Re-Ranking Models Based-on Small Training Data for Spoken Language Understanding

Marco Dinarelli
University of Trento
Italy
dinarelli@disi.unitn.it

Alessandro Moschitti
University of Trento
Italy
moschitti@disi.unitn.it

Giuseppe Riccardi
University of Trento
Italy
riccardi@disi.unitn.it

Abstract

The design of practical language applications by means of statistical approaches requires annotated data, which is one of the most critical constraint. This is particularly true for Spoken Dialog Systems since considerably domain-specific conceptual annotation is needed to obtain accurate Language Understanding models. Since data annotation is usually costly, methods to reduce the amount of data are needed. In this paper, we show that better feature representations serve the above purpose and that structure kernels provide the needed improved representation. Given the relatively high computational cost of kernel methods, we apply them to just re-rank the list of hypotheses provided by a fast generative model. Experiments with Support Vector Machines and different kernels on two different dialog corpora show that our re-ranking models can achieve better results than state-of-the-art approaches when small data is available.

1 Introduction

Spoken Dialog Systems carry out automatic speech recognition and shallow natural language understanding by heavily relying on statistical models. These in turn need annotated data describing the application domain. Such annotation is far the most expensive part of the system design. Therefore, methods reducing the amount of labeled data can speed up and lower the overall amount of work.

Among others, Spoken Language Understanding (SLU) is an important component of the systems above, which requires training data to translate a spoken sentence into its meaning representation based on semantic constituents. These are conceptual units instantiated by sequences of words.

In the last decade two major approaches have been proposed to automatically map words in concepts: (i) generative models, whose parameters refer to the joint probability of concepts and constituents; and (ii) discriminative models, which learn a classification function based on conditional probabilities of concepts given words.

A simple but effective generative model is the one based on Finite State Transducers. It performs SLU as a translation process from words to concepts using Finite State Transducers (FST). An example of discriminative model used for SLU is the one based on Support Vector Machines (SVMs) (Vapnik, 1995), as shown in (Raymond and Riccardi, 2007). In this approach, data is mapped into a vector space and SLU is performed as a classification problem using Maximal Margin Classifiers (Vapnik, 1995). A relatively more recent approach for SLU is based on Conditional Random Fields (CRF) (Lafferty et al., 2001). CRFs are undirected graphical and discriminative models. They use conditional probabilities to account for many feature dependencies without the need of explicitly representing such dependencies.

Generative models have the advantage to be more robust to overfitting on training data, while discriminative models are more robust to irrelevant features. Both approaches, used separately, have shown good accuracy (Raymond and Riccardi, 2007), but they have very different characteristics and the way they encode prior knowledge is very different, thus designing models that take into account characteristics of both approaches are particularly promising.

In this paper, we propose a method for SLU based on generative and discriminative models: the former uses FSTs to generate a list of SLU hypotheses, which are re-ranked by SVMs. To effectively design our re-ranker, we use all pos-

sible word/concept subsequences with gaps of the spoken sentence as features (*i.e.* all possible n-grams). Gaps allow for encoding long distance dependencies between words in relatively small sequences. Since the space of such features is huge, we adopted kernel methods, i.e. sequence kernels (Shawe-Taylor and Cristianini, 2004) and tree kernels (Collins and Duffy, 2002; Moschitti, 2006a) to implicitly encode them along with other structural information in SVMs.

We experimented with different approaches for training the discriminative models and two different corpora: the french MEDIA corpus (Bonneau-Maynard et al., 2005) and a corpus made available by the European project LUNA[1] (Dinarelli et al., 2009b). In particular, the new contents with respect to our previous work (Dinarelli et al., 2009a) are:

- We designed a new sequential structure (SK2) and two new hierarchical tree structures (MULTILEVEL and FEATURES) for re-ranking models (see Section 4.2). The latter combined with two different tree kernels originate four new different models.
- We experimented with automatic speech transcriptions thus assessing the robustness to noise of our models.
- We compare our models against Conditional Random Field (CRF) approaches described in (Hahn et al., 2008), which are the current state-of-the-art in SLU. Learning curves clearly show that our models improve CRF, especially when small data sets are used.

The remainder of the paper is organized as follows: Section 2 introduces kernel methods for structured data, Section 3 describes the generative model producing the initial hypotheses whereas Section 4 presents the discriminative models for re-ranking them. The experiments and results are reported in Section 5 and the conclusions are drawn in Section 6.

