Ph.D. Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Spoken Language Understanding: From Spoken Utterances to Semantic Structures

Marco Dinarelli

Advisor:

Prof. Giuseppe Riccardi

Università degli Studi di Trento

January 2010

Abstract

In the past two decades there have been several projects on Spoken Language Understanding (SLU). In the early nineties DARPA ATIS project aimed at providing a natural language interface to a travel information database. Following the ATIS project, DARPA Communicator project aimed at building a spoken dialog system automatically providing information on flights and travel reservation. These two projects defined a first generation of conversational systems. In late nineties "How may I help you" project from AT&T, with Large Vocabulary Continuous Speech Recognition (LVCSR) and mixed initiatives spoken interfaces, started the second generation of conversational systems, which later have been improved integrating approaches based on machine learning techniques.

The European funded project LUNA aims at starting the third generation of spoken language interfaces. In the context of this project we have acquired the first Italian corpus of spontaneous speech from real users engaged in a problem solving task, as opposed to previous projects. The corpus contains transcriptions and annotations based on a new multilevel protocol studied specifically for the goal of the LUNA project.

The task of Spoken Language Understanding is the extraction of the meaning structure from spoken utterances in conversational systems. For this purpose, two main statistical learning paradigms have been proposed in the last decades: generative and discriminative models. The former are robust to over-fitting and they are less affected by noise but they cannot easily

integrate complex structures (e.g. trees). In contrast, the latter can easily integrate very complex features that can capture arbitrarily long distance dependencies. On the other hand they tend to over-fit training data and so they are less robust to annotation errors in the data needed to learn the model.

This work presents an exhaustive study of Spoken Language Understanding models, putting particular focus on structural features used in a Joint Generative and Discriminative learning framework. This combines the strengths of both approaches while training segmentation and labeling models for SLU. Its main characteristic is the use of Kernel Methods to encode structured features in Support Vector Machines, which in turn re-rank the hypotheses produced by an first step SLU module based either on Stochastic Finite State Transducers or Conditional Random Fields. Joint models based on transducers are also amenable to decode word lattices generated by large vocabulary speech recognizers.

We show the benefit of our approach with comparative experiments among generative, discriminative and joint models on some of the most representative corpora of SLU, for a total of four corpora in four different languages: the ATIS corpus (English), the MEDIA corpus (French) and the LUNA Italian and Polish corpora (Italian and Polish respectively). These also represent three different kinds of domain applications, i.e. informational, transactional and problem-solving domains. The results, although depending on the task and in some range on the first model baseline, show that joint models improve in most cases the state-of-the-art, especially when a small training set is available.

Keywords
Machine Learning, Kernel Methods, Data-driven approaches, Stochastic models, Text Processing, Semantic Structures.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In last decades Information Technology has been the main topic in most research centers of the world. Human-Computer interaction is becoming more and more an easy available technology in everyday life. This thanks to impressive advancements in research topics like Machine learning, Automatic Speech Recognition and Understanding, Data-driven stochastic approaches, Text Processing and Speech Synthesis.

One class of applications relying heavily on this technologies is Spoken Dialog Systems. Spoken Dialog Systems allow humans to engage complex dialogs with machines using the most natural communication mean ever known: their voice. In the last decades, this class of applications has been made able to interact with humans and satisfy their needs in such a way that allows hoping that this technology will be soon available to everybody and in any kind of device, from the most powerful desktop PC to the cheapest laptops and mobile phones.

1.1 The Context

A Spoken Dialog System is composed of several modules which aim is to allow a user to engage a dialog with a machine in order to solve a problem or to retrieve some information. The modules are involved in the dialog one

at a time and work in sequence like a pipeline: the output of a module is the input of another one. In most cases each module processes the input using some prior acquired information and possibly other external information. Each module of a Spoken Dialog System is very complex and it took years of research and hard work to provide solutions that can be exploited in real applications.

This work focus on a particular module of a Spoken Dialog System application: the Spoken Language Understanding module. From a high level point of view, the Spoken Language Understanding module takes as input the transcription of a user utterance, based on words, and output its interpretation, based on concepts. The interpretation is used in the following modules of the system as a high-level layer of the information provided by the user. As it will be shown, providing an accurate interpretation of a user utterance is a hard problem that involves complex theories and technologies.

1.2 The Problem

Designing a Spoken Language Understanding module requires learning how words can be associated to concepts. Learning this association requires the availability of manually annotated data, specific for the application. These provides examples of how to map words into concepts. Words are the basic constituents of user utterance transcriptions. A Spoken Language Understanding module must be able to deal with transcriptions that can be generated in two different ways: manually, as transcriptions of human experts, or automatically using an Automatic Speech Recognition (ASR) system. The latter takes as input a speech file and produces the corresponding transcription as a text sentence. Alternatively ASR systems can generate a set of transcriptions from the same utterance. These are usually

encoded in a compact graph representation called "word lattice". Going from manual transcriptions to lattices, the SLU task complexity increases.

Manual transcriptions are used only for development and evaluation purposes, together with words they contain annotation of spontaneous speech phenomena and possibly punctuation. All this information can be used by a SLU module to increase the interpretation accuracy. Satisfactory performances are reached in most cases on this type of input. The only difficulties are related to manual data annotation errors and data sparseness.

Automatic transcriptions, as output of automatic systems, contain transcription errors. Further, current ASR systems are not able to exploit effectively spontaneous speech phenomena annotations and punctuation. Thus ASR output is a raw sequence of words without boundary and structure information. All these points make the SLU task much harder on automatic transcriptions. Since the aim of SLU is to be applied in Spoken Dialog System, an effective SLU module must be robust to ASR output.

Further noise is added when using ASR lattices as input to the SLU module. A lattice is a word graph containing all the possible transcriptions of a given utterance. The interesting aspect when using lattices is the fact that, among the huge space of possibilities, a correct transcription can be found. This goal is very difficult and has not been reached yet. The complexity of the SLU task, when using lattices as input, is related to the fact that also the SLU module can generate many different interpretations from the same sentence. Thus, applying SLU to lattices, transcription possibilities are multiplied by the interpretation possibilities. Discriminating correct interpretations in such huge space is very hard.

Words composing the transcription of a user utterance are associated to concepts. Concepts are basic semantic constituents taken from a predefined semantic representation. The difficulties involved in both learning this association and providing the application specific data are not trivial and

will be addressed extensively in this dissertation.

In the last decades the interpretation task tackled in the Spoken Language Understanding module has been based on stochastic approaches. A big effort has been done to find continuously more accurate models and in the last few years, the combination of different stochastic models has been successfully applied to this task.

1.3 The Solution

Despite the relatively good accuracy provided by the proposed models, Spoken Language Understanding remains a very complex problem when applied to real applications. In this respect, models combination seems to provide a more robust solution.

This work describes the study, the implementation and the evaluation of an approach based on the combination of different models using several information sources. The intuition behind this solution is that the different models used for Spoken Language Understanding, although providing similar performances, have very different characteristics that are someway complementary. Finding a suitable way to put together these characteristics results in an improved approach bringing together advantages of the combined models.

Our solution for models combination is based on well-known models and technologies, for instance the first model involved in the combination can be based on Weighted (or Stochastic) Finite State Transducers or Conditional Random Fields, while the second model is based on Support Vector Machines and kernel methods. The first generates a list of interpretations, the m-best hypotheses list. The second re-rank, i.e. re-sort, the list based on a different metric, providing possibly an interpretation more correct than the one provided by the first model alone.

Our main contribution is the implementation of a module working as junction between the two components of the joint models. The module takes as input the interpretations generated by the first model and convert them into tree-shaped structures. These structures can be augmented with additional information, e.g. coming from the application knowledge base, and coupled with other kinds of structures, e.g. syntactic parses. Interpretation hypotheses to be re-ranked can be selected based on a new semantic inconsistency metric. The metric is important for two reasons: first it allows to select hypotheses among those generated by the first model, instead of taking all its hypotheses ranked by the model probability. Second, we can generate a huge number of hypotheses with the first model, but only the m-best under this metric are kept to be converted into tree structures and re-ranked. Given the semenatic inconsistency metric, these hypotheses are likely to contain less errors. The resulting forest of structures is used as input by the second model, that can learn words to concepts association exploiting many sources of information. The second model also learns from the interpretations of the first model. Using this solution we can create an SLU system putting together benefits of both individual models. Our module can process all types of input used in SLU: manual and automatic transcriptions as well as ASR lattices.

We propose as well a post-processing phase on the joint models output. This module learns the correlation between interpretation correctness and posterior probabilities provided by the two combined models. This way the module can eventually choose the most correct interpretation between those provided by the two individual models. This goes beyond the traditional solution for the same kind of models combination.

We have evaluated our approach on four different corpora in four different languages: English, French, Italian and Polish. These corpora are among the most significant tasks for Spoken Language Understanding. We

provide comparative evaluations with the best SLU models on the same tasks.

We also integrated our Spoken Language Understanding models into a new Spoken Dialog System prototype for complex problem solving tasks. The system is an evolution of a call routing application: Initial call classification is done in parallel with our statistical Spoken Language Understanding module. The call-type class must agree with the SLU interpretation on the first user turn in order to move the dialog to the next step. The rest of the dialogue is used to elicit further task-relevant details from the user before passing on the call. The dialogue capability also allows us to obtain clarifications of the initial classifier guess. Based on an evaluation, we show that conducting a dialogue significantly improves upon call routing based on call classification alone. We present both subjective and objective evaluation results of the system according to standard metrics on real users.

1.4 Novel Contributions

The proposed approach outperforms in most cases the best state-of-the-art models on all the applications studied here. This is an even more important achievement taking into account significant and continuous improvements yielded on the same tasks by other researchers during our work.

Our solution is robust with respect to different sources of noise typical of the Spoken Language Understanding task and it can learn words to concepts association using very complex and structured features that we designed first.

We defined a new hypotheses selection criterion based on a semantic inconsistency metric specific for SLU. It exploits the semantic representation used in SLU to detect when it is likely that an interpretation contains

inconsistent concept names and values. The metric makes it possible combining SLU models also when their characteristics are not really complementary, e.g. Conditional Random Fields and Support Vector Machines are both discriminative models. The inconsistency metric is computed on interpretation hypotheses generated by the first model. It is used to choose more correct hypotheses to be converted in tree structures and then passed to the second model, thus enforcing the role played by the junction component. We designed an effective post-processing module based on a-posteriori probabilities provided by the combined models.

Since the approach involves several steps of processing, it can be easily extended and improved in each of its steps.

We implemented a state-of-the-art Spoken Dialog System integrating our SLU joint models. Other innovative aspects are proposed in the system: persistent dialog state, parallel SLU interpretation and call-type classification.

1.5 Structure of the Thesis

This dissertation is structured as follows: in Chapter 2 we give an introduction to the SLU task, in Chapter 3 we describe all state-of-the-art models used for Spoken Language Understanding, Chapter 4 describes one of our main contribution to SLU: the joint models studied during our research work. Chapter 5 describes another main contribution of our work: two improvement strategies studied and implemented for our joint models. Corpora, experimental setup and experiment results are given in Chapter 6. Chapter 7 describes the Spoken Dialog System (SDS) implemented during our research work in which the SLU models described in previous chapters have been integrated. Conclusions follow in Chapter 8.

Chapter 2

Overview of Spoken Language Understanding

Spoken Language Understanding (SLU) is the semantic interpretation of signs conveyed by a speech signal. The goal of SLU is to extract a conceptual representation from spoken sentence transcriptions in a natural language. This task is very complex. Signs to be used to produce conceptual representation are coded in the signal along with other noisy information. Spoken sentences many times don't follow the grammar of a language, they could contain self correction, hesitations, repetitions and many other irregular phenomena due to the spontaneous nature of spoken language. Furthermore SLU systems are applied to the output of an Automatic Speech Recognizer (ASR) so they must be robust to noise introduced by spontaneous events typical of spoken language and errors introduced by ASR. ASR components produce a stream of words with no information about sentence structure, like punctuation and sentence boundaries, so SLU systems must perform text segmentation and understanding at the same time. The level of complexity needed in order to represent the meaning of a spoken utterance depends mainly on the application targeted.

The application we consider is Spoken Dialog, in particular we focus on the understanding module of this class of applications. A high level

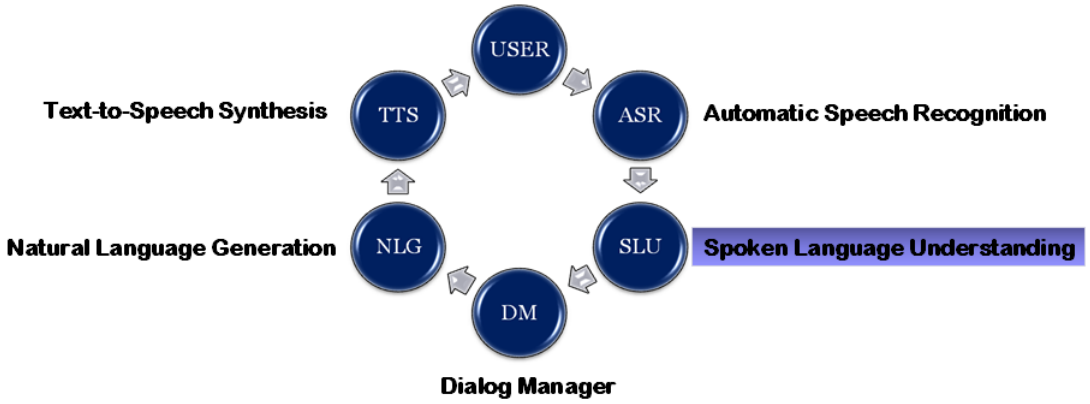


Figure 2.1: High level architecture of a Spoken Dialog System application, the SLU module has been highlighted since it is the main topic of this dissertation

schema of a Spoken Dialog System application is shown in Figure 2.1. The dialog is initiated by the user as response to an opening prompt from the system. The user utterance is automatically transcribed by the Automatic Speech Recognition (ASR) component. The ASR takes as input a speech signal and produces its transcription in textual format. The Spoken Language Understanding (SLU) module takes as input the output of ASR and generates a meaning representation. Based on the interpretation coming from the SLU module, the Dialog Manager (DM) select the next dialog turn, this is converted into a natural language sentence by the Natural Language Generation (NLG) module. Finally, the Text-To-Speech (TTS) module synthsizes the generated sentence as a speech signal, which is sent back to the user to continue the dialog. The loop depicted in Figure 2.1 is repeated until the application completes the modelled task.

Spoken dialog systems need sophisticated SLU models in order to implement dialog applications that go beyond solving simple tasks like call routing or form filling [35]. Three level of complexity can be defined for dialog applications. The first level involves the translation from words to basic conceptual constituents. The second level includes semantic compo-

sition on the basic constituents yielded in the first level. At the third level context-sensitive validation is performed. At this level each utterance is considered as a set of sub-utterances and a broad context is taken into account. The interpretation of a sub-utterance is performed context sensitive to the others. Going from level one to level three, tasks of increasing complexity can be solved, from call routing or classification of utterances [35] to Help-Desk application for hardware and software repairing [31].

SLU is performed as a semantic parsing of spoken sentences. Approaches based on syntactic analisys or directly on semantic analisys have been proposed, in any case semantic constituents are instantiated by one or more words that have a corresponding syntactic constituent. Among understanding modules proposed in the last two decades, first solutions were based on semantic grammars, but as the amount of data available for application development increased, as well as application complexity, stochastic approaches have been preferred.

MIT proposed the linguistic analizer TINA [85]. It is based on context-free rewrite rules with constraints which are converted at run-time in a network representing both syntactic and semantic categories. Rules are trained from data and can be used to constrain recognition based on sentence syntactic structure.

