JOINT GENERATIVE AND DISCRIMINATIVE MODELS FOR SPOKEN LANGUAGE UNDERSTANDING

Marco Dinarelli, Alessandro Moschitti, Giuseppe Riccardi
{dinarelli,moschitti,riccardi}@disi.unitn.it
DISI, Department of Engineering and Information Sciences
University of Trento, Italy

ABSTRACT

Spoken Language Understanding aims at mapping a natural language spoken sentence into a semantic representation. In the last decade two main approaches have been pursued: generative and discriminative models. The former is more robust to overfitting whereas the latter is more robust to many irrelevant features. Additionally, the way in which these approaches encode prior knowledge is very different and their relative performance changes based on the task.

In this paper we describe a training framework where both models are used: a generative model produces a list of ranked hypotheses whereas a discriminative model, depending on string kernels and Support Vector Machines, re-ranks such list. We tested such approach on a new corpus produced in the European LUNA project. The results show a large improvement on the state-of-the-art in concept segmentation and labeling.

Index Terms — Spoken Language Understanding, Generative and Discriminative Models, Stochastic Language Models, Kernel Methods, Finite State Transducers

1. INTRODUCTION

In Spoken Dialog Systems, the Language Understanding module performs the task of translating a spoken sentence into its meaning representation based on semantic constituents. These are often referred to as concepts and are instantiated by sequences of words. Therefore, a Spoken Language Understanding (SLU) module finds the association between words and concepts.

In the last decade two major approaches have been proposed to find this correlation: (i) generative models, whose parameters refer to the joint probability of concepts and constituents; and (ii) discriminative models, which learn a classification function to map words into concepts based on geometric and statistical properties. An example of simple and effective generative model is the one based on Finite State Transducers. It performs SLU as a translation process based on FST. An example of discriminative model used for SLU is the one based on Support Vector Machines (SVMs) [10], as shown in [1]. In this approach, data are mapped into a vector space and SLU is performed as a classification problem using Maximal Margin Classifiers [9].

Generative models have the advantage to be more robust to overfitting, while discriminative models are more robust to irrelevant features. Both approaches, used separately, have shown a good performance [1], but the way these approaches encode prior knowledge is very different, thus designing models able to take into account characteristics of both of them 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. These exploit all word subsequences (with gaps) of the spoken sentence (i.e. all n-grams) as features. Gaps allow for the encoding of long distance dependencies between words in relatively small n-grams. Given the huge size of this feature space, we adopted kernel methods and in particular sequence kernels [9] to implicitly encode n-grams in SVMs.

We experimented with different approaches for training the discriminative models on a new corpus acquired in the European project LUNA[1] and composed of dialogs recorded with a Wizard of Oz (WOZ) approach. The results show a great improvement with respect to both the FST-based model and the SVM model alone, which are the current state-of-the-art for concept classification on more well known corpora like MEDIA and ATIS [1]. The rest of the paper is organized as follows: sections 2 and 3 show the generative and discriminative models, respectively. The experiments and results are reported in Section 4 whereas the conclusions are drawn in Section 5.

2. GENERATIVE APPROACH FOR CONCEPT CLASSIFICATION

In the context of Spoken Language Understanding (SLU), concept classification is the task of associating the best sequence of concepts to a given sentence, i.e. word sequence. A concept represents the class of the words expressing the same domain semantic. In SLU, concepts are used as semantic units and are represented with concept tags. The association between words and concepts is learned from an annotated corpus.

[1] Contract n. 33549

The Generative model used in our work for concept classification is the same used in [1]. Given a sequence of words as input, a translation process is performed to output a sequence of concept tags. The translation process involves three steps: (1) the mapping of words into classes, (2) the mapping of classes into concepts and (3) the selection of the best concept sequence.

The first step is used to improve the generalization power of the model. The classes at this level can be both domain-dependent, for example Software in LUNA corpus**,** or domain-independent, e.g. numbers, dates etc. The class of a word not belonging to any class is the word itself.

In the second step, classes are mapped into concepts, where a sequence of words may be associated with more than one concept, i.e. more than one SLU hypothesis.

In the third step, the best or the *m*-best hypotheses are selected among those produced in the previous step. They are chosen according to the maximum probability evaluated by the Conceptual Language Model, described in the next section.

2.1 Stochastic Conceptual Language Model (SCLM)

An SCLM is an n-gram language model built on semantic tags. Using the same notation proposed in [3] and [1], our SCLM trains joint probability *P*(*W,C*) of word and concept sequences from an annotated corpus:

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

$$where: \quad W = w_1,\ldots,w_k \quad C = c_1,\ldots,c_k \quad h_i = \{w_{i-1}c_{i-1},\ldots,w_1c_1\}$$

Since, we use a 3-gram conceptual language model, the history h_i is $\{w_{i-1}\, c_{i-1},\, w_{i-2}\, c_{i-2}\}$.