2 Feature Engineering via Structure Kernels

Kernel methods are viable approaches to engineer features for text processing, *e.g.* (Collins and Duffy, 2002; Kudo and Matsumoto, 2003; Cumby and Roth, 2003; Cancedda et al., 2003; Culotta and Sorensen, 2004; Toutanova et al., 2004; Kudo et al., 2005; Moschitti, 2006a; Moschitti et al., 2007; Moschitti, 2008; Moschitti et al., 2008; Moschitti and Quarteroni, 2008). In the following, we describe structure kernels, which will be used to engineer features for our discriminative reranker.

2.1 String Kernels

The String Kernels that we consider count the number of substrings containing gaps shared by two sequences, *i.e.* some of the symbols of the original string are skipped. We adopted the efficient algorithm described in (Shawe-Taylor and Cristianini, 2004; Lodhi et al., 2000). More specifically, we used words and markers as symbols in a style similar to (Cancedda et al., 2003; Moschitti, 2008). For example, given the sentence: *How may I help you ?* sample substrings, extracted by the Sequence Kernel (SK), are: *How help you ?*, *How help ?*, *help you*, *may help you*, etc.

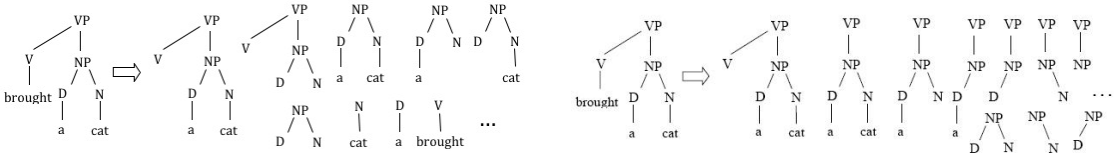
2.2 Tree kernels

Tree kernels represent trees in terms of their substructures (fragments). The kernel function detects if a tree subpart (common to both trees) belongs to the feature space that we intend to generate. For such purpose, the desired fragments need to be described. We consider two important characterizations: the syntactic tree (STF) and the partial tree (PTF) fragments.

2.2.1 Tree Fragment Types

An STF is a general subtree whose leaves can be non-terminal symbols (also called SubSet Tree (SST) in (Moschitti, 2006a)). For example, Figure 1(a) shows 10 STFs (out of 17) of the subtree rooted in `VP` (of the left tree). The STFs satisfy the constraint that grammatical rules cannot be broken. For example, `[VP [V NP]]` is an STF, which has two non-terminal symbols, `V` and `NP`, as leaves whereas `[VP [V]]` is not an STF. If we relax the constraint over the STFs, we obtain more general substructures called *partial trees fragments* (PTFs). These can be generated by the application of partial production rules of the grammar, consequently `[VP [V]]` and `[VP [NP]]` are valid PTFs. Figure 1(b) shows that the number of PTFs derived from the same tree as before is still higher (*i.e.* 30 PTs).

[1] Contract n. 33549



(a) Syntactic Tree fragments (STF)

(b) Partial Tree fragments (PTF)

Figure 1: Examples of different classes of tree fragments.

2.3 Counting Shared Subtrees

The main idea of tree kernels is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. To evaluate the above kernels between two T_1 and T_2, we need to define a set $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, *i.e.* a tree fragment space and an indicator function $I_i(n)$, equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over T_1 and T_2 is $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes, respectively and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_1) I_i(n_2)$. The latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes.

The algorithm for the efficient evaluation of Δ for the syntactic tree kernel (STK) has been widely discussed in (Collins and Duffy, 2002) whereas its fast evaluation is proposed in (Moschitti, 2006b), so we only describe the equations of the partial tree kernel (PTK).

2.4 The Partial Tree Kernel (PTK)

PTFs have been defined in (Moschitti, 2006a). Their computation is carried out by the following Δ function:

1. if the node labels of n_1 and n_2 are different then $\Delta(n_1, n_2) = 0$;
2. else $\Delta(n_1, n_2) =$
$1 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))$

where $\vec{I}_1 = \langle h_1, h_2, h_3, .. \rangle$ and $\vec{I}_2 = \langle k_1, k_2, k_3, .. \rangle$ are index sequences associated with the ordered child sequences c_{n_1} of n_1 and c_{n_2} of n_2, respectively, $\vec{I}_{1j}$ and $\vec{I}_{2j}$ point to the j-th child in the corresponding sequence, and, again, $l(\cdot)$ returns the sequence length, *i.e.* the number of children.