Many problems of interpretation in SLU systems derive from the fact that many sentences are ungrammatical and ASR hypotheses contain errors, so grammars have a limited coverage. These considerations suggest the use of more specific, but more robust, models. In the early nineties, the DARPA ATIS project started a series of task-dependent SLU systems. Data were collected in the domain of a flight information and reservation service. ATIS project aimed at providing a natural language interface to a travel information database and provided a benchmark for many spoken language understanding systems, one is discussed in [64].

Based on the model used to perform the interpretation process, other SLU systems can be classified as probabilistic SLU systems. An example is the Chronus system from AT&T [7]. This system implements the noisy-channel paradigm and represents knowledge as a Markov model in which observations are words. One state is used for each semantic concept.

Another type of SLU systems is based on classification models, which provides in practice a more robust spoken language understanding. This approach has been used in the context of dialog utterance and user intent classification. An example of this last context is the system "How May I Help You" from AT&T [35]. In this system user spoken utterances are classified into a number of predefined intents using a discriminative approach. It is basically a call classification application.

AT&T VoiceTone spoken language understanding system [36] mixes use of classification model and rule-based fixed grammars. A statistical classifier is used to determine user intents, while rule-based grammars are used for named-entities extraction. An active-learning framework is integrated in the system in order to improve the model, to adapt it to changes over time and to minimize human effort for data annotation and labeling. This last point is based on the selection of data to be annotated depending on a confidence measure given by the model.

All the stochastic models for Spoken Language Understanding proposed so far performs the translation from spoken sentence to a semantic constituents-based represention using statistical learning models that broadly fall in two main categories: generative and discriminative learning models. Such approaches are complementary for some aspects, the former are robust with respect to overfitting training data and the latter can easily integrate structured features like word sequences encoding long-distance dependencies. This work aims at prooving that the joint use of complementary models, or more in general models with different characteristics,

leads to more robust and accurate SLU systems.

Chapter 3

State of the Art

In order to complitely understand how Spoken Language Understanding is performed, all the different aspects of the task must be explained. Like all modules that are part of more complex systems, the SLU module can be explained describing its input, its output and how the latter is generated starting from the former. This chapter describes separately all these points. In particular, the extraction of a semantic interpretation from a user utterance transcription is performed in two different steps: Automatic Concept Labelling and Attribute-Value Extraction.

In the next section we describe what is the input for a SLU module, afterward we desctribe what is the generated output, in Section 3.2. The Automatic Concept Labelling phase of SLU is performed using stochastic models, in Section 3.3 we describe all the models used for SLU in the last decade, giving more details for models used in our research work. The second phase of SLU, Attribute-Value Extraction, can be performed either with rule-based or probabilistic approaches, these are described in Section 3.4. Finally, in Section 3.5, we address robustness issues to deal with when performing SLU.

3.1 SLU Input

As mentioned in previous chapter, three different types of input can be used when performing Spoken Language Understanding: Manual or automatic transcriptions of utterances and ASR word lattices. Word lattices are graph encoding all possible transcriptions an ASR module can generate for the same utterance.

Manual transcriptions are produced by human experts. Usually in the transcription process also spontaneous speech phenomena are taken into account and annotated using special tags. In some cases also punctuation is added. As an example, if an utterance contains the sentence "Good morning I have a problem with my printer", an hypothetical manual transcription can be

<**filler**> Good morning, <**filler**> I have a <**incomplete**>prob</**incomplete**>, a problem with my printer!

The example shows speech phenomena like hesitation, which leads in most cases the user to use fillers, truncation, the word "problem" has been truncated to "prob" that has been annotated as incomplete, and self-correction, the user repeated the truncated word to state clearly his problem.

All the information annotated in the transcription can be used to make the SLU task easier. For example fillers can be modelled in ASR systems to produce a more accurate automatic transcriptions, this in turns improves also SLU accuracy. Punctuation can be used for example to detect sentence boundaries for syntactic chunking. Although satisfactory accuracy is usually achieved on manual transcriptions, this output is used only for development and evaluation purposes. Effective SLU modules must be robust when using ASR transcriptions as input.

The spontaneous speech phenomena mentioned above, even if they are annotated, constitute a source of noise in the speech signal transcribed by an ASR. Most of these speech phenomena cannot be modeled properly in ASR systems. Further, current ASR systems are far to be perfect, they are likely to introduce transcription errors.

Using the same example above, a possible automatic transcription for the sentence "Good morning I have a problem with my printer" can be

Good morning I *had* a *podium* with my *printed*

Beyond the simple example, it is intuitive that ASR errors can affect tremendously SLU performance. In case words like "*problem*" or "*printer*", that are very important in this case to understand the user statement, are mistaken, the SLU cannot any more extract a correct interpretation.

ASR systems can usually generate more than one transcription for the same utterance. These are represented in a compact structure represented as a graph and called word lattice. An example of lattice for the same sentence above is showed in Figure 3.1. In this picture it can be noticed that, among the many possible paths, there's a path corresponding to a correct transcription. Extracting the correct transcription from a lattice, when even possible, is a hard task and it has not been solved yet. Using lattices as input to SLU, as mentioned in previous chapter, leads to a huge search space where all possible transcriptions appear associated with all possible concepts. Finding correct semantic interpretations in such huge space is difficult and remains an open issue.

3.2 Semantic Representations

SLU aims at extracting meaning from natural language sentences. Designing a general meaning representation which can capture the expressivity of

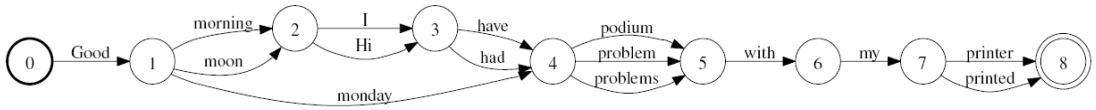


Figure 3.1: An example of ASR word lattice generated from an hypothetical utterance "Good morning I have a problem with my printer"

a spoken language is difficult. Therefore, in practice, meaning representations tend to depend on the capabilities desired for each application.

Semantic interpretation is based on the application of relations between signs and meaning. It can be seen as a translation process, performed with a semantic analyzer, in which signs are words of natural language and meaning is represented with a suitable Semantic Representation Language. In the past, rule-based SLU systems have been created using results from both syntactic and semantic analysis, e.g. the MIT TINA system introduced in previous chapter [85]. Syntactic and semantic analysis can be performed by a parser which produces a syntactic parse tree for a sentence with semantic labels attached to the nodes of the tree. Derivation of semantic interpretation from syntactic parse tree is discussed in details in [2]. As the availability of data for application development increased, as well as task complexity, probabilistic approaches have shown better robustness in SLU, an interesting comparison between rule-based, probabilistic and mixed approaches to SLU is shown in [8]. Additionally, given the spontaneous nature of utterances used as input in SLU tasks, and the consequent noise encoded within them, probabilistic SLU systems have been based only on semantic analysis, without making explicit use of syntactic structures. This was motivated by the fact that on such noisy input, syntactic parsers cannot reach a reasonable high accuracy, thus a manual annotation of syntactic trees would be required. This in turn would slow down significantly speech application development. Nevertheless semantic

representation can be still structured and each semantic constituent correspond implicitly to a syntactic constituent, like for example in [41], or can be designed to reconstruct complex context-dependent semantic structures starting from a flat tag annotation, like in [8].

A well-known representation of computer semantics is based on the use of frame semantics, dating back to 1968 and proposed with two different interpretations: the "case for case" lexical interpretation proposed by Fillmore [33] and then realized in the FrameNet project [3]; and the situation interpretation called "Scripts" proposed by Schank and Abelson [83]. Both representations try to model the way human brain captures abstract and concise semantic information from real world events or situations. These representations have been realized following linguistic theories, but also taking practical computer applications into account, such as Natural Language Processing tasks and Artificial Intelligence applications.

Although very interesting, the hystory and evolution of semantic representations for computers go beyond the purpose of this dissertation, which focus on models for SLU and doesn't attempt to find more effective representations. Thus in the following paragraphs we introduce the most diffused semantic representation based on frame semantics, we describe two different corpora where these representation is used, we introduce quickly the task of semantic analysis based on these corpora and we show the main difference with respect to SLU, motivating the different choice SLU semantic analysis relies on.

A frame is a structure identified by a name and a set of role-value pairs called slots. The use of frames allows to represent models in which can be included procedures to apply relations between signs and meaning. Procedures can be attached to frame slots. Each frame in the semantic knowledge of an application defines a prototype from which many instances can be obtained. An instance is obtained assigning value to roles. Finding

values for roles is a slot filling process performed by attached procedures.

A widely accepted semantic representation framework based on frames is FrameNet [3]. In FrameNet the annotation of semantic roles is made directly upon text sentences, instead of using the syntactic parses. Frames are triggered by particular words, called "*targets*", and the roles are related to the target by semantic relations and they are called semantic roles.

Another well-known representation for semantic analisys of text is the one adopted in the PropBank corpus [74]. The main difference with respect to the FrameNet corpus is that in PropBank target words are only verbs and semantic roles annotation is done upon manually checked syntactic parse trees. Subsequentely has been provided the mapping between roles on manual and automatic syntactic trees in order to be able to train systems also on automatic syntactic parsing.

The task of semantic analisys on text based on frame semantics is often referred to as Semantic Role Labeling (SRL). There is a huge literature available describing efficient appraoches to deal with this task, e.g. an overview is in [11] and [60], some particular SRL systems for different domains are presented in [46], [5] and [21].

The semantic analisys task modeled in FrameNet and PropBank corpora is based on syntactic analysis. This means that the first step in the processing is to generate a syntactic parse tree. The tree is then annotated with semantic information and this augmented tree is used to infer syntactic-semantic dependencies. These are used afterwards with the aim of performing a more effective semantic analysis. Syntactic analysis in turn can be based either on syntactic constituents, as the one used in [22] among many, or on syntactic dependencies, as described in [43] among others. This combined syntactic-semantic processing approach is motivated by the Chomsky hypothesis, where each syntactic constituent of a sentence maps into a conceptual constituent.

The kind of data used for Semantic Role Labelling is written text. This constitute a significant difference with respect to data processed when performing SLU. Data used in SLU tasks are generated starting from conversational speech. Transcription of this kind of data is much more noisy than written text, it contains many spontaneous speech phenomena, e.g. truncations, misspronunciations, hesitations, auto-corrections, restarting etc., that introduce noise and make the SLU task very hard. This also suggests that on the kind of transcriptions yielded from conversational speech, automatic syntactic parsing would contain many errors, making it even harder the understanding task based on the use syntactic trees.

For all these points, the general intuition of SLU community is that solution designed for Semantic Role Labeling would not be effective for SLU. Although semantic constituents used in SLU are still annotated upon syntactic chunks, in general no explicit information from syntactic analysis is used when performing SLU (see for example [30] for an annotation scheme based on syntactic chunking). The semantic representation used for SLU is very similar to the frame notation. The representation is based on attributes, like frame slots, which are semantic units instantiated by sequences of words, like for frames. Unlike frames, the attributes don't need to satisfy explicitly semantic relations to instantiate a specific frame, neither a target word is needed to instantiate a higher level structure like a frame. These attributes are annotated on manual transcription of utterances and are taken from a knowledge base designed for the specific task. Although, it is worth to note that recently, a work based on FrameNet semantic annotation on conversational speech has shown some relatively good results [23], subsequently refined in [22], hopefully opening an interesting research line towards the integration of syntactic and semantic features for Spoken Language Understanding.

For what it regards the corpora that will be described later in this dis-

sertation and that have been used for the experiments, for the ATIS corpus [26] the knowledge base is a relational database while for the MEDIA [8], the LUNA Italian [30] and the LUNA Polish [56] corpora a domain ontolgogy was designed (see [77] for LUNA Italian ontology). Following are some examples taken from each corpus to make it clear which kind of data is used in SLU tasks.

An annotation of the sentence *"I would like a flight from Phoenix to San Diego on April First"*, from the ATIS corpus, is:

> **null**{I would like a flight from} **departute_city**{Phoenix} **null**{to} **arrival_city**{San Diego} **null**{on} **departure_date.month**{April} **departure_date.day_number**{first}

where **departute_city**, **arrival_city**, **departure_date.month** and **departure_date.day_number** are domain concepts while **null** is the tag for words not meaningful for the task.

Given the sentence:

> *"Buongiorno io ho un problema con la stampante da questa mattina non riesco piu' a stamapare"*

taken from the LUNA corpus, that translates to *"Good morning I have a problem with the printer since this morning I cannot print any more"*, an example of the corresponding semantic annotation could be:

> **null**{Buongiorno io ho} **HardwareProblem.type**{un problema} **Peripheral.type**{con la stampante} **Time.relative**{da questa mattina} **HardwareOperation.negate**{non riesco} **null**{piu'} **HardwareOperation.operationType**{a stampare}

in this case the domain concepts are **HardwareProblem.type**, **Peripheral.type**, **Time.relative**, **HardwareOperation.negate** and **HardwareOperation.operationType**, while **null** has the same meaning as for

ATIS (and MEDIA as well).

Note that in the Italian corpus concepts are expressed as fields of a class, so that different concepts belonging to the same class can be eventually merged to construct more general and abstract semantic objects, similar to frames. As shown in [77], this representation can be exploited to perform semantic analysis based on ontology relations.

As an example taken from the MEDIA corpus, let us consider the sentence:

"Je veux une chambre double"

that translates to *"I want a double room"*, a possible semantic representation is:

null{Je veux} **nb_chambre**{une} **chambre_type**{chambre double}

where **nb_chambre** and **chambre_type** are the domain concepts.

This semantic interpretation contains attributes with the corresponding surface (chunks).

Another important point in the SLU process is the concept chunking, i.e. concepts can span over more than one word. In order to have an one-to-one association between words and concepts, useful in practice for model training, the beginning of a concept is distinguished by its continuation using markers equivalent to those of the *BIO*-notation (Begin, Inside, Outside) [79]: in particular the *Outside* marker (O) is replaced by the **null** tag introduced before.

Using this notation the semantic representation shown above would be:

null{Je veux} **nb_chambre-B**{une} **chambre_type-B**{chambre} **chambre_type-I**{double}

from this representation attribute names can be easily reconstructed.

3.3 Models for Spoken Language Understanding

In the last decade two major approaches have been studied to find correlation between words and concepts in the SLU tasks: (i) generative models, whose parameters refer to the joint probability of concepts and semantic constituents; and (ii) discriminative models, that learn a classification function based on conditional probabilities of concepts given words.

Assuming a labeling problem where observations X must be assigned a label y_i from a set Y, given a sample of data providing examples of labeling, generative models assume the sample was generated from a stochastic source following a joint probability distribution $P(X, Y)$ and they estimate the probability from the given data, either directly or via decomposition into a class-conditional probability $P(X|Y)$ and a prior label distribution $P(Y)$. In contrast, discriminative models directly estimate the posterior probability $P(Y|X)$ from data, without any attempt to model the underlying probability discribution. These two approaches lead to very different and complex formulations depending if we are dealing with simple classification problems, e.g. Naive Bayes classifiers or Logistic Regression ([62]), or sequence labelling problems, e.g. Hidden Markov Models or linear chains Conditional Random Fields ([78],[52]).

It is intuitive that theoretically these two models converges as the sample size goes to infinite, since in this ideal situation the sample distribution matches the real distribution. Nevertheless in real applications, where a limited amount of data is available for probability estimation, their behavior is not predictable and it can change depending on many points, e.g. size of available data, type of task, task complexity, also choosing one or the other can depend on practical aspects, e.g. training and classification time, resource request.