All the steps of the translation process described here and above are implemented as Finite State Transducers (FST) using the AT&T FSM/GRM tools [4] and the SRILM [6] tools. In particular the SCLM is trained using SRILM tools and then converted in an FST using scripts provided with SRILM toolkit. By representing the combination of all the translation steps as a transducer λ_{SLU} [1] in terms of FST operations:

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

where λ_W is the transducer representation of input sentence, λ_{W2C} is the transducer mapping words to classes and λ_{SLM} is the Semantic Language Model (SLM) described above, the best SLU hypothesis is given by

$$\hat{C} = project_C(bestpath(\lambda_{SLU}))$$

where *bestpath* performs a viterbi search on the FST and *project* performs a projection of the FST on the output labels, in this case the concepts.

2.2. Generation of m-best concept labeling

Using the FSTs described above, we can generate *m* best hypotheses ranked by the joint probability of the SCLM.

After an analysis of the *m*-best hypotheses of our SLU model, we noticed that many times the first hypothesis is not the closest to the correct concept sequence, i.e. its error rate is not the lowest among the *m* hypotheses. This means that re-ranking the *m*-best hypotheses in a convenient way could improve the SLU performance. The best choice in this case is a discriminative model, since it allows for the use of informative features, which, in turn, can model easily feature dependencies (also if they are infrequent in the training set).

3. DISCRIMINATIVE RE-RANKING

Our discriminative re-ranking is based on SVMs trained with pairs of conceptually annotated sentences. The classifiers learn to select which annotation has an error rate lower than the others so that the m-best annotations can be sorted based on their correctness.

3.1 SVMs and Kernel Methods

Kernel Methods refer to a large class of learning algorithms based on inner product vector spaces, among which Support Vector Machines (SVMs) are one of the most well-known algorithm. SVMs learn a hyperplane $H(\vec{x}) = \vec{w} \cdot \vec{x} + b = 0$, where $\vec{x}$ is the feature vector representation of a classifying object o, $\vec{w} \in \Re^n$ (a vector space) and $b \in \Re$ are parameters [10]. The classifying object o is mapped in $\vec{x}$ by a feature function ϕ. The kernel trick allows us to rewrite the decision hyperplane as $\sum_{i=1..l} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b = 0$, where y_i is equal to 1 for positive and -1 for negative examples, $\alpha_i \in \Re^+$, $o_i\ \forall i \in \{1,..,l\}$ are the training instances and the product $K(o_i, o) = \langle \phi(o_i) \cdot \phi(o) \rangle$ is the kernel function associated with the mapping ϕ. Note that, we do not need to apply the mapping ϕ, we can use $K(o_i, o)$ directly [9]. For example, next section shows a kernel function that counts the number of word sequences in common between two sentences, in the space of n-grams (for any n).

3.2 String Kernels

The string kernels (SK) that we consider count the number of word sequences shared by two input sequences, considering also gaps, i.e. some of the words of the original sentence are skipped. In our case the sequences are pair of annotated sentences, i.e. the hypotheses generated by the FST-based model. Let V be a word vocabulary, a word sequence $s \in V^*$ is described by $s = w_1..w_{|s|}$, $s[i:j]$ selects the subsequence $w_i w_{i+1}..w_{j-1} w_j$ from the i-th to the j-th word. A subsequence u of s is associated with a sequence of indexes $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \le i_1 < ... < i_{|u|} \le |s|$, such that $u = w_{i_1}..w_{i_{|u|}}$ or $u = s[\vec{I}]$ for short. $d(\vec{I})$ is the distance between the first and last word of the subsequence u in s, i.e. $d(\vec{I}) = i_{|u|} - i_1 + 1$. A sequence kernel is defined as:

$$SK(s_1,s_2) = \sum_{u \in V^*} \phi(s_1) \cdot \phi(s_2) = \sum_{u \in V^*} \sum_{\vec{I}_1 : u = s_1[\vec{I}_1]} \lambda^{d(\vec{I}_1)}$$

$$\sum_{\vec{I}_2 : u = s_2[\vec{I}_2]} \lambda^{d(\vec{I}_2)} = \sum_{u \in V^*} \sum_{\vec{I}_1 : u = s_1[\vec{I}_1]} \sum_{\vec{I}_2 : u = s_2[\vec{I}_2]} \lambda^{d(\vec{I}_1) + d(\vec{I}_1)}$$

Corpus LUNA	Train set		Test set	
	words	concepts	words	concepts
Dialogs Woz	183		67	
Dialogs HH	180		-	
Turns Woz	1,019		373	
Turns HH	6,999		-	
Tokens Woz	8,512	2,887	2,888	984
Tokens HH	62,639	17,423	-	-
Vocabul. Woz	1,172	34	-	-
Vocabul. HH	4,692	49	-	-
OOV rate	-	-	3.2%	0.1%

Table 1 Statistics on the LUNA corpus

where $\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{d(\vec{I})}$ for some $0 < \lambda \leq 1$ counts the number of occurrences of u in the sequence s and assigns them the sequence weight $\lambda^{d(\vec{I})}$ proportional to its length. Hence, the above equation returns the sum of all common subsequences weighted according to their frequency of occurrences and lengths d. It is worth to note that: (a) longer subsequences receive lower weights. (b) Any word subsequence of the original sentence is valid (some words can be omitted, i.e. gaps). (c) Gaps determine a weight since $d(.)$ is the number of words and gaps between the first and last word. 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.