Furthermore, we add two decay factors: μ for the depth of the tree and λ for the length of the child subsequences with respect to the original sequence, *i.e.* we account for gaps. It follows that $\Delta(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1) = l(\vec{I}_2)} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big), \quad (1)$$

where $d(\vec{I}_1) = \vec{I}_{1l(\vec{I}_1)} - \vec{I}_{11}$ and $d(\vec{I}_2) = \vec{I}_{2l(\vec{I}_2)} - \vec{I}_{21}$. This way, we penalize both larger trees and child subsequences with gaps. Eq. 1 is more general than the Δ equation for STK. Indeed, if we only consider the contribution of the longest child sequence from node pairs that have the same children, we implement STK.

3 Generative Model: Stochastic Conceptual Language Model (SCLM)

The first step of our approach is to produce a list of SLU hypotheses using a Stochastic Conceptual Language Model. This is the same described in (Raymond and Riccardi, 2007) with the only difference that we train the language model using the SRILM toolkit (Stolcke, 2002) and we then convert it into a Stochastic Finite State Transducer (SFST). Such method allows us to use a wide group of language models, backed-off or interpolated with many kind of smoothing techniques (Chen and Goodman, 1998).

To exemplify our SCLM let us consider the following input italian sentence taken from the LUNA corpus along with its English translation:

Ho un problema col monitor.
(I have a problem with my screen).

A possible semantic annotation is:

null{*ho*} ***PROBLEM***{*un problema*} ***HARDWARE***{*col monitor*},

where ***PROBLEM*** and ***HARDWARE*** are two domain concepts and ***null*** is the label used for words not meaningful for the task. To associate word sequences with concepts, we use *begin*

(*B*) and *inside* (*I*) markers after each word of a sequence, *e.g.*:

null$\{ho\}$ ***PROBLEM-B***$\{un\}$ ***PROBLEM-I***$\{problema\}$ ***HARDWARE-B***$\{col\}$ ***HARDWARE-I***$\{monitor\}$

This annotation is automatically performed by a model based on a combination of three transducers:

$$\lambda_{SLU} = \lambda_W \circ \lambda_{W2C} \circ \lambda_{SLM},$$

where λ_W is the transducer representation of the input sentence, λ_{W2C} is the transducer mapping words to concepts and λ_{SLM} is the Stochastic Conceptual Language Model trained with SRILM toolkit and converted in FST. The SCLM represents joint probability of word and concept sequences by using the joint probability:

$$P(W, C) = \prod_{i=1}^{k} P(w_i, c_i | h_i),$$

where $W = w_1..w_k$, $C = c_1..c_k$ and $h_i = w_{i-1}c_{i-1}..w_1c_1$.

4 Discriminative re-ranking

Our discriminative re-ranking is based on SVMs trained with pairs of conceptually annotated sentences produced by the FST-based generative model described in the previous section. An SVM learn to classify which annotation has an error rate lower than the others so that it can be used to sort the *m*-best annotations based on their correctness. While for SVMs details we remaind to the wide literature available, for example (Vapnik, 1995) or (Shawe-Taylor and Cristianini, 2004), in this section we focus on hypotheses generation and on the kernels used to implement our re-ranking model.

4.1 Generation of m-best concept labeling

Using the FST-based model described in Section 3, we can generate the list of *m* best hypotheses ranked by the joint probability of the Stochastic Conceptual Language Model (SCLM). The Re-ranking model proposed in this paper re-ranks such list.

After an analysis of the *m*-best hypothesis list, we noticed that many times the first hypothesis ranked by SCLM is not the most accurate, *i.e.* the error rate evaluated with its Levenshtein distance from the manual annotation is not the lowest among the *m* hypotheses. This means that re-ranking hypotheses could improve the SLU accuracy. Intuitively, to achieve satisfactory results, different features from those used by SCLM should be considered to exploit in a different way the information encoded in the training data.

4.2 Structural features for re-ranking

The kernels described in previous sections provide a powerful technology for exploiting features of structured data. These kernels were originally designed for data annotated with syntactic parse trees. In Spoken Language Understanding the data available are text sentences with their semantic annotation based on basic semantic constituents. This kind of data has a rather flat structure with respect to syntactic parse trees. Thus, to exploit the power of kernels, a careful design of the structures used to represent data must be carried out, where the goal is to build tree-like annotation from the semantic annotation. For this purpose, we note that the latter is made upon sentence chunks, which implicitly define syntactic structures as long as the annotation is consistent in the corpus.