Here we describe the most important generative and discriminative mod-

els used for Spoken Language Understanding in the last decade, showing also advantages and disadvantages of each model. Some of the models described in the following sections (Dynamic Bayesian Networks, Maximum Entropy, Statistical Machine Translation, Conditional Random Fields and Stochastic Finite State Transducers) as well as the attribute value extraction approaches described in Section 3.4 have been studied, implemented and applied to SLU in the context of the European funded project LUNA, together with other partners from University of Avignon [1], RWTH Aachen University [2] and Polish-Japanese Institute of Information Technology of Warsaw [3].

An example of generative model is the Hidden Vector State model (HVS) [41]. This approach extends the discrete Markov model encoding the context of each state as a vector. State transitions are performed as stack shift operations followed by a push of a preterminal semantic category label like for a tree parser. This way the model can capture semantic hierarchical structures without the need of tree-annotated data (using a minimal manually annotated bootstrap data set).

Another generative model is the one based on Stochastic Finite State Transducers (SFST), which performs SLU as a translation process from words to concepts. This model has shown high accuracy despite its simplicity ([81]). Another interesting aspect is its very easy integrability in speech recognition systems, where the output can be a word lattice, which in turn is naturally encoded as a stochastic FST.

A more recent generative model for SLU is based on Dynamic Bayesian Networks (DBN). Dynamic Bayesian networks have been applied to many sequential data modeling tasks, for example automatic speech recognition [96], part-of-speech tagging [92], dialog-act tagging [42], DNA se-

[1] http://lia.univ-avignon.fr/
[2] http://www.rwth-aachen.de/go/id/bdz/
[3] http://www.pjwstk.edu.pl/en/

quence analysis. DBN have shown to provide a great flexibility for complex stochastic system representation with good performance compared to other stochastic approaches. Two interesting studies of DBN for Spoken Language Understanding are described in [54] and [55].

The DBN-based SLU system described in [54] and [55] performs a pure stochastic understanding process. Unlike many models described in this work, DBN can integrate both phases of SLU in a single model, i.e. automatic concept labelling and attribute-value extraction. In contrast many models perform the second phase in a separate processing step, using a different model. This means that a DBN based SLU system gives the possibility of a stochastic value normalization phase, as opposed to the other main approach for value extraction based on deterministic rules (both approaches are described in Section 3.4). A new formulation of the concept decoding is needed when using a DBN model, the concept sequence is combined with the value sequence as follows:

$$\hat{c}_1^N, \hat{v}_1^N = \underset{c_1^N, v_1^N}{\operatorname{argmax}}\, p(c_1^N, v_1^N | w_1^T) = \underset{c_1^N, v_1^N}{\operatorname{argmax}}\, p(w_1^T | c_1^N, v_1^N) p(v_1^N | c_1^N) p(c_1^N)$$

Hypothesization of concepts is then performed as follows:

$$\hat{c}_1^N = \underset{c_1^N}{\operatorname{argmax}} \sum_{v_1^N} p(w_1^T | c_1^N, v_1^N) p(v_1^N | c_1^N) p(c_1^N) \tag{3.1}$$

Another important difference with respect to other models is that decoding is performed at segment level, i.e. the models have an inner mechanism to deal with transitions from a concept sequence to another. In contrast other models perform a token-wise decoding.

Figure 3.2 shows two generative DBN models used in the SLU system. For the sake of simplicity, some additional vertices (*variables*) and edges (*conditional dependency*) of the actual DBNs are not represented. In the

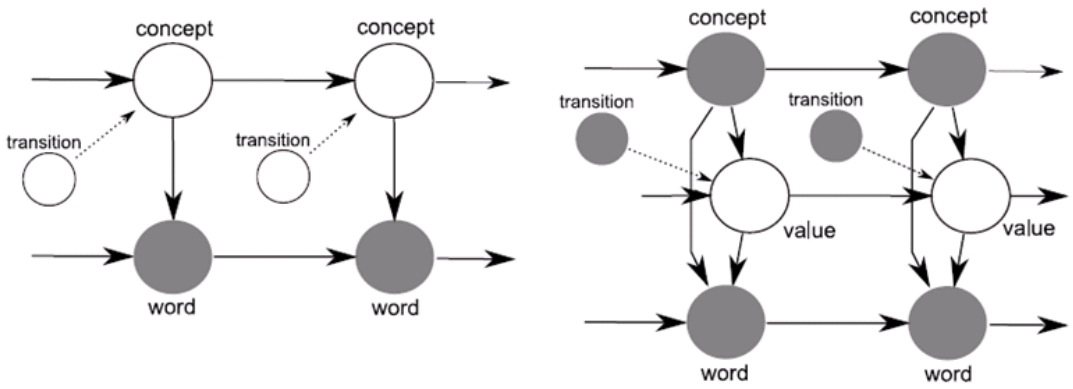


Figure 3.2: DBN-based 2+1-level SLU system. Concept model (graph on the left) is used for concept decoding. Value model (graph on the right) uses concept sequences as observations for value identification.

figure, only two time slices (corresponding to two words) are depicted. In practice, the regular pattern (*chunk*) is repeated so as to fit the entire word sequence under consideration. With reference to Figure 3.2, filled nodes are observed variables whereas blank ones are hidden. Plain lines represent conditional dependencies between variables; dashed lines indicate switching parents (variables influencing the relationship between others). An example of a switching parent is given by the *transition* node which influences the concept node: when *transition* is null, concept is a mere copy of the previous concept but when it is set to 1 the new concept value is determined accordingly to $p(c_n|c_{n-1})$.

In our context, all variables are observed during training, thus no iterative algorithm is needed to estimate the parameters. The edge's conditional probability tables can be directly derived from observation counts. However, in order to improve their estimates, factored language models (FLM) have been used along with generalized parallel backoff (GPB) [6]. FLM are an extension of standard LM where the prediction is based upon a set of features of the observations, instead of only the observations alone. To complement this new framework, GPB allows to extend the standard

backoff procedures to the case where heterogeneous feature types are considered and no obvious temporal order exists (contrary to classical LM, features in FLM can happen at any time, including the time of the prediction). Several FLM implementations are used in the SLU models, each one of them corresponding to an arrow in the DBN graph representations (see Figure 3.2) and estimates the following probabilities:

- $p(c_1^N) \simeq \prod p(c_i|c_{i-1}^{i-h})$: attribute name sequences,

- $p(v_1^N|c_1^N) \simeq \prod p(v_i|c_i)$: attribute values conditioned on attribute names,

- $p(w_1^T|c_1^N) \simeq \prod p(w_i|w_{i-1}^{i-h}, c_i)$: word sequences conditioned on attribute names (GPB works with order w_{i-1}^{i-h}, c_i),

- $p(w_1^T|v_1^N, c_1^N) \simeq \prod p(w_i|w_{i-1}^{i-h}, v_i, c_i)$: word sequences conditioned on attribute names and values (GPB works with order w_{i-1}^{i-h}, c_i, v_i).

where h represents an history which could vary according to the length of the model used ($\{-1\}$ for 2-grams, $\{-1, -2\}$ for 3-grams etc). GPB uses the modified Kneser-Ney discounting technique in all conditions.

In the DBN models described in [54] and [55], the concept and value decoding steps are decoupled and correspond to the two graphs depicted in Figure 3.2. The conceptual decoding process generates concept and transition sequences that become observed variables for the value decoding (described in Section 3.4). Conditional probabilities are either 2 or 3-gram FLM.

Another model used for SLU comes from the Machine Translation community, it is used for SLU in [39] and [38]. In this approach is used a standard phrase-based machine translation method which combines several

models. The incorporated models include phrase-based models in source-to-target and target-to-source direction, IBM-1 like scores at phrase level, again in source-to-target and target-to-source direction, a target language model, and additional word and phrase penalties. These models are log-linearly combined [61]:

$$p(c_1^N|w_1^N) = \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(c_1^N, w_1^N)\right)}{\sum_{\tilde{c}_1^N} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(\tilde{c}_1^N, w_1^N)\right)} \tag{3.2}$$

where $\tilde{c}_1^N$ represents all the concept sequences different of c_1^N. The $h_m(\cdot)$ represent feature functions and the λ_m the corresponding scaling factors. These factors are optimized using some numerical algorithm in order to maximize translation performance on a development corpus. In this case, optimization of the scaling factors is done with respect to the target score (Concept Error Rate in SLU), using the Downhill Simplex algorithm. In contrast to general translation models, NLU reordering of the target phrases composing the translation is not needed.

An example of discriminative model used for SLU is the one based on Support Vector Machines (SVMs) [94], as shown in [81]. In this approach, data is mapped into a vector space and SLU is performed as a sequence of classification problems using Maximal Margin Classifiers.

A relatively more recent discriminative approach for SLU is based on Conditional Random Fields (CRF) [52]. CRFs are undirected graph models. They train conditional probabilities taking into account many features of the input sequence. Since they are conditionally trained they don't need to represent explicitly feature dependencies, conditional dependence is captured using feature functions. CRF belongs to the family of log-linear models, the difference from other models of this family is in the factor used for probability normalization. Choosing a sequence level normalization leads to linear chain CRF, choosing a positional level normalization leads to the

Maximum Entropy model [4]. This model has been applied to SLU in [39] and [38] showing good performances.

In the following sections we describe more in details those models used in experiments performed during our research work: Stochastic Finite State Transducers model (SFST), Support Vector Machines (SVM), Conditional Random Fields (CRF) and finally Kernel Methods used in combination with SVM for our combined models.

3.3.1 A Generative Model: Stochastic Finite State Transducers (SFST)

The model based on Stochastic Finite State Transducers performs SLU as a translation process from words to concepts using SFSTs. The decoding process is realized in four different steps using four different transducers:

- λ_w is the representation of the input sentence as a FST;
- λ_G is a transducer mapping words to word categories in order to carry out some generalization (words not belonging to any category are mapped into themselves);
- λ_{W2C} is a transducer mapping words/word-categories into concepts;
- λ_{SCLM} is the transducer representation of a Stochastic Conceptual Language Model (SCLM).

In the third step, performed by λ_{W2C}, the mapping is not one-to-one, thus several different annotation hypotheses are generated. The fourth FST, λ_{SCLM}, ranks the hypotheses based on the joint probabilities of words and concept sequences, i.e.:

$$P(W_1^N, C_1^N) = \prod_{i=1}^{N} P(w_i, c_i | h_i), \tag{3.3}$$

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

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 decoding steps of the translation process are implemented using the AT&T FSM/GRM tools [63] and the SRILM [90] tools. In particular the SCLM is trained using SRILM tools and then converted to a SFST. This allows the use of a wide set of stochastic language models [12] (both back-off and interpolated models with several discounting techniques like Good-Turing, Witten-Bell, Kneser-Ney etc.).

SFST operations are used to combine all the transducers

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

and to find the best interpretation hypothesis performing a Viterbi search:

$$(W, C) = bestpath_1(\lambda_{SLU}),$$

where $bestpath_n$ searches and outputs the n-best hypotheses (in this case n is 1 for the 1-best hypothesis).

This model is very simple and very fast in both training and classification phases and it has a relatively good accuracy [81]. On the other hand, since it is a generative model based on joint probabilities of a Stochastic Language Model, it is affected by the presence of Out-of-Vocabulary words.

Further, this model is not suitable to integrate complex features since a reliable estimation of joint probabilities of words, concepts and features requires a huge amount of data. In this situation, the problem is made tractable usually assuming some independencies between stochastic variables and the joint probability is split into several conditional probabilities, leading to Bayesian Network models described earlier. Moreover, depen-

dencies captured explicitly by this type of models are limited to the order of n-grams used in the stochastic language model.

3.3.2 A Discriminative Model: Conditional Random Fields (CRF)

A relative more recent approach for SLU is based on Conditional Random Fields (CRF) [52]. CRF are undirected graph log-linear models training conditional probabilities of concept sequences $c_1^N = c_1, \ldots, c_N$ given word sequences $w_1^N = w_1, \ldots, w_N$

$$p(c_1^N | w_1^N) = \frac{1}{Z} \prod_{n=1}^{N} \exp\left(\sum_{m=1}^{M} \lambda_m \cdot h_m(c_{n-1}, c_n, w_{n-2}^{n+2}) \right) \tag{3.4}$$

where λ_m is the vector of parameters to be trained, $h_m(c_{n-1}, c_n, w_{n-2}^{n+2})$ are the feature functions used to capture dependencies between input features (words and other features that can be associated with words in a certain window around the current word to be labeled) and the output concept [58]. Different kind of feature functions are used, all binary valued:

1) Lexical Features: they capture dependencies between words and concept. Let w_i be the ith word in an input sentence and c_i the corresponding concept prediction, then lexical feature functions are defined as

$$h_{w,d,c}(c_i, w_{i-d}^{i+d}) = \delta(w_{i+k}, w) \cdot \delta(c_i, c) \quad k \in \{-d.. + d\} \tag{3.5}$$

where $\delta(.,.)$ is the Kronecker function. These feature functions fire when the word w_{i+k} is equal to the word w and the current concept prediction c_i is equal to c.

2) Prefix and Suffix Features: These feature functions are useful to overcome the problem of Out-Of-Vocabulary words, i.e. words that have not been seen in the training data. Let the word w be composed of $\alpha\beta$, so that α (β) is prefix (suffix) of w, then prefix (suffix) features are defined as:

$$h_{\bar{\alpha},c}(c_i, w_{i-d}^{i+d}) = \begin{cases} 1 & \text{if } \exists \alpha, \beta : w_i = \alpha\beta \wedge \alpha = \bar{\alpha} \wedge c_i = c \\ 0 & \text{otherwise.} \end{cases}$$

$$h_{\bar{\beta},c}(c_i, w_{i-d}^{i+d}) = \begin{cases} 1 & \text{if } \exists \alpha, \beta : w_i = \alpha\beta \wedge \beta = \bar{\beta} \wedge c_i = c \\ 0 & \text{otherwise.} \end{cases}$$

3) Capitalization Features: capitalization is a feature typical of proper names, that starts with a capital letter, and acronyms, that contains only capital letter, thus using a feature function capturing this characteristic of words is important. Capitalization feature functions are defined as:

$$h_{CAP,c}(c_i, w_i) = \begin{cases} 1 & \text{if } w_i \text{ starts with a capital letter } \wedge c_i = c \\ 0 & \text{otherwise.} \end{cases}$$

$$h_{ALL,c}(c_i, w_i) = \begin{cases} 1 & \text{if } w_i \text{ is all capitalized } \wedge c_i = c \\ 0 & \text{otherwise.} \end{cases}$$

in the same way we can define a feature function firing when a word contains a capital letter in general, but usually only the two functions above are used.

4) Transition Features: these feature functions capture dependencies between concepts at different positions, in current implementations of CRF only the previous concept is taken into account (bigram features). Transition feature functions are defined as:

$$h_{c',c}(c_{i-1}, c_i) = \delta(c_{i-1}, c') \cdot \delta(c_i, c) \tag{3.6}$$

5) Prior Features: Prior features are needed to compute a prior distribution of concepts, basically they provide unigrams counts:

$$h_c(c_i) = \delta(c_i, c) \tag{3.7}$$

6) Compound Features: This kind of features allow capturing dependencies between whole phrases and concepts. This way the same information provided by n-grams can be extracted, possibly with gaps, i.e. one or more words can be skipped:

$$h_{\{w_1,d_1\}\ldots\{w_K,d_K\},c}(c_i, w_{i-d_k}^{i+d_k}) = \prod_{k=1}^{K} \delta(w_{i-j}^{i+j}, w_k) \cdot \delta(c_i, c) \tag{3.8}$$

$$j \in \{-d_k \ldots + d_k\}$$

In order to have a unique notation for all types of feature functions, we can define the input for all functions as:

$$h_m(c_{i-1}, c_i, w_{i-d}^{i+d})$$

then computing the function depending on the type of features we are considering. Additionally we can associate an index $m \in \{1..6\}$ to each kind of feature function shown above, this way we can write formula 3.4 consistently.

