU is a translation process in which stochastic language models are implemented by Finite State Machines (FSM). There is an FSM for each elementary concept. These FSMs are transducers that take words as input and output the concept tag conveyed by the accepted phrase. All these transducers are grouped together into a single transducer, called λ_{w2c}, which is the union of all of them. A stochastic conceptual language model is computed as the joint probability $P(W, C)$:

$$\begin{aligned} P(W,C) &= \prod_{i=1}^{k} P(w_i c_i | h_i) \\ \text{where } h_i &= \{w_{i-1}c_{i-1}, \ldots, w_1 c_1\}, \end{aligned}$$

where $C = c_1, c_2, \ldots, c_k$ is the sequence of concepts and $W = w_1, w_2, \ldots, w_k$ is the sequence of words. h_i is approximated by $\{w_{i-1}c_{i-1}, w_{i-2}c_{i-2}\}$ as 3-gram model. This model called λ_{SLM} is also encoded as an FSM. Given a new sentence W and its FSM representation λ_W, the translation

[2] `http://ai-nlp.info.uniroma2.it/moschitti/`

process finds the best path of the transducer resulting of the next composition:

$$\lambda_{SLU} = \lambda_W \circ \lambda_{w2c} \circ \lambda_{SLM}$$

All operations are done using the AT&T FSM/GRM Library [16].

The second approach, Poly-SVM, is based on SVMs using a polynomial kernel and the Yamcha generated features [2]. These are constituted by the target word and the context words in a windows of [-3, 1] around the target. We used the polynomial kernel of degree 2 defined by $Poly(\vec{x_1}, \vec{x_2}) = (\vec{x_1} \cdot \vec{x_2} + 1)^2$, where $\vec{x_1}$ and $\vec{x_2}$ are the feature vectors of two concepts, c_1 and c_2. Moreover, we used the default value of SVM-Light for the trade-off parameter and a cost-factor equal to 100.

The third approach relies on maximum entropy models like MEMMs or CRFs [17]. A CRF [17] is an undirected graphical model globally conditioned on the observation sequence. The structure of the graph represents the conditional independencies in the label sequences being modeled. When modeling sequences, the most common graph structure encountered is the simple first-order chain structure. According to the used graph, CRFs specify a single joint probability distribution over the entire label sequence given by:

$$\begin{aligned} p(y|x) &= \frac{1}{Z(x)} exp(\sum_{c \in C} \sum_{k} \lambda_k f_k(y|c, x, c)) \\ Z(x) &= \sum_{y} exp(\sum_{c \in C} \sum_{k} \lambda_k f_k(y|c, x, c)), \end{aligned}$$

where x is an observation sequence, y a label sequence, f_k are functions called feature functions, c is the maximal clique of the graph G on which these functions apply, λ_k are weights associated with each features f_k and λ_k are the parameters of the model estimated during the training process. In most cases, the feature functions are binary functions returning 1 if there is a match, 0 if not (the domain knowledge is encoded in the model using these functions).
In these experiments, we used CRF++ [18] an open source implementation of Conditional Random Fields.

Finally, since FST, Poly-SVM and CRF represent the state-of-the-art, it is worthwhile to combine the best of them with the most accurate semantic tree approach. Indeed, an interesting property of Kernel Methods is that we can combine feature spaces by simply summing different kernels. Thus, we also experimented with the Combined Kernel Model (CKM), which combines Poly-SVM with PAMCT as follows:

$$CK(c_1, c_2) = \frac{TK(T_1, T_2)}{|TK(T_1, T_2)|} + \frac{Poly(\vec{x_1}, \vec{x_2})}{|Poly(\vec{x_1}, \vec{x_2})|} \quad (3)$$

where T_1 and T_2 are the trees associated with the two concepts, c_1 and c_2.

	SPT	CMT	ACMT	PAMCT
Accuracy	30.06	89.89	90.24	91.98

Table 1. Accuracy of the Semantic Tree Kernel model on ATIS test set

3.3. Model Results

The first experiment aims at evaluating which semantic tree based models is the most accurate. We selected the kernel parameter λ (see Eq. 2.3) equal to 0.1 and the cost-factor parameter equal to 100 from a randomly generated validation set. Then, we trained the 65 concept classifiers and evaluated them on the ATIS test set.

The multiclassification results for the models described in Section 2 are reported on Table 1. We note that according with the initial observations, no node marking, *i.e.* SPT, does not allow SVMs to learn concepts (a very low accuracy of 30.30% is obtained) whereas if at least the target concept node is marked, *i.e.* in the CMT model, SVMs achieve a good accuracy, *i.e.* 89.89%. When all the semantic information is used, *i.e.* in the AMCT model, and is distributed on the whole tree, *i.e.* in the PAMCT model, SVMs achieve an accuracy of 90.24% and 91.98%, respectively. The latter outcome is very interesting as it relevantly improves the best result in [6], *i.e.* 89.28%[3]

The above results are remarkably high but they are lower than the current state-of-the-art [13]. Indeed, our experiments with CRF, FST and Poly-SVM show an accuracy of 94.91%, 94.25% and 95.53%, respectively. Thus, to improve the state-of-the-art, we combined PAMCT with the best model, *i.e.* Poly-SVM. Table 2, shows the comparative results between Poly-SVM, CRF, FST, PACMT and the combination, *i.e.* CKM (see Eq. 3). The multiclassification accuracy, reported in the last column, *i.e.* 95.88%, shows that CKM slightly improves Poly-SVM, therefore improving the current state-of-the-art (although such result may not be statistically significant). This limited impact is due to the extremely high accuracy of Poly-SVM, *i.e.* 95.53%, which does not leave room for a large improvement.