We took into account the characteristics of the presented kernels and the structure of semantic annotated data. As a result we designed the tree structures shown in figures 2(a), 2(b) and 3 for STK and PTK and sequential structures for SK defined in the following (where all the structures refer to the same example presented in Section 3, *i.e. Ho un problema col monitor*). The structures used with SK are:

($SK1$) ***NULL*** *ho* ***PROBLEM-B*** *un* ***PROBLEM-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-I*** *monitor*

($SK2$) ***NULL*** *ho* ***PROBLEM B*** *un* ***PROBLEM I*** *problema* ***HARDWARE B*** *col* ***HARDWARE I*** *monitor*,

For simplicity, from now on, the two structures will be referred as *SK1* and *SK2* (String Kernel 1 and 2). They differer in the use of chunk markers *B* and *I*. In *SK1*, markers are part of the concept, thus they increase the number of semantic tags in the data whereas in *SK2* markers are put apart as separated words so that they can mark effectively the beginning and the end of a concept, but for the same reason they can add noise in the sentence. Notice that the order of words and concepts is meaningful since each word is preceded by its corresponding concepts.

The structures shown in Figure 2(a), 2(b) and 3

have been designed for STK and PTK. They provide trees with increasing structure complexity as described in the following.

The first structure (FLAT) is a simple tree providing direct dependency between words and chunked concepts. From it, STK and PTK can extract relevant features (tree fragments).

The second structure (MULTILEVEL) has one more level of nodes and yields the same separation of concepts and markers shown in *SK1*. Notice that the same separation can be carried out putting the markers *B* and *I* as features at the same level of the words. This would increase exponentially (in the number of leaves) the number of subtrees taken into account by the STK computation. Since STK doesn't separate children, as described in Section 2.3, the structure we chose is lighter but also more rigid.

The third structure (FEATURES) is a more complex structure. It allows to use a wide number of features (like Word categories, POS tags, morpho-syntactic features), which are commonly used in this kind of task. As described above, the use of features exponentially increases the number of subtrees taken into account by kernel computations but they also increase the robustness of the model. In this work we only used Word Categories as features. They are domain independent, e.g. *"Months"*, *"Dates"*, *"Number"* etc. or POS tags, which are useful to generalize target words. Note also that the features in common between two trees must appear in the same child-position, hence we sort them based on their indices, *e.g.'F0*' for words and '*F1*' for word categories.

4.3 Re-ranking models using sequences

The FST generates the *m* most likely concept annotations. These are used to build annotation pairs, $\langle s^i, s^j \rangle$, which are positive instances if s^i has a lower concept annotation error than s^j, with respect to the manual annotation. Thus, a trained binary classifier can decide if s^i is more accurate than s^j. Each candidate annotation s^i is described by a word sequence with its concept annotation. Considering the example in the previous section, a pair of annotations $\langle s^i, s^j \rangle$ could be

s^i: ***NULL*** *ho* ***PROBLEM-B*** *un* ***PROBLEM-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-I*** *monitor*

s^j: ***NULL*** *ho* ***ACTION-B*** *un* ***ACTION-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-B*** *monitor*

where **NULL**, **ACTION** and **HARDWARE** are the assigned concepts. The second annotation is less accurate than the first since *problema* is erroneously annotated as ***ACTION*** and *"col monitor"* is split in two different concepts.

Given the above data, the sequence kernel is used to evaluate the number of common *n*-grams between s^i and s^j. Since the string kernel skips some elements of the target sequences, the counted *n*-grams include: concept sequences, word sequences and any subsequence of words and concepts at any distance in the sentence.

Such counts are used in our re-ranking function as follows: let e_k be the pair $\langle s_k^1, s_k^2 \rangle$ we evaluate the kernel:

$$\begin{aligned} K_R(e_1, e_2) &= SK(s_1^1, s_2^1) + SK(s_1^2, s_2^2) \\ &- SK(s_1^1, s_2^2) - SK(s_1^2, s_2^1) \end{aligned} \quad (2)$$

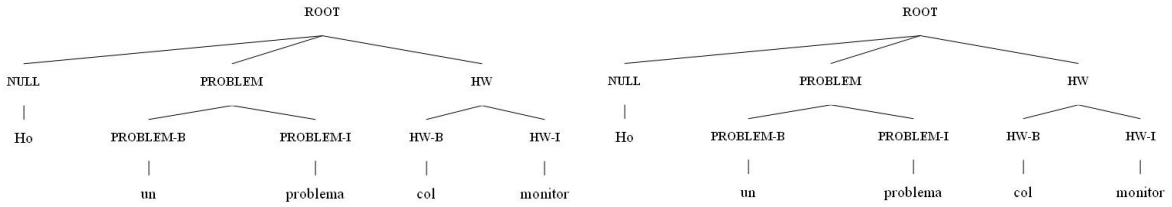
This schema, consisting in summing four different kernels, has been already applied in (Collins and Duffy, 2002; Shen et al., 2003) for syntactic parsing re-ranking, where the basic kernel was a tree kernel instead of SK. It was also used also in (Shen et al., 2004) to re-rank different candidates of the same hypothesis for machine translation. Notice that our goal is different from the one tackled in such paper and, in general, it is more difficult: we try to learn which is the best annotation of a given input sentence, while in (Shen et al., 2004), they learn to distinguish between *"good"* and *"bad"* translations of a sentence. Even if our goal is more difficult, our approach is very effective, as shown in (Dinarelli et al., 2009a). It is more appropriate since in parse re-ranking there is only one best hypothesis, while in machine translation a sentence can have more than one correct translations.