In formula 3.4, Z is a normalization factor used to model a well defined probability distribution (summing up to 1):

$$Z = \sum_{\tilde{c}_1^N} \prod_{n=1}^{N} H(\tilde{c}_{n-1}, \tilde{c}_n, w_{n-2}^{n+2}) \tag{3.9}$$

where $\tilde{c}_{n-1}$ and $\tilde{c}_n$ are the concepts predicted for the previous and current words.

As mentioned in Section 3.3, using a positional normalization factor, that is:

$$Z = \prod_{n=1}^{N} \sum_{\tilde{c}} H(c_{n-1}, \tilde{c}, w_{n-2}^{n+2}) \tag{3.10}$$

leads to another log-linear model, called Maximum Entropy model [4].

There is a certain relation between log-linear models and the SMT model presented in Section 3.3. The feature functions in SMT approach are statistical models which return float values and thus the features are no more binary. Merely seven parameters for the combination of the models are tuned in contrast to the millions of parameters which are used within CRF. Also, the optimization problem is not convex as in the case of CRFs.

Conditional Random Fields provide someway advantages of both generative and discriminative models. They can easily take many features of the input into account since they are conditionally trained, like discriminative models, and they use previous decisions to trade-off the best label at the current position.

In current implementations [47, 38], CRFs can use features in a limited window around the current word to be labeled (like SVMs). Since conditional dependence is modelled by feature functions using such window, this creates some limits on the dependency distances between features and labels learned from data, as highlighted also in [50]. Finally, the use of many features and the complexity of the modelled task, in terms of number of labels, produces training time problems like for SVMs.

3.3.3 Support Vector Machines (SVM)

Support Vector Machines (SVMs) are well-known machine learning algorithms belonging to the class of linear classifiers [94]. Their main idea is to learn a hyperplane

$$H(\vec{x}) = \vec{w}\vec{x} + b = 0 \tag{3.11}$$

which divides training examples with maximum margin, where $\vec{x}$, the feature vector representation of a classifying object o, $\vec{w} \in \mathbb{R}$, and $b \in \mathbb{R}$ are the learned parameters [94].

The problem of learning the hyperplane can be solved by applying the lagrangian optimization theory, which leads to the following dual form of Eq. 3.11:

$$\sum_{i=1..l} y_i \alpha_i \vec{x_i}\vec{x} + b = 0, \tag{3.12}$$

where $\vec{x_i}$ are the training examples, y_i (+1 or -1) is the label associated with $\vec{x_i}$ and α_i are the lagrange multipliers [4].

Support Vector Machines solve the concept labeling problem as a sequence of classification problems using binary classifiers. Each classifier can be trained taking into account many non-local features but, at classification time, the current concept in the sequence is decided locally without using decisions made at previous steps. Each classifier can be trained with an *One-VS-All* or a *Pair-Wise* approach and the final decision is made with a voting (weighted or not) scheme.

An interesting aspect of SVMs is that they can take into account many non-local features, but in a limited window around the current word (target of the labeling). As drawbacks, SVMs make local decisions for labeling words and even more they are trained with quadratic programming algorithms, this results in a relatively long training time. Classification time depends on the size of the model, but it can be easily speed-up decomposing the computation in several parts and combining afterwards partial results.

[4] In the last decades, many alternative algorithms have been proposed to solve Eq. 3.12 more and more efficiently.

3.3.4 Kernel Methods

Support Vector Machines have proved to be very accurate algorithms and, as a further quality, they can implicitly represent data in high dimensional vector spaces (see e.g. [87]). Indeed, Eq. 3.12 shows that SVM algorithm only depends on the inner product between instances. This allows to use functions mapping objects onto higher dimensional vector space as follows

$$\sum_{i=1..l} y_i \alpha_i \phi(o_i)\phi(o) + b = 0 \tag{3.13}$$

where o_i and o are two objects described by the feature vectors $\vec{x_i}$ and $\vec{x}$, respectively, and ϕ is a mapping function.

The final formulation of the dual problem is:

$$\sum_{i=1..l} y_i \alpha_i K(o_i, o) + b = 0 \tag{3.14}$$

Note that in this form we do not need to know the vector space induced by the function K provided that K is a valid kernel function [5].

Since in many real world cases data cannot be classified using a simple linear classifier, kernel methods can be used to carry out learning in more complex spaces. In this work we use kernel functions particularly suitable for Natural Language Processing (NLP) tasks, i.e.: string kernels [87] and Tree Kernels [14, 65].

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. Gaps modify the weight associated with the target substrings as shown in the following.

[5] Satisfying the Mercer's theorem conditions [87]

Let Σ be a finite alphabet, $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$ is the set of all strings. Given a string $s \in \Sigma^*$, $|s|$ denotes the length of the strings and s_i its compounding symbols, i.e $s = s_1..s_{|s|}$, whereas $s[i:j]$ selects the substring $s_i s_{i+1}..s_{j-1}s_j$ from the i-th to the j-th character. u is a subsequence of s if there is a sequence of indexes $\vec{I} = (i_1, ..., i_{|u|})$, with $1 \leq i_1 < ... < i_{|u|} \leq |s|$, such that $u = s_{i_1}..s_{i_{|u|}}$ or $u = s[\vec{I}]$ for short. $d(\vec{I})$ is the distance between the first and last character of the subsequence u in s, i.e. $d(\vec{I}) = i_{|u|} - i_1 + 1$. Finally, given s_1, $s_2 \in \Sigma^*$, $s_1 s_2$ indicates their concatenation.

The set of all substrings of a text corpus forms a feature space denoted by $\mathcal{F} = \{\sqcap_\infty, \sqcap_\in, ..\} \subset \pm^*$. To map a string s in $\mathbb{R}^\infty$ space, we can use the following functions: $\phi_u(s) = \sum_{\vec{I}:u=s[\vec{I}]} \lambda^{d(\vec{I})}$ for some $\lambda \leq 1$. These functions count the number of occurrences of u in the string s and assign them a weight $\lambda^{d(\vec{I})}$ proportional to their lengths. Hence, the inner product of the feature vectors for two strings s_1 and s_2 returns the sum of all common subsequences weighted according to their frequency of occurrences and lengths, i.e.

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

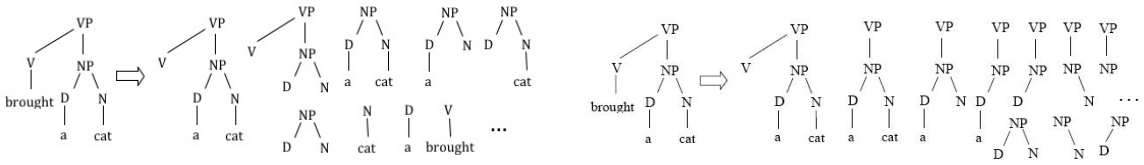
$$\sum_{\vec{I}_2 : u = s_2[\vec{I}_2]} \lambda^{d(\vec{I}_2)} = \sum_{u \in \Sigma^*} \sum_{\vec{I}_1 : u = s_1[\vec{I}_1]} \sum_{\vec{I}_2 : u = s_2[\vec{I}_2]} \lambda^{d(\vec{I}_1) + d(\vec{I}_2)},$$

where $d(.)$ counts the number of characters in the substrings as well as the gaps that were skipped in the original string. It is worth noting that:

(a) longer subsequences receive lower weights;

(b) some characters can be omitted, i.e. gaps; and

(c) gaps determine a weight since the exponent of λ is the number of characters and gaps between the first and last character.



(a) Syntactic Tree fragments (STF)

(b) Partial Tree fragments (PTF)

Figure 3.3: Examples of different classes of tree fragments used as features by Tree Kernels.

Characters in the sequences can be substituted with any set of symbols. In our study we preferred to use words so that we can obtain word sequences. 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.

Tree kernels

Tree kernels represent trees in terms of their sub-structures (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.

Tree Fragment Types

An STF is a general subtree whose leaves can be non-terminal symbols. For example, Figure 3.3(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 3.3(b) shows that the number of PTFs derived from the same tree as before is still higher (i.e. 30 PTs).

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. In the following sections we report the equation for the efficient evaluation of Δ for ST and PT kernels.

Syntactic Tree Kernels (STK)

The Δ function depends on the type of fragments that we consider as *basic* features. For example, to evaluate the fragments of type STF, it can be defined as:

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

$$\Delta(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + \Delta(c_{n_1}^j, c_{n_2}^j)) \tag{3.15}$$

where $nc(n_1)$ is the number of 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, $nc(n_1) = nc(n_2)$. $\Delta(n_1, n_2)$ evaluates the number of STFs common to n_1 and n_2 as proved in [14].

Moreover, a decay factor λ can be added by modifying steps (2) and (3) as follows[6]:

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. 3.15 is $O(|N_{T_1}| \times |N_{T_2}|)$ but as shown in [65], the average running time tends to be linear, i.e. $O(|N_{T_1}| + |N_{T_2}|)$, for natural language syntactic trees.

The Partial Tree Kernel (PTK)

PTFs have been defined in [65]. 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.

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

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 $\Delta(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1,\vec{I}_2,l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big), \tag{3.16}$$

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. 3.16 is more general than Eq. 3.15. Indeed, if we only consider the contribution of the longest child sequence from node pairs that have the same children, we implement the STK kernel.

3.3.5 SLU Models Combination

The individual models described in previous sections show different characteristics and performances. As consequence, new SLU models can be found combining individual models. There are two main approaches that have been studied for models combination for SLU: the "Recognizer Output Voting for Error Reduction" (ROVER) [34] and the re-ranking approach.

ROVER performs an alignement of different system outputs training token level weights and applying a voting scheme in order to choose the best SLU interpretation. ROVER was first studied for speech recognition [34] and it has been applied to SLU in [38] where, since a single best output was considered for each combined system, ROVER is just a majority voting on concept level after a Levenshtein alignment. Additionally, in the approach described in [38], the system weights are optimized using Powell's method (`multistart`) [76].

Given three hypothetical SLU system outputs

1. a b c d

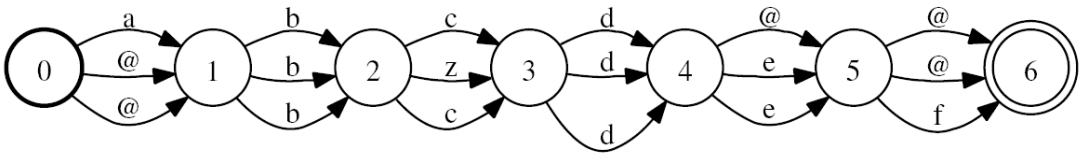


Figure 3.4: Example of SLU system outputs alignment carried out by the ROVER algorithm. @ is used as empty string symbol.

2. b z d e

3. b c d e f

the ROVER approach performs first the output alignement, as shown in Figure 3.4. With reference to this figure, the symbol @ is used for empty tokens.

Once the alignment has been performed, the best token at each position is chosen. The set of tokens at each position is called correspondence set (CS), for example in Figure 3.4, the correspondence set at first position is $\{a, @\}$. The number of occurrences of the token t in the correspondence set at position i is $N(t, i)$, while the total number of tokens in CS at position i is $N(i)$. Using these counts, ROVER can be performed using relative frequency as scoring function at each position. For example, relative frequency of tokens in the CS at position 1, in Figure 3.4 is 0.33 for a and 0.66 for @ ($N(a, 1) = 1$, $N(@, 1) = 2$, $N(1) = 3$). Usually also the confidence score given by the models can be used as scoring function. In this case the confidence of the empty string symbol, $Conf(@)$, is a parameter and must be trained from a validation set. The confidence of a token t in the CS at position i is $C(t, i)$.

In [34] three different scoring schemes are used to choose the best token at each position. The first is based only on relative frequency, the second is based only on confidence scores given by SLU models, the third

scheme combines these two approaches choosing the best token in the CS at position i with the following scoring function:

$$Score(t,i) = \alpha(N(t,i)/N(i)) + (1-\alpha)C(t,i) \tag{3.17}$$

where $0 < \alpha < 1$ is another parameter to be trained from a validation set. As shown in [34], this third scheme gives the best results.

The Re-ranking approach for models combination has been used in several tasks: Syntactic Parsing [15, 17, 88, 49, 50, 65], Named Entity Recognition [15, 14], Machine Translation [89], Question Answering [70], Semantic Role Labelling [68, 43], and more recently in Spoken Language Understanding [29, 28, 27].

Reranking is performed in two phases using two different models. A first model generates a list of candidate hypotheses for each input sentence. The second model select possibly the most correct candidate. The second model is trained on the hypotheses generated by the first model in order to minimize a ranking loss function [14]. This way the second model learns to classify as positive examples hypotheses containing less annotation errors, with respect to a gold standard annotation (the manual annotation of classified data).

In order to capture different information, some meaningful features must extracted from the hypotheses. For these purpose, two different feature representations have been proposed in literature: explicit representation using feature vectors [15, 14, 50] or implicit representation using kernel functions [17, 14, 70, 68, 28]. The advantage in using explicit representation is only in computationl cost, the drawback is the need to manually engineer effective features, which requires expert knowledge of the task. In contrast, using kernel functions is computationally expensive, but doesn't require features engineering. Objects are compared in arbitrary complex feature spaces, e.g. tree-fragment space shown in Section 3.3.4, implicitly

represented by the kernel function, basically without loss of performance [16, 14]. Additionally, using kernel functions can result in a redundant feature space [50], but this problem is included in the computational cost problem.

Regarding the model used to generate candidate hypotheses, there is basically no limitation: Probabilistic Context Free Grammar (PCFG) [17], Maximum Entropy model [16], Conditional Random Fields [50], SVM [70], Stochastic Finite State Transducers (SFST) encoding a Semantic Language Model [28].

Also for the second model, used to select the best candidate, different learning algorithms have been tried: Perceptron [16, 17, 14], Boosting [16, 50] and SVM [70, 28].

Since re-ranking is the most important basic block of our combined models, it will be described in details in the following chapters, giving also a comparison with ROVER in terms of characteristics and advantages.

3.4 Attribute-Value Extraction: Rule-based and Stochastic Approaches

Traditionally, after a concept sequence has been decoded from an input sentence, the segmented word substrings are converted to a normalized form, as defined in the task semantic dictionary. More generally, the dictionary is based on the database entities values. In the sentence

"*I'd like a room charged not more than fifty euros*",

taken from the MEDIA corpus, the normalization module translates the sequence

"*no more than*"

instantiating the attribute name `comparative-payment-room` to the normalized attribute value form

Table 3.1: Semantic concept (att./value) representation for the query *"Please give me the fares since I'd like a room charged not more than fifty euros"*.

words	mode	attribute name	normalized value
donnez-moi	+	`null`	
le	?	`refLink-coRef`	`singular`
tarif	?	`object`	`payment-amount-room`
puisque	+	`connectProp`	`imply`
je voudrais	+	`null`	
une chambre	+	`number-room`	`1`
qui coûte	+	`object`	`payment-amount-room`
pas plus de	+	`comparative-payment`	`less than`
cinquante	+	`payment-amount-integer-room`	`50`
euros	+	`payment-unit`	`euro`

"*less-than*" (cf. Table 3.1).