However, if we examine the F1 measure of the single classifiers, we note that the CKM potentials are remarkable. For example, it increases the *city.name* F1 by about 5 absolute percent points, *i.e.* 67.44 vs 71.91. Although many other categories are improved, the most frequent categories, *i.e. fromloc.cityname* and *toloc.cityname* do not receive any benefit

[3] It should be noted that the results in [6] are not perfectly comparable with ours since they refer to the slot filling f-measure rather than to concept classification accuracy. However, since the concept segmentation in ATIS is a trivial task, concept classification is a more difficult problem. This because in slot filling, it is possible to assign a *null* value whereas in concept classification always a true value must be selected, *i.e.* if a classifier outputs a *null* value, it is always considered as an error. This, in the f-measure perspective, is counted as a double error, *i.e.* one error in Precision and one error in Recall.

Concept	Poly-SVM			PACMT			CKM		
	P	R	F1	P	R	F1	P	R	F1
depart_time.time	81.82	94.74	87.80	73.24	91.23	81.25	82.35	98.25	**89.60**
arrive_time.time	100	82.35	90.32	82.86	85.29	84.06	91.43	94.12	**92.75**
city_name	100	50.88	67.44	78.05	56.14	65.31	100	56.14	**71.91**
fromloc.city_name	99.44	100	**99.72**	93.10	97.73	95.36	99.15	100	99.58
toloc.city_name	98.74	99.16	**98.95**	94.11	98.87	96.43	98.74	99.02	98.88
meal	100	36.36	53.33	100	90.91	**95.24**	100	81.82	**90.00**
Overall Accuracy	95.53			91.98			95.88		

Table 2. Comparison between Semantic Tree Kernel model on ATIS test set

from CKM. Since their number of test instances is an order of magnitude higher than the others, they dominate the overall accuracy by limiting the impact of CKM.

4. CONCLUSION

In this paper, we have proposed a model to automatically produce Tree Structure Meaning Representations. Our approach uses an off-the-shelf statistical parser to generate an initial syntactic parse tree which is semantically enriched with conceptual information.

Given the noise introduced by the automatic syntactic parsing and the general approach to add semantics, we adopted a feature representation based on tree kernels. These along with SVMs select the correct and relevant semantic substructures. The comparative experiments with the ATIS corpus show that our model is valid and also improves the state-of-the-art in concept classification. Moreover, error analysis shows that its limited impact on ATIS is due to the intrinsic simplicity of such dataset. Thus, our approach is a promising research line that should be explored in the design of complex dialog systems.

5. REFERENCES

[1] Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg, "Expanding the scope of the atis task: the atis-3 corpus," in *HLT*, 1994, pp. 43–48.

[2] Taku Kudo and Yuji Matsumoto, "Chunking with support vector machines," in *NAACL*, Morristown, NJ, USA, 2001, pp. 1–8.

[3] Ye-Yi Wang and Alex Acero, "Discriminative models for spoken language understanding," in *ICSLP*, September 2006.

[4] Minwoo Jeong and Gary Geunbae Lee, "Exploiting non-local features for spoken language understanding.," in *ACL*, 2006.

[5] T. Joachims, "Making large-scale SVM learning practical.," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds., 1999.

[6] Yulan He and Steve Young, "Semantic processing using the hidden vector state model," *Computer Speech and Language*, vol. 19, no. 1, pp. 85–106, 2005.

[7] Esther Levin and Roberto Pieraccini, "Concept-based spontaneous speech understanding system," in *EUROSPEECH*, Madrid, Spain, 1995, pp. 555–558.

[8] Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz, "Hidden understanding models of natural language," in *Proceedings of ACL*, Morristown, NJ, USA, 1994, pp. 25–32.

[9] Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin, and Roberto Basili, "Engineering of syntactic features for shallow semantic parsing," in *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, Ann Arbor, Michigan, 2005, pp. 48–56.

[10] Michael Collins and Nigel Duffy, "New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron," in *ACL02*, 2002.

[11] Alessandro Moschitti, "Efficient convolution kernels for dependency and constituent syntactic trees," in *Proceedings of ECML*, Berlin, Germany, 2006.

[12] Ye-Yi Wang, Alex Acero, Milind Mahajan, and John Lee, "Combining statistical and knowledge-based spoken language understanding in conditional models," in *Proceedings of COLING/ACL 2006*, Sydney, Australia, pp. 882–889.

[13] Christian Raymond and Giuseppe Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Interspeech*, 2007.

[14] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, Vol. 5, 101-141, 2004.

[15] Christophe Servan, Christian Raymond, Frédéric Béchet, and Pascal Nocéra, "Conceptual decoding from word lattices: application to the spoken dialogue corpus media," in *ICSLP*, Pittsburgh, USA, 2006.

[16] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer, Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[17] John D. Lafferty, Andrew Mccallum, and Fernando C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, San Francisco, CA, USA, 2001, pp. 282–289.

[18] Taku Kudo, "Crf++," online: http://chasen.org/˜taku/software/CRF++/.