Additionally, in (Moschitti et al., 2006; Moschitti et al., 2008) a tree kernel was applied to semantic trees similar to the one introduced in the next section to re-rank Semantic Role Labeling annotations.

4.4 Re-ranking models using trees

Since the aim of concept annotation re-ranking is to exploit innovative and effective source of information, we can use, in addition to sequence kernels, the power of tree kernels to generate correlation between concepts and word structures.

Figures 2(a), 2(b) and 3 describe the structural association between the concept and the word



(a) *FLAT* Tree (b) *MULTILEVEL* Tree

Figure 2: Examples of structures used for STK and PTK

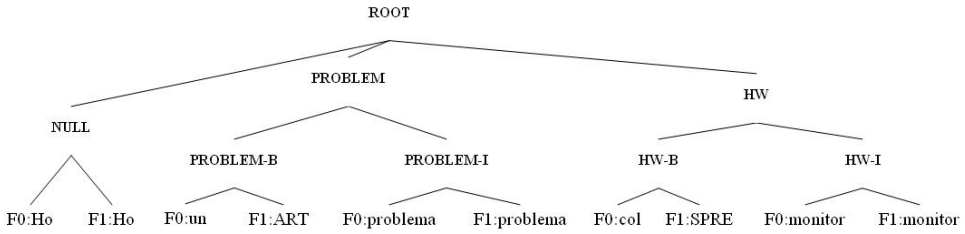


Figure 3: The *FEATURES* semantic tree used for STK or PTK

Corpus LUNA	**Train set**		**Test set**	
	words	**concepts**	**words**	**concepts**
Dialogs	183		67	
Turns	1.019		373	
Tokens	8.512	2.887	2.888	984
Vocab.	1.172	34	-	-
OOV rate	-	-	3.2%	0.1%

Table 1: Statistics on the LUNA corpus

Corpus Media	**Train set**		**Test set**	
	words	**concepts**	**words**	**concepts**
Turns	12,922		3,518	
# of tokens	94,912	43,078	26,676	12,022
Vocabulary	5,307	80	-	-
OOV rate	-	-	0.01%	0.0%

Table 2: Statistics on the MEDIA corpus

level. This kind of trees allows us to engineer new kernels and consequently new features (Moschitti et al., 2008), *e.g.* their subparts extracted by STK or PTK, like the tree fragments in figures 1(a) and 1(b). These can be used in SVMs to learn the classification of words in concepts.

More specifically, in our approach, we use tree fragments to establish the order of correctness between two alternative annotations. Therefore, given two trees associated with two annotations, a re-ranker based on tree kernel can be built in the same way of the sequence-based kernel by substituting SK in Eq. 2 with STK or PTK. The major advantage of using trees is the hierarchical dependencies between its nodes, allowing for the use of richer n-grams with back-off models.

5 Experiments

In this section, we describe the corpora, parameters, models and results of our experiments on re-ranking for SLU. Our baseline is constituted by the error rate of systems solely based on either FST or SVMs. The re-ranking models are built on the FST output, which in turn is applied to both manual or automatic transcriptions.

5.1 Corpora

We used two different speech corpora:

The LUNA corpus, produced in the homonymous European project, is the first Italian dataset of spontaneous speech on spoken dialogs. It is based on help-desk conversations in a domain of software/hardware repairing (Dinarelli et al., 2009b). The data is organized in transcriptions and annotations of speech based on a new multi-level protocol. Although data acquisition is still in progress, 250 dialogs have been already acquired with a WOZ approach and other 180 Human-Human (HH) dialogs have been annotated. In this work, we only use WOZ dialogs, whose statistics are reported in Table 1.