Many lexical sequences can correspond to the same normalized value. The attribute values are numeric units (e.g. phone numbers, codes, dates), proper names (e.g. customer's name) or semantic classes merging lexical units which are synonyms for the task (e.g. singular and plural names as well as different tenses of the same verb). The set of normalized values associated to each attribute is defined in the semantic dictionary, for example for the MEDIA task, there are three different possible configurations:

- a value enumeration (e.g. the concept "comparative" with possible values "around", "less-than", "maximum, "minimum" and "more-than"),
- regular expressions (as for dates) or
- open values (i.e. no restrictions, as for client's names).

This normalization step is commonly based on deterministic rules, but can also be introduced in the global stochastic model through an additional

level. Stochastic approaches for attribute value extraction have been integrated within the DBN and CRF models proposed in [55] and [38], respectively, and described in previous sections.

Approaches based on deterministic rules

In the case of the Stochastic Finite State Transducer approach, the normalization step based on manual rules can be applied either with a finite state transducer approach (i.e. also rules are encoded as SFST) or a script language based approach. Both use simple concept attribute dependent expressions to convert phrases supporting an attribute name into a normalized attribute value.

For SFST, the transducer encoding the rules is induced from the training data. The main interest of modeling rules as finite state transducers is their ability to process word graphs, like ASR lattices. Thus, the complete search space can be kept until the end of the process (see [80] and [86]). For the experiments reported in this dissertation, only single-best input has been used and thus only the script-based approaches have been utilized with the SFST model.

For all the tasks (corpora) used in our experiments, script based approaches exist. These comprise manually designed regular expressions, lists of named entities, functions to map written numbers to digits designed in order to process any single-best output coming from any system. For the four tasks considered in this work, rule-based approaches are more accurate than statistical approaches.

Stochastic approaches

DBN In the context of stochastic value identification, the concept sequence is combined with the value sequence in the concept decoding, accordingly to Equation 3.1. As a consequence, the word sequence probabil-

ities become conditioned both on the concepts and their normalized values. The complexity of the conceptual model becomes untractable in this conditions, so the decoding setup must be revised. However traditional sub-optimal decoding setups (such as beam search) lead to poor performance [54]. To avoid this problem, the normalization level is not really embedded in the conceptual model, but it is performed after the concept decoding. So, in the so-called 2+1-level approach, v is first marginalized (see Equation 3.1) then v is decoded given a constant c ($\hat{c}$, the hypothesis from the former level):

$$\hat{v}_1^N = \underset{v_1^N}{\operatorname{argmax}}\, p(w_1^T|\hat{c}_1^N, v_1^N)p(v_1^N|\hat{c}_1^N)p(\hat{c}_1^N) \tag{3.18}$$

Under the assumption that the normalized values have a slight or no influence on the segmentation process, Equation 3.1 allows for a better generalization of the conceptual model.

This stochastic approach has been applied to the DBN model in combination with script-based rules as described in [54].

CRF Knowing the location and the attribute name of content words given by the attribute name extraction, the next step is to extract normalized values for most of the attribute names, e.g. concerning the following examples from the Polish corpus and for the values "*Request*" or "*151*":

@Action[Request]{chciałam} @BUS[151]{linie sto piećdziesiat jeden} ...
@Action[Request]{I would like} @BUS[151]{line one hundred fifty one}
...

A 1-to-1 mapping like in attribute name extraction is not used, instead exactly one value is hypothesized per attribute name. As features, lexical features on the predecessor, the current, and the successor word can be used. For attribute names with a huge number of values and in particular

attribute names with an infinite set of values (e.g. numbers, dates, phone numbers), the attribute value extraction is left to a rule based approach in a possible post-processing step.

The number of possible values varies highly between attribute names. For example, the attribute name *Reaction* can take either the value "Confirmation" or "Negation" and is triggered by only few content words. In contrast, the value of *STREET_NUMB* can be any possible number. In principle, attribute value extraction can be realized using machine learning. This is a quite easy task when the number of possible values is low, but can become difficult for attribute names with a huge number of possible values like street or bus numbers. These numbers can not be covered completely by the training corpus, which is the only information source at least for purely data driven approaches.

This CRF model for attribute value extraction has only been applied to the CRF approach described in [38] and always in combination with rules.

In the SLU task the surfaces realizing a concept are normalized and only their associated values are returned. Considering again the example of Section 3.2 taken from MEDIA, a possible attribute-value interpretation would be:

nb_chambre[1] **chambre_type**[double]

which is the so-called flat attribute-value annotation output by an SLU module as final results. Note that at this level the **null** tags are removed since they are used to annotate words not relevant for the task and so they bring no semantic information. Even more values are a normalization of the corresponding surfaces in the sense that only keywords are kept for each concept and in some cases words are converted into digits (e.g. like for numbers, codes, dates etc.). The example above, shows the typical output of SLU module. In a Spoken Dialog System context, this output is passed

to the next processing module, the Dialog Manager, which computes the next move to be done in the dialog in order to fit user needs in the best way possible.

3.5 Models Robustness: Confidence Scores and Other Confidence Metrics

Current state-of-the-art speech recognition and understanding systems are far from perfect. In speech recognition there are a number of factors, like environment, telephone line quality, speaker variability, which can affect recognition performance. Moreover, the understanding component can, in some cases, generate an incorrect interpretation, leading the dialog in completely wrong path. Since is not feasible to make systems perfect, the only solution to these problems is to detect them and to apply a recovery strategy. Error detection in speech recognition and understanding systems is based on the evaluation of a confidence measure.

Estimating the confidence of an interpretation involves several issues:

- Choosing the span of the confidence measure: confidence measurement can be applied at the word level, concept level, utterance level or their combination.
- Defining the set of features used in order to estimate the confidence measure: modern approaches use ASR features, like acoustic scores and linguistic scores, SLU features and dialog context.
- Defining an efficient way to combine the different kind of features.
- Choosing a decision strategy in order to decide when interpretation is correct or incorrect.

An approach to recognition confidence scoring and a method for integrating confidence scores into the understanding and dialog component is presented in [40]. In this work confidence scoring is performed at different levels. At phonetic level is used a normalized acoustic score produced by the acoustic model of the recognizer. At utterance level fifteen different features, both acoustic and linguistic, are used to estimate a confidence measure. At word level ten different features, again acoustic and linguistic, are combined to produce a unique measure. At utterance and word level, the features are put together in a features vector. The final confidence measure is obtained with a scalar product of the feature vector by a projection vector which components are trained from data using a minimum classification error training technique.

Two methods to incorporate semantic information into word and concept level confidence measures are proposed in [82]. The two methods use two sets of statistical features to model semantic information in sentences. The first method relies on semantic parse tree and uses nodes and extension scores corresponding to tags and labels. This first method is motivated from the fact that correct sentences are easily parsed and have a higher score in the parse tree. So, even if spontaneous speech has ungrammatical constructions, since the parser is trained with a data-driven approach, the parser model learns ungrammatical structures contained in the data. The second method is based on joining a maximum entropy model of word sequences and semantic parse trees.

In order to improve user utterance interpretation and classification performance, a very important aspect for confidence measure estimation is to integrate information related to dialog context. The speech understanding problem in the context of a spoken dialog system is presented in [1]. In this work the dialog flow is modeled in a maximum likelihood framework taking into account both understanding system and dialog manager state

transitions on one side and user-directed state transitions on the other side. Language model adaptation based on the dialog context is performed through the clustering of dialog prompts, which are used as features in conjunction with acoustic and linguistic features.

The work descirbed here focus on models for Spoken Language Understanding, so we didn't study new approaches for confidence measure estimation, we used existing approaches for what it regards ASR and SLU confidence scores, but we designed a new confidence-based interpretation re-selection approach for our joint models that will be described deeply later on.

Chapter 4

Models Combination Via Discriminative Re-ranking

Most of the work described in this document focus on models combination using discriminative re-ranking [14]. The re-ranking approach we propose is based on a model able to classify pairs. This means that training and classification instances are pairs of objects instead of single objects as in traditional classification tasks.

Once a model M able to generate a list L of semantic annotation hypotheses on an input sentence is available, the following steps take place to perform re-ranking:

- M generates the m most likely semantic annotations for each input sentence I, i.e. the list $L = \{s^1, ..., s^m\}$. m is a parameter chosen at the beginning of the process

- L is used to build pairs of hypotheses $\langle s^i, s^j \rangle$, with $i \neq j$, that are the training and classification instances for the re-ranking model. Pairs are built in different way for the training and classification phases.

- The Re-ranker is trained and the resulting model R is applied on classification instances

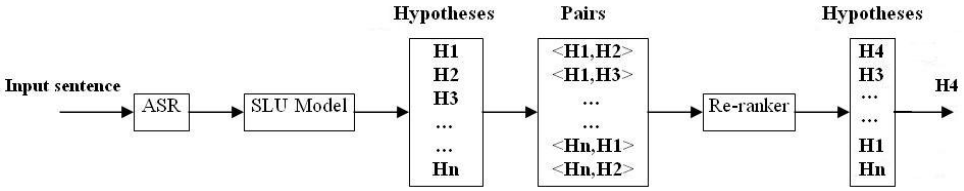


Figure 4.1: A general diagram of re-ranking framework showing the entire chain of processing, from speech input to the SLU interpretation

- The scores S_R computed on the classification instances using R, are used to re-rank, i.e. re-arrange, the list of hypotheses generated by the first model based on a different metric. The top ranked hypothesis of the re-ranked list is the new best interpretation

If we define $L = M(I)$ as the list of hypotheses generated by M on the input sentence I, $S_R(L)$ as the list of scores given by R for each hypothesis in L, then the best hypothesis given by the reranking model R is the one associated to the best score $argmax_L(S_R(M(I)))$.

All these steps hold in general, regardless which is the model used to generate hypotheses lists and the model used to re-rank such lists. For the sake of clarity, this general re-ranking framework is depicted in Figure 4.1.

Re-ranking hypotheses generated with a model allows to exploit the knowledge coming from the training data in two different ways and to put together advantages coming from the two combined models: the first is trained directly from the training; the second takes the hypotheses as input, thus exploiting the knowledge encoded by the first model, and extracts different features, adding further knowledge.

The pairs used in the re-ranker model are built from the list of hypotheses generated by the first model. Pairs must be constructed in such a way that allows the re-ranker to learn which hypotheses are better than the others, i.e. which hypotheses contain less mistakes with respect to a reference semantic annotation and a given metric.

In the following sections we describe in more details the steps involved when performing re-ranking: in Section 4.1 we describe how we generate interpretation hypotheses to be re-ranked, in Section 4.2 we describe how pairs of hypotheses should be built in order to learn how to classify the most correct hypotheses. In Section 4.3 are described the structures used to represent hypotheses in SVM, using kernel functions. Finally, in Section 4.4, we describe how training and classification phases are performed using a particular kernel function designed for re-ranking.

4.1 Hypotheses Generation

The re-ranking framework proposed in our solution is based on SVMs and Kernel Methods, described in the previous chapter. As we mentioned earlier, all the stochastic models used for SLU, and in general all the stochastic models, can be characterized as generative and discriminative models. These two classes of models have different and complementary characteristics. As consequence, it seems intuitive that combining a generative and a discriminative model is more effective than combining models with same characteristics.

All the work we have done in SLU on models combination using discriminative re-ranking has been done following this intuition ([29, 28, 27]). Nevertheless, recently we have applied to our re-ranking models a new hypotheses selection criterion, described later, that allows to re-rank hypotheses generated by any kind of model, generative or discriminative.

Thus, the model used to generate the hypotheses lists can be either the SFST model described in Section 3.3.1 or the CRF model, described in Section 3.3.2. From now on, we will refer to the model used to generate the hypotheses as first model (of the re-ranking framework) or the re-ranked model.

The first model takes as input the transcription of a spoken sentence and produces the m most likely conceptual annotations of the sentence, ranked on the joint probability of the Stochastic Conceptual Language Model (SCLM) when using the SFST model, while they are ranked on the posterior probability of concepts given words when using CRF. In particular the algorithm used to find the m most likely annotations is a Viterbi search with the SFST model, while with CRF the search is perfomed in two steps: Viterbi search in a forward step and A^* search in a backward step.

A pool of m hypotheses is generated from each of the sentences in the training and classification data. For example, from the sentence

"Ho un problema col monitor"

which translates to *"I have a problem with my screen"*, the following hypotheses can be generated

1. ***NULL***{*ho*} ***PROBLEM-B***{*un*} ***PROBLEM-I***{*problema*} ***HARDWARE-B***{*col*} ***HARDWARE-I***{*monitor*}
2. ***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. Beyond this simple example, in many cases the top ranked hypotheses is not the most correct, applying re-ranking we aim at finding the most correct hypothesis for each input sentence.

4.2 Pairs Generation

The m-best list of hypotheses is used to build annotation pairs $\langle s^i, s^j \rangle$, where $i, j \in [1..m]$ and $i \neq j$. Pairs are built in two different ways for training and classification phases:

I) Support Vector Machines are binary classifiers, i.e. they are able to classify instances in two classes that are tradictionally called positive and negative class. In order to train a re-ranking model it is needed to define positive and the negative instances. In training phase, the pairs are positive instances if s^i has a lower concept annotation error rate than s^j, with respect to the manual semantic annotation in the training data. The metric to compare hypotheses with the reference is the edit distance, that is used as evaluation metric for Automatic Speech Recognition systems (ASR). It is computed as the ratio between inserted, deleted and substituted tokens over the number of tokens in the reference.

The edit distance is computed for all the hypotheses and the most correct is selected (i.e. the hypothesis with the lowest error rate). Let s^k be the most correct hypotheses in the m best list, positive instances for training the re-ranker are pairs $\langle s^k, s^i \rangle$ for $i \in [1..m]$ and $i \neq k$. Since the model is simmetric, negative instances are built simply inverting the elements in the positive ones, i.e. $\langle s^i, s^k \rangle$.

Organizing hypotheses in this way, given a generic pair $\langle s^i, s^j \rangle$, a trained binary classifier can decide if s^i is more accurate than s^j.

II) In classification phase, we cannot rely on assessment of hypotheses based on the edit distance with respect to the reference annotation, so from the m-best list of hypotheses, all possible pairs are generated. This means that $o(m^2)$ pairs are generated from the hypotheses list. Nevertheless, applying the same simplification applied in [14], only $o(m)$ instances are generated and each instance is a single hypothesis (or tree), providing a

significant speed up in classification phase.

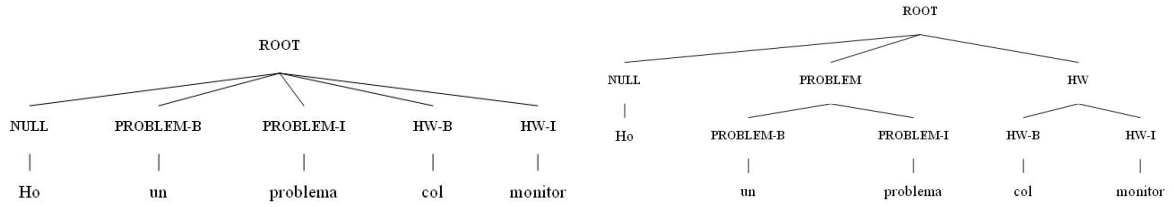
The re-ranking model captures the meaningful features of each pair, the pairs with the highest number of these features receive a higher score, so that to bring possibly in the top the pair containing the most correct hypothesis.

4.3 Structures for Kernels

Given the pairs described in previous section, we need to represent them in SVMs. One alternative to the usual approach of n-gram extraction is the use of kernel functions applied to structures built from pairs. Kernel methods are viable approaches to engineer features for text processing, e.g. [14, 49, 25, 10, 24, 93, 50, 65, 70, 66, 68, 69].