3.3 Re-Ranking Model

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 word error rate than s^j, with respect to the manual annotation in the corpus and are negative instances otherwise. Thus, a binary classifier trained on such examples can decide if s^i is more accurate that s^j.

A candidate annotation s^i is described by a word sequence where each word is followed by its concept annotation. For example, given the sentence:

ho (*I have*) *un* (*a*) *problema* (*problem*) *con* (*with*) *la* (*the*) *scheda di rete* (*network card*) ora (*now*),

a pair of annotations $\langle s^i, s^j \rangle$ could be:

s^i: *ho* ***Null*** *un* ***Null*** *problema* ***Problem-B*** *con* ***Null*** *la* ***Null*** *scheda-**HW-B** di **HW-I** rete **HW-I** ora* ***RelativeTime-B***

s^j: *ho* ***Null*** *un* ***Null*** *problema* ***Action-B*** *con* ***Null*** *la* ***Null*** *scheda-**HW-B** di **HW-B** rete **HW-B** ora* ***RelativeTime-B***

where ***Null, action, RelativeTime,*** and ***HW*** are the assigned concepts.

Additionally, we can encode the word-chunking information in the sequence by means of the tags ***B*** and ***I***, i.e. the usual begin and internal tags for concept subparts.

Note that, the second annotation is less accurate than the first since, in it, *problema* is incorrectly annotated as an action and "*scheda di rete*" is incorrectly split in three different concepts.

Given the above representation, the sequence kernel can be used to evaluate the number of common n-grams[2] (with gaps) between s^i and s^j. More specifically, let e_i be the pair $\langle s_i^1, s_i^2 \rangle$, where s^1 and s^2 are two annotated sentences. We used the following re-ranking kernel:

$$K_R(e_1, e_2) = SK(s_1^1, s_2^1) + SK(s_1^2, s_2^2) - SK(s_1^1, s_2^2) - SK(s_1^2, s_2^1)$$

This schema consisting in summing four different kernels has been already applied in [2] for syntactic parsing re-ranking, where the basic kernel was a tree kernel instead of SK.

4. EXPERIMENTS

In this section, we describe the corpora, parameters, models and the results of our experiments. Our baseline relates to FST and SVM error rate. The re-ranking model is based on the FST output. Different ways of producing training data determine different results.

4.1. Corpora

In order to test our model we used a new corpus:

The corpus LUNA acquired in the related European project LUNA is the first Italian corpus of spontaneous speech: it reports the help-desk conversation in the domain of software/hardware repairing [5]. The data are organized in transcriptions and annotations of speech based on a new multi-level protocol. Data acquisition is still in progress. Currently, 250 dialogs acquired with a WOZ approach and of 180 Human-Human (HH) dialogs are available. Statistics on LUNA corpus are reported in Table 1. We report statistics on the entire corpus for completeness, but we run our experiments using only the WOZ dialogs.

4.2. Experimental Setup

We split the LUNA corpus in training and test set. Given the small size of LUNA corpus, we did not carried out parameterization on a development set but we used default or a priori parameters.

We trained all the SCLMs used in our experiments with the SRILM toolkit [6] and we used an interpolated model for probability estimation with the Kneser-Ney discount [7]. We then converted the model in an FST as described in Section 2.1. The model used to obtain the SVM baseline for concept classification was trained using YamCHA [8]. For the re-ranking model using string kernel we used the SVM-Light-TK toolkit [12] (available at dit.unitn.it/moschitti). For λ (see Section 3.2), cost-factor and trade-off parameters, we used, 0.4, 1 and 1, respectively.

4.3. Training approaches

The FST model generates the m-best annotations, i.e. the data used to train the re-ranker based on SVMs and SK. Different training approaches can be carried out according

[2] In our experiments n is from 1 up to 3 but higher values can be used.

Corpus	**WOZ**	
Approach	**MT**	**ST**
FST	**23.2%**	**23.2%**
SVM	26.7%	26.7%
RR - A	19.1%	**16.1%**
RR - B	**19.0%**	19.0%
RR - C	19.1%	16.6%

Table 2 Results of experiments (CER) using FST and SVM on the LUNA WOZ corpus

to the use of the corpus and the method to generate the *m*-best. We apply two different methods for training: ***Monolithic Training*** and ***Split Training***.