The corpus MEDIA was collected within the French project MEDIA-EVALDA (Bonneau-Maynard et al., 2005) for development and evaluation of spoken understanding models and linguistic studies. The corpus is composed of 1257 dialogs (from 250 different speakers) acquired with a Wizard of Oz (WOZ) approach in the context of hotel room reservations and tourist information.

Statistics on transcribed and conceptually annotated data are reported in Table 2.

5.2 Experimental setup

Given the small size of LUNA corpus, we did not carried out any parameterization thus we used default or a priori parameters. We experimented with LUNA and three different re-rankers obtained with the combination of SVMs with STK, PTK and SK, described in Section 4. The initial annotation to be re-ranked is the list of the ten best hypotheses output by an FST model.

We point out that, on the large Media dataset the processing time is considerably high[2] so we could not run all the models.

We trained all the SCLMs used in our experiments with the SRILM toolkit (Stolcke, 2002) and we used an interpolated model for probability estimation with the Kneser-Ney discount (Chen and Goodman, 1998). We then converted the model in an FST again with SRILM toolkit.

The model used to obtain the SVM baseline for concept classification was trained using YamCHA (Kudo and Matsumoto, 2001). As re-ranking models based on structure kernels and SVMs, we used the SVM-Light-TK toolkit (available at disi.unitn.it/moschitti). For λ (see Section 3), cost-factor and trade-off parameters, we used, 0.4, 1 and 1, respectively (*i.e.* the default parameters). The number m of hypotheses was always set to 10.

The CRF model we compare with was trained with the CRF++ tool, available at http://crfpp.sourceforge.net/. The model is equivalent to the one described in (Hahn et al., 2008). As features, we used word and morpho-syntactic categories in a window of [-2, +2] with respect to the current token, plus bigrams of concept tags (see (Hahn et al., 2008) and the CRF++ web site for more details).

Such model is very effective for SLU. In (Hahn et al., 2008), it is compared with other four models (Stochastic Finite State Transducers, Support Vector Machines, Machine Translation, Positional-Based Log-linear model) and it is by far the best on MEDIA. Additionally, in (Raymond and Riccardi, 2007), a similar CRF model was compared with FST and SVMs on ATIS and on a different version of MEDIA, showing again to be very effective.

[2]The number of parameters of the models and the number of training approaches make the exhaustive experimentation very expensive in terms of processing time, which would be roughly between 2 and 3 months of a typical workstation.

Structure	STK	PTK	SK
FLAT	18.5	19.3	-
MULTILEVEL	20.6	19.1	-
FEATURES	19.9	18.4	-
SK1	-	-	16.2
SK2	-	-	18.5

Table 3: CER of SVMs using STK, PTK and SK on LUNA (manual transcriptions). The Baselines, FST and SVMs alone, show a CER of **23.2%** and **26.3%**, respectively.

Model	MEDIA (CER)	LUNA (CER)
FST	13.7%	23.2%
CRF	11.5%	20.4%
SVM-RR (PTK)	12.1%	18.4%

Table 4: Results of SLU experiments on MEDIA and LUNA test set (manual transcriptions).

We ran SLU experiments on manual and automatic transcriptions. The latter are produced by a speech recognizer with a WER of 41.0% and 31.4% on the LUNA and the MEDIA test sets, respectively.

5.3 Training approaches

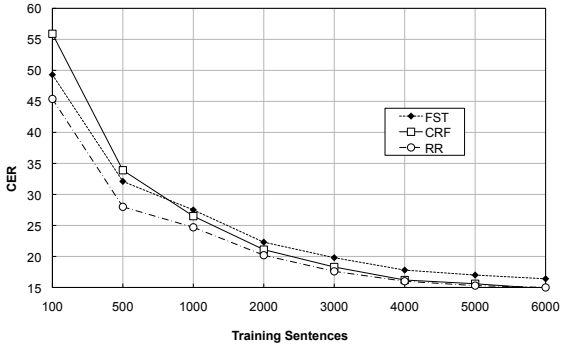
The FST model generates the *10*-best annotations, *i.e.* the data used to train the re-ranker based on SVMs. Different training approaches can be carried out based on the use of the data. We divided the training set in two parts. We train FSTs on part 1 and generate the *10*-best hypotheses using part 2, thus providing the first chunk of re-ranking data. Then, we re-apply these steps inverting part 1 with part 2 to provide the second data chunk. Finally, we train the re-ranker on the merged data.

For classification, we generate the *10*-best hypotheses of the whole test set using the FST trained on all training data.