The kernels described in Section 3.3.4 provide a powerful technology to capture structural features from data. In particular, Tree Kernels were originally designed for data represented as syntactic parse trees. In Spoken Language Understanding the data are made of transcriptions of spoken sentences with their semantic annotation (i.e. basic semantic constituents). This type of annotation is rather flat with respect to syntactic parse trees, therefore, to exploit the power of kernels, a careful structure design for data representation must be carried out. The goal is to build the most suitable representation for each specific kernel: sequences for the String Kernel (SK) and tree-like structures for (Partial) Tree Kernel (STK and PTK), starting from semantic annotation. Note that the latter is made upon sentence chunks, which implicitly define syntactic structures as long as the annotation is consistent in the corpus.

Taking into account the characteristics of the target kernel functions and the structure of semantic annotation, we designed several different structures that are variants of two main types of structures representing



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

Figure 4.2: Examples of "*FLAT*" and "*MULTILEVEL*" semantic trees used for STK and PTK

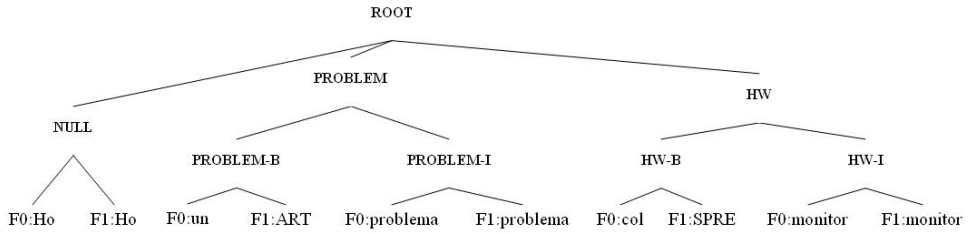


Figure 4.3: An example of "*FEATURES*" semantic tree used for STK or PTK

sequences or trees (see [28]). They refer to the same annotation example reported in Section 4.1:

- two types of sequential structures exploitable by SK, e.g.:

  *SK*1 ***NULL*** *ho* ***PROBLEM-B*** *un* ***PROBLEM-I*** *problema* ***HARDWARE-B*** *col* ***HARDWARE-I*** *monitor*

  *SK*2 ***NULL*** *ho* ***PROBLEM B*** *un* ***PROBLEM I*** *problema* ***HARDWAREB*** *col* ***HARDWARE I*** *monitor*,

  where the B/I tags characterize the *Begin* and continuation, *Inside*, of the multiword concepts.

- the tree structures shown in Figures 4.2(a), 4.2(b) and 4.3, which can be exploited by STK and PTK.

Structure variants are defined by:

- The position and the use of BIO-like tags (they can be used or not)

- Using or not a tag also for marking the end of a concept
- Using a fake level of nodes in the trees needed when using Syntactic Tree Kernel (which doesn't split node children, so the whole sequence of nodes must match at each level of the tree)
- Adding some redundancy in the tree to fit the way kernels compute similarity in practical implementations
- Using features together with the words instantiating concepts (features used are usually word categories and morpho-syntactic features)

The main characteristics of the structures are:

For SK1 and SK2 the order of words and concepts is meaningful since each word is preceded by its corresponding concept, so a generic sequence $\texttt{concept}_i$ $\texttt{word}_j$ capture a dependence between i and j while the sequence $\texttt{word}_j$ $\texttt{concept}_i$ does not. The difference between *SK1* and *SK2* is in the use of BIO-like 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.

The structures shown in Figure 4.2(a), 4.2(b) and 4.3 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 "flat" 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 *SK2*. 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 3.3.4, this structure 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 SLU. 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, except for particular models described later, we only used Word Categories as features. They can be domain independent categories, 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, i.e. *'F0'* for words and *'F1'* for word categories.

4.4 Training and Classification

Once pairs of hypotheses have been generated and they have been represented as semantic structures exploitable in kernel functions, the re-ranking model can be trained and used for classification.

Previous work, e.g. [14], has shown that, given the pair of structures $e_k = \langle s_k^1, s_k^2 \rangle$, the most effective re-ranking kernel is:

$$\begin{aligned} K_R(e_1, e_2) &= K(s_1^1, s_2^1) + K(s_1^2, s_2^2) \\ &- K(s_1^1, s_2^2) - K(s_1^2, s_2^1), \end{aligned} \tag{4.1}$$

where K could be any kernel function, in our models we used the kernels

described in Section 3.3.4, i.e. String Kernel (SK), Syntactic Tree Kernel (STK) and Partial Tree Kernel (PTK).

This re-ranking schema, consisting in summing four different kernels, has been already applied in [14, 88] for syntactic parsing re-ranking, where the basic kernel was a Tree Kernel.

In [89] a re-ranking model was applied to different candidate hypotheses for machine translation but the goal was different and, in general, simpler: in our task the best annotation of a given input sentence must be learned while in [89], the model learns to distinguish between *"good"* and *"bad"* translations of a sentence. Our approach, for semantic parsing re-ranking, is more appropriate since there is only one best hypothesis for each sentence, while in machine translation a sentence can have more than one correct translation.

The re-ranking kernel in Equation 4.1 can be generalized to the case of t-uple of structures, i.e. several different structures can be associated to each hypothesis and used with different kernels, these are afterwards put together to build instances of n structures. Thus, the generic instance e_k has the form: $\langle s_k^1 ... s_k^n \rangle$, where the first $\frac{n}{2}$ structures are associated to the first hypothesis and the others to the second. The generalized form of the re-ranking kernel results to be:

$$\begin{aligned} K_R(e_1, e_2) &= \sum_{i=1}^{n/2} (K_i(s_1^i, s_2^i) + K_i(s_1^{i+n/2}, s_2^{i+n/2})) \\ &- \sum_{i=1}^{n/2} (K_i(s_1^i, s_2^{i+n/2}) + K(s_1^{i+n/2}, s_2^i)), \end{aligned} \tag{4.2}$$

where K_i are $n/2$ kernel functions used with the $n/2$ structures associated to each hypothesis.

The re-ranking approach brings several advantages, but also some disadvantages. Combining two models can put together their benefits but also

their drawbacks: (a) the SFST model affects re-ranking since it always outputs **"null"** concept for Out-of-Vocabulary (OOV) words in all hypothesis and (b) the re-ranker training time is linked to SVMs implementations, which are trained with a quadratic programming algorithm.

Finally, two main advantages can be observed in the re-ranking model: the first is the ability to put together characteristics of two different models encoding complex features from generated hypotheses. More important, using kernels like String and Tree Kernels (see definitions in Section 3.3.4), the re-ranking model can capture arbitrarily long distance dependencies, unlike the other models described in this work.

4.5 Re-ranking and ROVER for models combination

As was mentioned in Section 3.3.5, another approach used in SLU for models combination is ROVER [34]. ROVER has been applied for the first time to SLU in [38].

ROVER and re-ranking are two very different approaches for models combination:

1. ROVER performs an alignement of different system outputs at token level using a weighted voting scheme, it allows potentially to find the most correct hypotheses subparts. Re-ranking finds the best semantic annotation for every input sentence, it works on sentence units.

2. Since it uses a voting scheme, ROVER needs the outputs of at least 3 systems to yield improvements with respect to individual system baselines. Re-ranking takes output of a single system.

3. ROVER is a light-weight system combination approach, in the sense that training voting weights is relatively fast. Re-ranking is trained

on the output of another model, the cost of training depends on the task complexity and the training data size.

Regarding point 1, it is difficult to find a priori which is the best solution, finding correct hypotheses subparts seems more difficult than finding an alternative hypothesis for an input sentence, but the first solution gives the possibility to find the correct interpretation of a sentence even if all the combined hypotheses contain mistakes.

Point 2 is clearly a disadvantage for the ROVER approach, ROVER relatively a not expensive solution for models combination, but training three (at least) systems for SLU can be quite expensive. In [38], 5 systems were combined, comprising a CRF and a SVM model, which are quite expensive. Point 3 is actually related to point 2.

A comparison of performances of these two approaches will be given in Chapter 6, together with all results.

Chapter 5

Improved Strategies for Reranking-based Models Combination

Joint models based on discriminative re-ranking described in previous chapter have shown to be effective with respect to the other state-of-the-art models used for SLU ([28, 27]). The aim when using a re-ranking approach for models combination is to find an alternative interpretation to the one provided by the re-ranked model. Even using information encoded in the first model hypotheses and extracting more complex features, re-ranking models have limitations that impact on the final performance:

i) Since the model is trained using information in the best interpretation of the first model, it is assumed that the re-ranking model always provides an interpretation more (or at least as) correct than the previous best. In general, the alternative interpretation provided as final result, can contain more mistakes than the best interpretation output by the first model;

ii) While for training phase we can measure the correctness of each hypothesis computing the edit distance with respect to the reference manual annotation, no assessment is performed in classification phase on the hypotheses generated by the first model. The first m-best hypotheses, ranked

by the probability of the first model, are kept for re-ranking.

The first point is addressed applying confidence based model robustness assessment, the second is overcome using a new hypotheses selection criterion recently defined.

In the next section we describe how to re-select the final best interpretation using the confidence measures provided by the two models involved in the re-ranking approach. In Section 5.2 we describe an hypotheses selection criterion that allows selecting hypotheses generated by the first model that are likely to be more correct than the others.

5.1 Confidence-Based Models Combination: Re-Rank Selection (RRS)

The fact that the re-ranking approach can improve an individual system is related to the way the two models encode prior knowledge in the training data. In particular a re-ranking model is trained on the hypotheses generated by another model in such a way that allows to "correct" mistakes made by the first model, in the sense that the result provided after re-ranking is not a new generated one, it is an alternative, possibly more correct, result of the first individual model.

These points can lead to assume that re-ranking is always more accurate than the re-ranked model. Unfortunately this is not correct, in some cases, i.e. for some input sentences, re-ranking can provide an interpretation less accurate than the previous best.

Results provided by stochastic models can be assessed using confidence scores computed on the output. CRF models provide posterior probabilities that can be used as confidence measures. The SFST model provide a hypotheses level likelihood computed with stochastic conceptual language model it encodes. The SVM model used for re-ranking provide as score

the margin, i.e. the distance of the interpretation from the hyperplane, in the features space induced by the used kernel. The Likelihood of the SFST model can be converted into a posterior probability applying the forward-backward algorithm on the semantic network resulting from the combination of the input sentence with FST encoding the language model.

The margin of SVM can be conferted into a posterior probability using a sigmoid transform: let C be the class predicted by SVM ("+1" or "-1"), f the margin computed with the kernel on the given hypothesis, then the margin is converted into probability with the following function

$$P(C|f) = \frac{1}{1 + e^{A \cdot f + B}} \tag{5.1}$$

where A and B are two parameters trained with the Platt's algorithm ([91]).

The posterior probabilities computed by each model involved in the re-ranking process can be used as confidence measures to provide an assessment of results. This means that when the confidence is low, below a certain threshold, we can decide to reject the result of the model. Since two models are involved in the re-ranking framework, when the confidence of one model is low, we can take as final result the output of the other model.

What it follows is based on considerations and analysis made on the output of our joint models. Although results have not been described yet, we prefer to describe the strategy based on hypotheses re-selection in this chapter to have an overview on the entire work we have done on re-ranking, giving afterwards a description of results obtained with all the different models.

Considering the final result, a re-ranking model has a better performance of the re-ranked model. This means that the idea of rejecting the re-ranked best hypothesis can improve the overall performance of the com-

bined models only under the following assumption: the two models used in the re-ranking framework make different mistakes on different input sentences. This seems intuitive given that the two combined models have different characteristics, nervetheless it is a strong assumption, so an error analisys on the models output is mandatory to proove the effectiveness of the rejection idea.

In order to show the evidence, in Figure 5.1 we report the confusion matrices of the FST and SVM based re-ranker model on the LUNA Italian test set (described later), together with the difference between such matrices.

The confusion matrix is constructed putting in the columns the reference concepts and in the rows the hypothesized concepts. The entry of the matrix with coordinates i, j reports the number of times the reference concept j is confused with the concept i, so the entries on the diagonal reports how many times concepts are correctly recognized. Given the high accuracy of the models, the values on the diagonal would be much higher than all the others, so to emphasize only errors, we removed such values.

With reference to Figure 5.1, in an extreme case in which FST and the re-ranker models make the same mistakes, the difference matrix would contain only zeros and so no picks would be depicted. The fact that the difference matrix is not flat suggests that the two models tend to make different mistakes. In particular, reversed picks are values where the re-ranker makes more mistakes than the FST model on the corresponding concept. Notice also that the error analysis described refers to the LUNA Italian test set, so a small data set. This choice is needed to obtain a readable plot.

The presented analysis shows that the re-ranking model improves the overall performance of SLU, but it cannot "correct" all the mistakes made by the FST and even more it introduces other mistakes. Figure 5.1 remarks

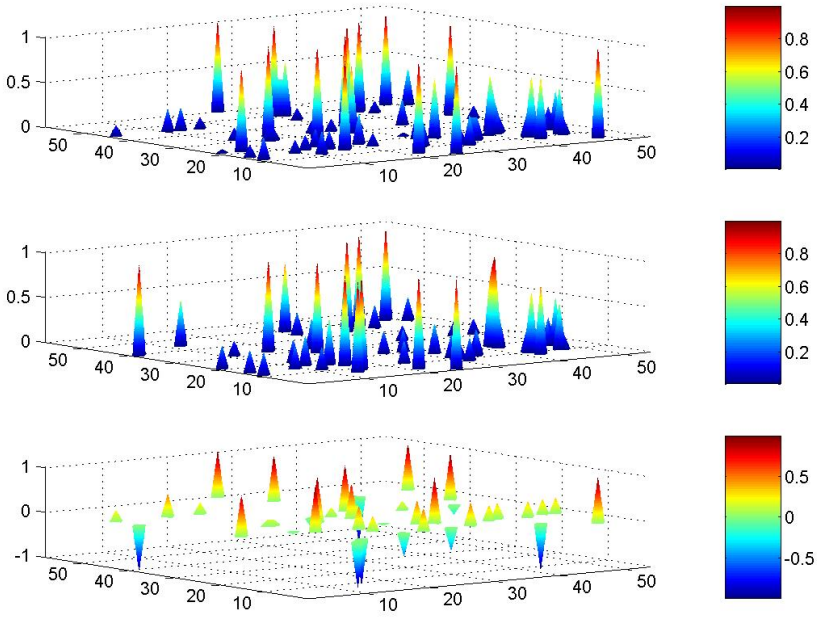


Figure 5.1: Confusion matrixes computed from the output of the FST and SVM based re-ranker (first and second plots) and their difference (third plot). On the axis concept identifiers are reported: values are normalized on columns to emphasize errors for each reference concept.

that in several cases, i.e. for several concepts, the two models have a completely different prediction accuracy. This is reflected in the annotation error rate of the hypotheses of the two models. In particular, there are cases where the best hypothesis provided by one model is much less accurate than the best hypotheses provided by the other model and there are cases where the opposite situation holds. Using features of these hypotheses combined with their scores provided by the respective models, it is possible to infer when the re-ranking model provides a best hypothesis less accurate than the best hypotheses of the SFST model. For these cases, we reject the re-ranked best hypothesis and keep the first model best hypothesis. We call this approach "Re-Rank Selection" (RRS).

In order to apply Re-Rank Selection in practice, we use the two mod-

els posterior probabilities associated with the respective best hypothesis as features and we train two thresholds for these scores. Given such optimal thresholds, we choose the best SLU interpretation with the following decision function:

$$BestHypothesis = \begin{cases} HYP_{RR} & \text{if } C_{fst/crf} \leq T_{fst/crf} \text{ and } C_{RR} \geq T_{RR} \\ HYP_{fst/crf} & \text{otherwise.} \end{cases}$$

where HYP_{RR}, $HYP_{fst/crf}$, $C_{fst/crf}$ and C_{RR} are the best hypotheses and the score associated with the best hypotheses of the re-ranked (FST or CRF) and re-ranking models, respectively. $T_{fst/crf}$ and T_{RR} are the two thresholds trained for the decision function.