In the former, FSTs are learned with the whole training set. The *m*-best hypotheses generated by such models are then used to train the re-ranker classifier.

In Split Training, the training data are divided in two parts to avoid bias in the FST generation step. More in detail, we train FSTs on part 1 and generate the *m*-best hypotheses using part 2.

Then, we re-apply these procedures inverting part 1 with part 2. Finally, we train the re-ranker on the merged *m*-best data. At the classification time, we generate the *m*-best of the test set using the FST trained on all training data.

Regarding the generation of the training instances $\langle s_i, s_j \rangle$, we set *m* to 10 and we choose one of the 10-best hypotheses as the second element of the pair, s_j, thus generating 10 different pairs. The first element instead can be selected according to three different approaches: (A): s_i is the manual annotation taken from the corpus; (B) s_i is the most accurate annotation, in terms of the edit distance from the manual annotation, among the 10-best hypotheses of the FST model; and (C) as for (B) but s_i is selected among the 100-best hypotheses. The pairs are also inverted to generate negative examples

4.4. Results

All the results of our experiments, expressed in terms of concept error rate (CER), are reported in Table 2. In the header are reported the corpus and the training approach used, i.e. Monolithic Training (MT) and Split Training (ST). Column 1 shows the concept classification model used, i.e. the baselines FST and SVM, and the re-ranking models (RR) applied to FST. A, B and C refer to the three approaches for generating training instances described above.

We note that: first, using our corpus of WOZ dialogs, the baseline CER of FST and SVMs models are 23.2% and 26.7%, respectively.

Second, the re-ranking models using different training approaches, A, B and C, produce a remarkable improvement on LUNA WOZ, e.g. form 23.2% to 16.1% using s*plit training* and A method for data generation. This corresponds to an enhancement of roughly 30% of the baseline.

Finally, since our corpus is new no previous result is available for comparison with other approaches. However, we computed our baseline with the SVM and FST models, which have been shown to reach state-of-the-art results [1][11].

5. CONCLUSIONS

In this paper, we propose discriminative re-ranking of concept annotation to capitalize from the benefits of generative and discriminative approaches. Our generative model (FST) is the state-of-the-art in concept classification as shown in [1]. We show that, using our re-ranking model, it can be improved by 7 points (until 30% of relative improvement).

It should be noted that the design of our re-ranker is only based on a sequence kernel. Kernel Methods have shown that combinations of feature vectors, sequence kernels and other structured kernels, e.g. on shallow or deep syntactic parse trees, provide much more accurate model. Thus the design of more complex kernels appears to be a promising research line. Also, the experimentation with automatic speech transcriptions is interesting since it can show if our approach is robust to transcription errors.

Finally, as future work, we would like to apply our model on well-known corpora such as MEDIA and ATIS to provide results directly comparable with other models, in particular with the methods used in [11].

REFERENCES

[1] C. Raymond, and G. Riccardi, “Generative and Discriminative Algorithms for Spoken Language Understanding”, Interspeech, Antwerp, 2007.

[2] M. Collins and N. Duffy, “New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron,” in *ACL’02*, 2002.

[3] A. Moschitti, G. Riccardi, and C. Raymond, “Spoken Language Understanding with Kernenls for Syntactic/Semantic Structures”, ASRU, Kyoto, 2007.

[4] M. Mohri, F. Pereira, M. Riley, “Weighted finite-state transducers in speech recognition”, Computer, Speech and Language, 2002.

[5] C. Raymond, G. Riccardi, K. J. Rodrigez, and J. Wisniewska, “The LUNA Corpus: an Annotation Scheme for a Multi-domain Multi-lingual Dialogue Corpus”, Decalog, Trento, 2007.

[6] A. Stolcke, “SRILM: an Extensible Language Modeling Toolkit”, in Proc. Intl. Conf. SLP, Denver, 2002.

[7] S. F. Chen, and J. Goodman, “An Empirical Study of Smoothing Techniques for Language Modeling”, Tech. Report Computer Science Group, Harvard, 1998.

[8] T. Kudo, and Y. Matsumoto, “Chunking with Support Vector Machines”, NAACL, Pittsburg, 2001.

[9] J. Shawe-Taylor and N. Cristianini, “Kernel Methods for Pattern Analysis”, Cambridge University Press, 2004.

[10] V. Vapnik, “The Nature of Statistical Learning Theory”, Springer, 1995.

[11] S. Hahn, P. Lehnen, C. Raymond, H. Ney, „A Comparison of Various M Methods for Concept Tagging for Spoken Language Understanding“, LREC, Marrakech, 2008.

[12] Alessandro Moschitti, Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.