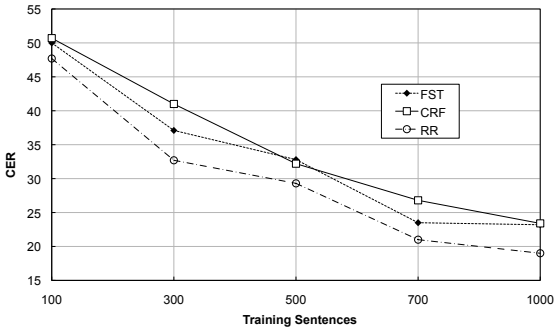
5.4 Re-ranking results

In Tables 3, 4 and 5 and Figures 4(a) and 4(b) we report the results of our experiments, expressed in terms of concept error rate (CER). CER is a standard measure based on the Levensthein alignment of sentences and it is computed as the ratio between inserted, deleted and confused concepts and the number of concepts in the reference sentence.

Table 3 shows the results on the LUNA corpus using the different training approaches, kernels and structures described in this paper. The



(a) Learning Curve on MEDIA corpus using the RR model based on SVMs and STK



(b) Learning Curve on LUNA corpus using the RR model based on SVMs and SK

Figure 4: Learning curves on MEDIA and LUNA corpora using FST, CRF and RR on the FST hypotheses

Model	MEDIA (CER)	LUNA (CER)
FST	28.6%	42.7%
CRF	24.0%	41.8%
SVM-RR (PTK)	25.0%	38.9%

Table 5: Results of SLU experiments on MEDIA and LUNA test set (automatic transcriptions with a WER 31.4% on MEDIA and 41% on LUNA)

dash symbol means that the structure cannot be applied to the corresponding kernel. We note that our re-rankers significantly improve our baselines, *i.e.* 23.2% CER for FST and 26.3% CER for SVM concept classifiers. For example, SVM re-ranker using SK, in the best case, improves FST concept classifier of 23.2-16.2 = 7 points.

Note also that the structures designed for trees yield quite different results depending on which kernel is used. We can see in Table 3 that the best result using STK is obtained with the simplest structure (FLAT), while with PTK the best result is achieved with the most complex structure (FEATURES). This is due to the fact that STK does not split the children of each node, as explained in Section 2.2, and so structures like *MULTILEVEL* and *FEATURES* are too rigid and prevent the STK to be effective.

For lack of space we do not report all the results using different kernels and structures on MEDIA, but we underline that as MEDIA is a more complex task (34 concepts in LUNA, 80 in MEDIA), the more complex structures are more effective to capture word-concept dependencies and the best results were obtained using the *FEATURES* tree.

Table 4 shows the results of the SLU experiments on the MEDIA and LUNA test sets using the manual transcriptions of spoken sentences and a re-ranker based on PTK and the *FEATURES* structure (already reported in the previous table). We used PTK since it is enough efficient to carry out the computation on the much larger Media corpus although as previously shown it is less accurate than SK.

We note that on a big corpus like MEDIA, the baseline models (FST and CRF) can be accurately learned thus less errors can be "corrected". As a consequence, our re-ranking approach does not improve CRF but it still improves the FSTs baseline of 1.6% points (11.7% of relative improvement).

The same behavior is reproduced for the SLU experiments on automatic transcriptions, shown in Table 5. We note that, on the LUNA corpus, CRFs are more accurate than FSTs (0.9% points), but they are significantly improved by the re-ranking model (2.9% points), which also improves the FSTs baseline by 3.8% points. On the MEDIA corpus, the re-ranking model is again very accurate improving the FSTs baseline of 3.6% points (12.6% relative improvement) on attribute annotation, but the most accurate model is again CRF (1% points better than the re-ranking model).

5.5 Discussion

The different behavior of the re-ranking model in the LUNA and MEDIA corpora is due partially to the task complexity, but it is mainly due to the fact that CRFs have been deeply studied and experimented (see (Hahn et al., 2008)) on MEDIA. Thus CRF parameters and features have been largely optimized. We believe that the re-ranking model can be relevantly improved by carrying out parameter optimization and new structural feature de-

sign.

Moreover, our re-ranking models achieve the highest accuracy for automatic concept annotation when small data sets are available. To show this, we report in Figure 4(a) and 4(b) the learning curves according to an increasing number of training sentences on the MEDIA and LUNA corpora, respectively. To draw the first plot, we used a re-ranker based on STK (and the FLAT tree), which is less accurate than the other kernels but also the most efficient in terms of training time. In the second plot, we report the re-ranker accuracy using SK applied to *SK1* structure.

In these figures, the FST baseline performance is compared with our re-ranking (RR) and a Conditional Random Field (CRF) model. The above curves clearly shows that for small datasets our RR model is better than CRF whereas when the data increases, CRF accuracy approaches the one of the RR.