5.2 Hypotheses Selection Criteria

Together with the structured features design, the most important part of our re-ranking framework is the hypotheses selection criteria. These criteria affect how hypotheses are chosen to create pairs to be re-ranked. In contrast with what we have done in our previous work, described in [28, 27], the new hypotheses selection is more sophisticated and yields an effective measure of semantic inconsistency for SLU hypotheses. In contrast with error rate (ER), that is used to select the best hypothesis in training phase, the inconsistency metric is measured directly on a single hypothesis and so can be used for both training and classification phase since it doesn't make use of the reference annotation. This is a very important point for re-ranking models since such a measure open a room for further studies and improvements on hypotheses selection criteria.

The hypotheses selection criteria applied in previous works, was based on the simple semantic-tags annotation of the input sentence. Using the same SLU example reported in section 3.2 ("*Buongiorno io ho un prob-*

lema con la stampante da questa mattina non riesco piuá stampare" which translates to "Good morning I have a problem with my printer since this morning I cannot print any more"),

null{Buongiorno io ho} **HardwareProblem.type**{un problema} **Peripheral.type**{con la stampante} **Time.relative**{da questa mattina} **HardwareOperation.negate**{non} **null**{riesco piu'} **HardwareOperation.operationType**{a stampare}

the corresponding segmentation and annotation produced by the SLU model is

null{Buongiorno} **null**{io} **null**{ho} **HardwareProblem.type-B**{un} **HardwareProblem.type-I**{problema} **Peripheral.type-B**{con} **Peripheral.type-I**{la} **Peripheral.type-I**{stampante} **Time.relative-B**{da} **Time.relative-I**{questa} ...

where not all tokens have been reported to keep readability. In previous work, all the hypotheses were compared with the reference transcription via the edit-distance using this representation, the best hypothesis was then used with all the other in the n-best list to create pairs as described previously.

In classification phase, since the ER is measured with respect to a reference annotation, no check was done on correctness of hypotheses. The only contraint was unicity of hypotheses after re-segmentation of concepts, i.e. if the phrase"*la mia stampante*" ("*my printer*") is labeled as

Peripheral.type-I{la} **Peripheral.type-I**{mia} **Peripheral.type-I**{stampante}

since BIO-like markers are just a convention to yield one-to-one association between words and concepts and the first word "*la*" is in any case the first word of the phrase, this was re-segmented as

Peripheral.type-B{la} **Peripheral.type-I**{mia} **Peripheral.type-I**{stampante}

Beyond this constraint, the m-best with respect to the first model score were kept and then re-ranked.

The new hypotheses selection criteria makes use of attribute-value extraction, which in our case is a module based on regular expression rules. A set of rules is defined separately for each concept and, if the input phrase matches one of these rules, the corresponding value is returned. Using this module, the semantic interpretation extracted from the example above would be:

HardwareProblem.type[problem] **Peripheral.type**[printer] **Time.relative**[morning]
HardwareOperation.negate[non] **HardwareOperation.operationType**[print]

where, as described in Section 3.4, "null" concepts are removed since they don't bring any semantic content.

Notice that this representation, since hypotheses are unique in the m-best list, allows to bring in the list much more various hypotheses, possibly more correct. For the same reason this representation is more sparse, from different segmentations of the same concepts, the same interpretation can be produced and only one, for each hypothesis, is kept. For example, if the phrase "*con la stampante*" (litterally "with the printer") is annotated in different hypotheses as:

- **Peripheral.type**{con la stampante}
- **null**{con} **Peripheral.type**{la stampante}
- **null**{con} **null**{la} **Peripheral.type**{stampante}

always the same interpretation would be generated: **Peripheral.type**[printer]. So from three different hypotheses, even if different segmentations are generated, using attribute-value representation only one hypothesis is kept in the n-best list.

This sparseness implies that this hypotheses representation can be adopted only when sufficiently big corpora are used, this way enough different instances are generated for each input sentence and all the meaningful features for each corresponding hypothesis can be captured in training phase. This is also the reason why this representation was not used in previous work ([29],[28]), the previous version of the LUNA Italian corpus was too small and experiments on the big MEDIA corpus are too expensive, in terms of time, to perform preliminar studies.

Using this representation, in classification phase we can compute a consistency (or inconsistency) metric on hypotheses. Since rules can extract values only from phrases containing key words for each concept, e.g. from "*con la*" the value "*printer*" cannot be extracted, while from "*stampante*", even if not appearing in the training data, since regular expressions can easily capture many different phrase variations, we can extract the correct value, we can exploit this property to compute an inconsistency metric simply as the number of possibly incorrect values in a hypothesis. A value, given a concept, is correct if we have the same attribute-value pair in the training set.

Moreover, since usually a robust word categorization is designed for a given task, even

values not appearing in the training set can be assessed for correctness. For example, the corpus MEDIA was provided with a categorization file containing all the cities, the streets, the proper names of places etc. annotated in the corpus as an application knowledge base. Once the model recognize the correct concept (note that this is possible also on OOV words thanks to these categories), e.g. "`+localisation-ville`" ("`localization-city`") in MEDIA, we can impose that the extracted value is in the "`XVILLE`" ("`CITY`") category, otherwise we can assess the given attribute-value pair as inconsistent.

Furthermore, if a phrase that should instantiate a specific concept is wrongly annotated with another concept, the rules used for value extraction cannot extract any value, and since an empty value is usually not admitted, also this can be used as inconsistency measure. For example, again from MEDIA, if "*Marseille*" (a french city) is wrongly annotated as "`+localisation-rue`" ("`localization-street`", this happens since in MEDIA there is also "*rue de Marseille*", "*Marseille Street*"), rules will not be able to extract any value since, for this specific concept, some prefixes are expected in the phrase, e.g. "*street*", "*avenue*", "*square*" and so on, so no regular expression would match the given phrase. Again, an empty value is usually not admitted for any concept so this would mean that most probably the phrase has been annotated with a wrong concept.

Finally, for those values with an infinite domain, e.g. dates and phone numbers, the format can be checked to provide a measure of value inconsistency. For example 30/02, 30/15 or 42/12 are not correct values for dates since February has only 28 days (or 29), the months are only 12 and December has only 31 days, respectively.

Once this inconsistency metric is computed for all hypotheses, they are sorted on increasing inconsistency values and the m-best are used in the classification phase for re-ranking. Note that re-ranking is needed in any case since inconsistency is not perfectly correlated to ER. Two hypotheses with slightly different inconsistency values can have different ER. Thus, the top most consistent hypothesis is not necessarily the most correct in terms of ER. Nevertheless, applying this metric to hypotheses in classification phase, brings in the m-best hypotheses pool of each input sentence much better hypotheses, this way the re-ranker is more likely to find a more correct one.

Beyond simple examples provided here, results will show that this method is very effective and allows to improve significantly the "tradictional" re-ranking model.

Chapter 6

Experimental results

Despite the fact discriminative re-ranking is not a new idea, the fact that we have applied this approach to spoken language and we have not used syntactic analisys as the basis of our structures, imposed a deep study on the behavior of models as a function of several aspects involving both training and classification phases.

Given the high cost of experiments, not all cases have been studied on all corpora. Instead, preliminar studies on models have been conducted on a smaller corpus and only the most promising solutions have been applied to the other corpora.

This experimentation protocol could have missleaded our research line in some cases, nervertheless it is fairly robust, first because we have compared our work with the best state-of-the-art approaches, e.g. Conditional Random Fields, Support Vector Machines, Maximum Entropy, Dynamic Bayesian Networks, second because a model working well in all conditions (small, medium and larger corpora) should be preferred to a model providing good results only in specific conditions.

In the remainder of this chapter, we describe first the corpora used for experiments, then the experimental setup and finally all the results obtained with our joint models. Experiments have been conducted with the aim of providing:

- Comparative results on the kernel structures designed for our tasks (Section 4.3)
- Comparative results on the training criteria
- Comparative results on the robustness of our re-ranking approach with respect to the first model baseline
- Comparative results on the different joint models, FST Re-ranking and CRF Re-ranking

ATIS	training		test	
# turns	4,978		893	
	words	concepts	words	concepts
# tokens	52,178	16,547	8,333	2,800
# Vocabulary	1,045	80	484	69
# OOV rate [%]	–	–	1.0	0.1

Table 6.1: Statistics of the ATIS training and test sets used in the experiments

MEDIA	training		development		test	
# sentences	12,908		1,259		3,005	
	words	concepts	words	concepts	words	concepts
# tokens	94,466	43,078	10,849	4,705	25,606	11,383
# vocabulary	2,210	99	838	66	1,276	78
# OOV rate [%]	–	–	1.33	0.02	1.39	0.04

Table 6.2: Statistics of the MEDIA training, development and evaluation sets used for all experiments.

- Comparative results on different tasks, in particular on training data size and task complexity
- Comparative results on hypotheses selection criteria
- Comparative results on experiments conducted using increasing m-best list size

6.1 Corpora Description

The corpora used in our experiments are the most representative for Spoken Language Understanding. They come from several years of research and provide several important experimentation conditions:

- Different Languages: English, Italian, French and Polish
- Differnet sizes: from the Italian and English corpora to the French and Polish ones, the size of the available training data changes significantly.
- Task complexity: from English to Polish, task complexity changes tremendously, not only for the kind of task modeled, but also for issues inherent to the languages.

LUNA Polish	training		development		test	
# sentences	12,908		1,259		3,005	
	words	concepts	words	concepts	words	concepts
# tokens	53,418	28,157	13,405	7,160	13,806	7,490
# vocabulary	4,081	195	2,028	157	2,057	159
# OOV rate [%]	–	–	4.95	0.13	4.96	0.11

Table 6.3: Statistics of the Polish LUNA training, development and evaluation sets used for experiments.

LUNA Italian	training		development		test	
# sentences	3,171		387		634	
	words	concepts	words	concepts	words	concepts
# tokens	30,470	18,408	3,764	2,258	6,436	3,783
# vocabulary	2,386	42	777	38	1,059	38
# OOV rate [%]	–	–	4,22	0.0	3.68	0.0

Table 6.4: Statistics of the latest version of the LUNA Italian training, development and evaluation sets used for all experiments.

- Data acquisition characteristics: some corpora have been aquired in ideal conditions with the aim of providing data suitable for application development, other corpora have been acquired in real conditions, basically with the aim of studying linguistic phenomena or to provide more realistic conditions.

The Air Travel Information System (ATIS) corpus [26] has been used for the last decade to evaluate models of Automatic Speech Recognition and Understanding. It is made of single turns acquired with a Wizard of Oz (WOZ) approach, where users ask for flight information. Statistics for this corpus are reported in Table 6.1.

The corpus MEDIA was collected within the French project MEDIA-EVALDA [8] for development and evaluation of spoken understanding models and linguistic studies. The corpus is composed of 1.257 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 6.2.

The data for the Polish corpus has been collected at the Warsaw Transportation call-center [59]. As part of the LUNA project, the manual annotation of these human-human dialogues has been performed [71]. This corpus covers the domain of transportation information like e.g. transportation routes, itinerary, stops, or fare reductions. Three subsets

LUNA Italian (tmp)	training		test	
# turns	1,019		373	
	words	concepts	words	concepts
# tokens	8,512	2,887	2,888	984
# Vocabulary	1,172	34	-	-
# OOV rate [%]	–	–	3.2	0.1

Table 6.5: Statistics on the first, intermediate version, of the LUNA Italian corpus

have been created using the available data: a training set comprising approximately 8k sentences, a development and an evaluation set containing roughly 2k sentences each. It is the first SLU database for Polish, statistics of this corpus are shown in Table 6.3.

The LUNA Italian 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 [30]. The data is organized in transcriptions and annotations of speech based on a new multi-level protocol. Experiments described in this chapter have been conducted on two different version of this corpus: the first version has been released in the middle of the LUNA project (approximately March 2008) and it is a small subset (250 dialogs) of the final version. The data of the final version of the Italian corpus are extracted from 723 Human-Machine dialogs (HM) acquired with a WOZ approach. The data have been split in training, development and test sets. Statistics of this corpus are reported in Table 6.4. Additionally, also Human-Human dialogs have been acquired, transcripted and annotated following the same protocol used for Human-Machine dialogs. These data have been used in some of our previous work ([29]), but after the release of the final version of the corpus, Human-Machine data alone were sufficient to conduct our studies and, even more, Human-Human dialogs were too much noisy to provide releable models evaluation.

6.1.1 Differences among corpora

Hereafter, we report shared and different corpus aspects:

First, domain of the application; from this point of view ATIS, MEDIA and the Polish corpora are rather similar, the first is a corpus of flight information and reservation, the second is a corpus of hotel information and reservation, the third has been designed for providing information of the Warsaw transportation system.

Second, data collection paradigm; all corpora, except Polish, have been acquired with a WOZ approach but with a different setup. In ATIS the data acquisition unit is a single

turn, where the users ask flight information. MEDIA and the LUNA Italian are corpora of entire dialogs. The Polish corpus is a corpus of Human-Human dialogs.

Third, size of the data; LUNA is the smallest corpus (3.171 and roughly 1.000 turns for training in the two different versions), while MEDIA is very big (almost 13.000 sentences for training). ATIS and Polish are in the middle with roughly 5.000 and 8.000 sentences for training.

Next, the task complexity is usually measured in terms of number of concepts with respect to the size of the available training data. From this point of view the Italian corpus, with only 42 concepts, would be the simplest task. ATIS and MEDIA would have a comparable complexity since the former includes 69 concepts and the original number of concepts in MEDIA is 65. Nevertheless MEDIA is much more complex since there are concepts with different specifiers and modes (see [8]). Thus the real number of semantic tags to be recognized in MEDIA increases to 99. The most complex task from this point of view is the Polish corpus, with roughly 200 concepts.

It should be noted that the automatic annotation of ATIS can be easier than other SLU tasks since: (a) most sentences have the form: *"Information Request about"* flights from DEPARTURE_CITY to ARRIVAL_CITY TIME, where *"Information Request about"* is one of several ways of asking information, *DEPARTURE_CITY* and *ARRIVAL_CITY* are the names of two cities and *TIME* is the specification of a day and/or hour of departure. This kind of sentences with small variations constitute more than 90% of the corpus. (b) In the data available for the SLU task on ATIS, which is the same used in [81] and in [41], concepts are almost always associated with a single token so there is no need of segmenting them using BIO-like markers as shown in Section 3.2.

As an example the following ATIS sentence:

"I would like a flight from Phoenix to San Diego on April First"

is semantically annotated as:

null{I would like a flight} **null**{from} **departute_city**{Phoenix} **null**{to} **arrival_city**{San-Diego} **departure_date.month**{April} **departure_date.day_number**{first}

while, using the annotation style of the other corpora, the sentence would be annotated as:

null{I would like a flight} **departute_city**{from Phoenix} **arrival_city**{to San Diego} **departure_date.month**{April} **departure_date.day_number**{first}

in this case the concepts **departute_city** and **arrival_city** would have a span of two and

three words respectively. In other words, ATIS only concerns with the problem of token labeling: no concept segmentation is performed in this task. For these reasons, our work on ATIS only relates to concept labeling: values correspond in most cases to the word surface forms or can be retrieved exactly from words.