Regarding the use of kernels two main findings can be derived:

- Kernels producing a high number of features, *e.g.* SK, produce accuracy higher than kernels less rich in terms of features, *i.e.* STK. In particular STK is improved by 18.5-16.2=2.3 points (Table 3). This is an interesting result since it shows that (a) a kernel producing more features also produces better re-ranking models and (b) kernel methods give a remarkable help in feature design.

- Although the training data is small, the re-rankers based on kernels appear to be very effective. This may also alleviate the burden of annotating large amount of data.

6 Conclusions

In this paper, we propose discriminative re-ranking of concept annotation to jointly exploit generative and discriminative models. We improve the FST-based generative approach, which is a state-of-the-art model in LUNA, by 7 points, where the more limited availability of annotated data leaves a larger room for improvement. Our re-ranking model also improves FST and CRF on MEDIA when small data sets are used.

Kernel methods show that combinations of feature vectors, sequence kernels and other structural kernels, *e.g.* on shallow or deep syntactic parse trees, appear to be a promising future research line[3]. Finally, the experimentation with automatic speech transcriptions revealed that to test the robustness of our models to transcription errors.

In the future we would like to extend this research by focusing on advanced shallow semantic approaches such as predicate argument structures, *e.g.* (Giuglea and Moschitti, 2004; Moschitti and Cosmin, 2004; Moschitti et al., 2008). Additionally, term similarity kernels, *e.g.* (Basili et al., 2005; Bloehdorn et al., 2006), will be likely improve our models, especially when combined syntactic and semantic kernels are used, i.e. (Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b).

References

Roberto Basili, Alessandro Moschitti, and Maria Teresa Pazienza. 1999. A text classifier based on linguistic processing. In *Proceedings of IJCAI 99, Machine Learning for Information Filtering*.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-2005*, Ann Arbor, Michigan.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In proceedings of CIKM '07*.

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. In *Proceedings of ICDM 06, Hong Kong, 2006*.

H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn, and D. Mostefa. 2005. Semantic annotation of the french media dialog corpus. In *Proceedings of Interspeech2005*, Lisbon, Portugal.

N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. 2003. Word sequence kernels. *J. Mach. Learn. Res.*, 3.

S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. In *Technical Report of Computer Science Group*, Harvard, USA.

[3] A basic approach is the use of part-of-speech tags like for example in text categorization (Basili et al., 1999) but given the high efficiency of modern syntactic parsers we can use the complete parse tree.

M. Collins and N. Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete structures, and the voted perceptron. In *ACL02*, pages 263–270.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL'04*.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2009a. Re-ranking models for spoken language understanding. In *Proceedings of EACL2009*, Athens, Greece.

Marco Dinarelli, Silvia Quarteroni, Sara Tonelli, Alessandro Moschitti, and Giuseppe Riccardi. 2009b. Annotating spoken dialogs: from speech segments to dialog acts and frame semantics. In *Proceedings of SRSL 2009 Workshop of EACL*, Athens, Greece.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge Discovery using Framenet, Verbnet and Propbank. In A. Meyers, editor, *Workshop on Ontology and Knowledge Discovering at ECML 2004*, Pisa, Italy.

Stefan Hahn, Patrick Lehnen, Christian Raymond, and Hermann Ney. 2008. A comparison of various methods for concept tagging for spoken language understanding. In *Proceedings of LREC*, Marrakech, Morocco.

T. Kudo and Y. Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL2001*, Pittsburg, USA.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML2001*, US.

Huma Lodhi, John S. Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2000. Text classification using string kernels. In *NIPS*.

Alessandro Moschitti and Adrian Bejan Cosmin. 2004. A semantic kernel for predicate argument classification. In *CoNLL-2004*, Boston, MA, USA.

Alessandro Moschitti and Silvia Quarteroni. 2008. Kernels on linguistic structures for answer extraction. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel joint inference. In *Proceedings of CoNLL-X*, New York City.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*, Prague, Czech Republic.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006a. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of ECML 2006*, pages 318–329, Berlin, Germany.

Alessandro Moschitti. 2006b. Making Tree Kernels Practical for Natural Language Learning. In *Proceedings of EACL2006*.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceeding of CIKM '08*, NY, USA.

C. Raymond and G. Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proceedings of Interspeech2007*, Antwerp,Belgium.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP'06*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184.

A. Stolcke. 2002. Srilm: an extensible language modeling toolkit. In *Proceedings of SLP2002*, Denver, USA.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.