Finally, the task complexity is also affected by the characteristics of utterances. ATIS and MEDIA were acquired with a WOZ approach with optimal environmental setup (high quality microphones and absence of noise in the channel). The LUNA Italian corpus has been acquired in a noisy environment. Additionally, utterances in this corpus are rather more spontaneous, this adds further noise. Furthermore, the annotation of the turns in the Italian LUNA corpus has been done taking into account turn context. The same words can be annotated with a different concept in case the context is different.

For example, the phrase "it is not working" can be a "**HardwareOperation**" in case it refers to a "**Peripheral**", while it is a "**SoftwareOperation**" if it refers to "**Software**". Beyond the trivial example, this annotation adds complexity to the task.

For these characteristics, even if the number of concepts to be recognized is smaller, the Italian corpus is not simpler than the others.

Regarding the Polish corpus, since it is composed of Human-Human dialogs, as opposed to all the other corpora, it is by far the most complex task. Further complexity is added from the characteristics of the Polish language, which shows some properties of both inflexive (more than Italian and French, since it uses 7 cases and 3 genders) and agglutinative languages (adjectives are often composed with the noun they refer to).

Additionally, many concepts are closely related. To emphasize the complexity of Polish as an inflectional language with a relatively free word order, here are examples of different ways of inflection for Polish location names:

- (jestem) na Polnej$_{adj,fem,loc}$/Dkabrowskiego$_{adj,masc,gen}$

  (I am) on Polna Street / Dkabrowskiego Street

- (jadę) z Polnej$_{adj,fem,loc}$ /Dkabrowskiego$_{adj,masc,gen}$

  (I am coming) from Polna Street / Dkabrowskiego Street

- (jadę) na Polnka$_{adj,fem,acc}$ / Dkabrowskiego$_{adj,masc,gen}$

  (I am going) to Polna Street / Dkabrowskiego Street

In theses phrases are three different concepts describing places: LOCATION_STR, SOURCE_STR and GOAL_STR (STR is an abbreviations for street).

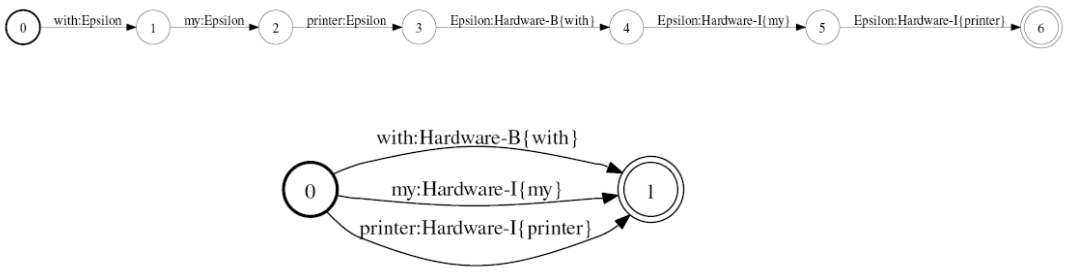


Figure 6.1: Comparison between the two approaches for mapping words/categories into concepts used in the SFST model described in [81] (above) and the one used for our modified SFST model (below).

6.2 Experimental Setup

The experimental setup reported in this section is in general referred to all experiments described later. If experiments have been performed with different setup, this will be specified case by case.

An important aspect regarding experiments is related to the SFST-based model described in Section 3.3.1. During the research work described in this document, the SFST-based model has been improved on different aspects. Practical processing problems have been detected in the original model, thus we modified the structure of SFSTs involved in concepts decoding. We give the explanation of these modifications with some examples. First, given the phrase "with my printer" that can be tagged with the concept "**Hardware**", the model described in [81] and our model perform mapping from words to concepts (with the transducer λ_{W2C}, see Section 3.3.1) in two different ways, shown in Picture 6.1, above for the original model and below for our model.

These two approaches have different advantages and disadvantages:

Since the model presented in [81] emits the concept tagging only after matching the entire surface, it is more rigid. If the test set contains surfaces not seen in the training set, the transducer cannot traverse the entire path reaching the final state, so no concept is emitted. If for example in test phase the phrase "the printer" is given as input, that should be tagged again as "**Hardware**", the model would not tag the phrase since the fst cannot be traversed with this phrase. Our approach would always tag the given surface in these cases, the BIO markers used to segment the concept are just a convention to yield a one-to-one association between words and concept tags, they are removed in a post-processing phase to reconstruct the concept name.

On the other hand, our approach can produce insertion problems since it emits tagged

words also if only partial phrases are given. Nevertheless this problem is unlikely since the Stochastic Conceptual Language Model used as last step in the decoding process, gives much higher scores to well formed phrases.

Another improvement of our model is in the generalization step of decoding (performed by λ_G), mapping words into word categories.

The original SFST model performs the generalization step mapping a word into its category. Categories are defined in an external application knowledge base. After the generalization phase, categories are mapped into concepts concatenated to the corresponsing category. As an example, taken from the French corpus MEDIA, if the word "Marseille" belongs to the category "CITY" and can support the concept "**Location**", in the original model there are the following steps, the first in the FST λ_G, the second in λ_{W2C}:

1. *Marseille* → CITY
2. CITY → **CITY.LOCATION-B**

where "*B*" is again the marker for concept beginning. This way the model can correctly tag also city names not seen in the training set if these cities are defined in the knowledge base: suppose the city "Bordeaux" is not in the training set, but it is defined as "CITY" in the application data base, then the generalization FST maps "Bordeaux" into "CITY" and the correct concept tagging can be performed.

Nevertheless, since the knowledge base of the application is defined a priori, with no regards to which words realize the different concepts, it is possible that two words realizing two different concepts belongs to the same category. In this situation the two words can be tagged with each of the two concepts, even if this association was not seen in the training set. For example, again from MEDIA, in the training set there is the word "Italy" instantiating the concept "**LOCALISATION-STREET**" or "**LOCALISATION-COUNTRY**" and the word "Alsace" instantiating the concept "**NAME**". "Italy" and "Alsace" belong to the same category "XCOUNTRY".

This means that in the generalization FST λ_G there are the paths:

1. *Italy* → *XCOUNTRY*
2. *Alsace* → *XCOUNTRY*

then, in the FST mapping categories into concepts (λ_{W2C}) there are the paths:

1. *XCOUNTRY → XCOUNTRY.LOCALISATION-STREET-B*
2. *XCOUNTRY → XCOUNTRY.LOCALISATION-COUNTRY-B*
3. *XCOUNTRY → XCOUNTRY.NAME-B*

As concequence, "Italy" can be tagged also as "**NAME**" and "Alsace" can be tagged as "**LOCALISATION-STREET**" or "**LOCALISATION-COUNTRY**", regardless if these associations were seen in the training set. In general, any word belonging to a given category can be mapped to any concept associated to this category. Note that this provide a very high generalization power, nevertheless it adds a lot of noise, the score given by the SCLM alone (λ_{SLM}) will not discriminate between correct and wrong taggings, the most frequent associations will be always preferred, taking into account also the context.

In order to overcome this problem, we have redefined the generalization step using concept-dependent categories. Considering the same example above, the two steps of the decoding process performed by λ_G and λ_{W2C} are, respectively:

1. *Italy → XCOUNTRY.LOCALISATION*
2. *Alsace → XCOUNTRY.NAME*

1. *XCOUNTRY.LOCALISATION → XCOUNTRY.LOCALISATION-STREET-B*
2. *XCOUNTRY.LOCALISATION → XCOUNTRY.LOCALISATION-COUNTRY-B*
3. *XCOUNTRY.NAME → XCOUNTRY.NAME-B*

Finally, another improvement with respect to the original FST model is in the tuning of the SCLM. As explained before, instead of using AT&T tools to train the language model, we use SRILM tools.

All these small improvements together, provide a more effective model with performances much closer to state-of-the-art models with respect to its original implementation.

All the Stochastic Conceptual Language Models (SCLMs) that we apply in the experiments either for the FST model baseline or to produce the input for the re-ranking models, are trained with the SRILM toolkit [90]. All SCLMs are interpolated models to which the Kneser-Ney discount technique has been applied [12]. With only exception for the Polish corpus, where the best model resulted in a *4*-gram language model, for all the other corpora *3*-grams are used. The SCLMs are converted into SFST using SRILM toolkit. The decoding process for the SFST model is performed using AT&T FSM/GRM Tools

[63]. N-grams order, best discount technique as well as other language model paramters (e.g. minimum count for lower order n-grams) have been optimized on the development set of each corpus.

The model used to obtain the SVM baseline for concept classification was trained using YamCHA [48], available at http://chasen.org/∼taku/software/yamcha/. The CRF model was trained with the CRF++ tool, available at http://crfpp.sourceforge.net/ (from the same author of YamCHA). The setting that we used for it is equivalent to the one described in [38], which is the state-of-the-art on the corpora we consider[1]. We used slightly different features for SVM and CRF baselines: for both we used word categories and morpho-syntactic features, for SVM we added some Yes/No features (e.g. "does the word contain symbols?", "does the word contain numbers?","is the word preceded by symbols" etc.). The feature windows, i.e. the features around the current position considered to predict the label for the current word, are different for the different tasks when using CRF++: [-2, +2] for ATIS, [-1,+1] for MEDIA, [-3,+1] for Italian and [-1,+1] for Polish. Only bigrams of concept tags can be used for CRF (see CRF++ web site and [38] for more details). The window used for YamCHA instead is always [-3,+3] (see YamCHA web site) and only the previous concept with respect to the current position is used to predict the label for the current word. All these settings, like for SCLMs, have been optimized on the different development sets.

The re-ranking models based on structure kernels and SVMs were trained using the SVM-Light-TK toolkit (available at htttp://disi.unitn.it/moschitti). The number of hypotheses used for re-ranking was always set to 10.

For the ATIS experiments, we did not apply any parameter optimization, i.e. we used default parameters or parameters from previous work. For the first version of the LUNA Italian corpus (see Table 6.5), since no developement set was available, we optimized parameters with a 3-fold cross-validation on the training set.

The results are expressed in terms of concept error rate (CER). This 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. When not specified, CER is computed only on attribute names (**Attr**), otherwise CER is computed for both attribute names and values (**Attr+Val**). Regarding attribute-value extraction techniques, in all our experiments and for both FST, SVM, CRF baselines and for our joint models, we always used rule-based approaches, while results

[1]In [38], CRFs are compared with other four models (Stochastic Finite State Transducers, Support Vector Machines, Machine Translation, Positional-Based Log-linear model) showing that it is by far the best model on the MEDIA corpus.

reported in [38], in [37] and all the other state-of-the-art results we will compare with, unless it is specified explicitely, are obtained using combined rule-based and stochastic approaches.

All models have been tested on two different kind of inputs: manual transcriptions of utterances and automatic transcriptions. The latter were produced by speech recognizers with a WER of 10.4% on the ATIS test set, 28.5% 27.0% on the Italian developemnt and test sets respectively, 30.3%, 31.4%, 39.5% and 38.9% on MEDIA and Polish development and test sets, respectively.

Additionally, only for the Italian task, FST Re-ranking has been applied also on ASR lattices. As mentioned in Section 3.3.1, the SFST model is suitable to be applied to word lattices, since the latter are naturally encoded as SFST. Also CRFs can be encoded as SFST, but this is not possible with the publicly available tool we used. Some work in this direction has been done by the LUNA project partner from RWTH Aachen University (http://www.rwth-aachen.de/go/id/bdz/). At the moment, applying CRF on word lattices doesn't give any improvement with respect to using ASR 1-best, same result is obtained applying FST Re-ranking, so, beyond some results shown in this work, we didn't spend any further effort to investigate in this direction.

In all cases the language model used for ASR is an interpolated model with Kneser-Ney discount [12], which gives a better performance in most cases.

Another important point in our experiments relates to different corpora versions. Previous work ([29, 28]) as been done on different version of the Italian and French corpora, so those results, in terms of the final scores, cannot be compared with the other experiments reported here, but still they provide significant studies on our joint models as well as on the structures used with kernels.

6.3 Results Description

6.3.1 Comparison of Training Approaches and Pairs Generation Strategies

In the joint models based on re-ranking, the first model generates the m-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 corpus and the method to generate the m-best list. We apply two different methods for training: **Monolithic Training** and **Split Training**.

In the former, SFSTs are learned with the whole training set. The m-best hypotheses

generated by such models on the training set itself are then used to train the re-ranking classifier. In other words, the data used to train the SFST and the data used to generate the hypotheses are the same data, the training set of the corpus. This may seem a trivial approach, nevertheless generative models like SFST don't overfit complitely training data, so erroneous hypotheses can still be generated.

In Split Training, the training data are divided in two parts to provide more various hypotheses for each input sentence and to generate hypotheses on unseen data. More in details, we train SFSTs on part 1 and generate the m-best hypotheses using part 2 as a test set. Then we re-apply this procedure 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 hypotheses on the test (or development) set using the SFSTs trained on all training data (as usual).

Regarding the generation of the training instances $\langle s^i, s^j \rangle$, we set m to 10 and we choose alternatively 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 SFST model;

`(C)` as above but s^i is the best annotation among the 100-best hypotheses.

Experiments with these settings have been performed using also a perceptron (PCT) as basic learning algorithm for the re-ranking models.

Results are shown in Tables 6.6, using Monolithic Training (MT), and 6.7, using Split Training (ST). These tables show the best results obtained with the two training approaches described above and the best semantic structure (described in Section 4.3) chosen for each kernel used: "*FLAT*" semantic tree for the Syntactic Tree Kernel (STK), "*FEATURES*" semantic tree for Partial Tree Kernel (PTK) and "*SK1*" sequence structure for the String Kernel (SK). These results show: first, the perceptron is much less effective than SVMs providing in many cases worst results with respect to the SFST model baseline. Results are also very different using the two training approaches and the different pairs generation strategies. This means that perceptron is most probably not really suitable for SLU re-ranking in these conditions, even if some effective approaches based on perceptron, e.g. [14], have been shown for syntactic parse re-ranking.

Comparing results of Table 6.6 and Table 6.7 we can conclude that SVMs is more effective than perceptron and that "*Split Training*" (ST) is a better approach for learning the re-ranker. This is suggested first by our intuition. MT learning provides biased hy-

WOZ	Monolithic Training					
	SVM			PCT		
	STK	PTK	SK	STK	PTK	SK
RR-A	**18.5**	18.6	19.1	24.2	28.3	23.3
RR-B	**18.5**	19.3	19.0	29.4	23.7	20.3
RR-C	**18.5**	19.3	19.1	31.5	30.0	20.2

Table 6.6: Results of experiments, in terms of Concept Error Rate (CER), on the LUNA WOZ corpus using Monolithic Training approach. The baseline with FST and SVMs used as individual models are **23.2%** and **26.7%** respectively.

WOZ	Split Training					
	SVM			PCT		
	STK	PTK	SK	STK	PTK	SK
RR-A	20.0	18.6	**16.1**	28.4	29.8	27.8
RR-B	19.0	19.0	19.0	26.3	30.0	25.6
RR-C	19.0	18.4	16.6	27.1	26.2	30.3

Table 6.7: Results of experiments, in terms of Concept Error Rate (CER), on the LUNA WOZ corpus using Split Training approach. The baseline with FST and SVMs used as individual models are **23.2%** and **26.7%** respectively.

potheses since, as stated earlier, the data used to train SFST and to generate hypotheses are the same. Even if generative models don't overfit completely the training data, hypotheses generated in these conditions contain relatively few mistakes, that are essential for the re-ranker in order to learn which structures are meaningful for both learning which features are contained in correct hypotheses and which features are contained in wrong hypotheses, so that to distinguish between them. Second, results are much more various using ST approach and, in average, better than using MT. Third, results reported in these section are obtained on the first temporary version of the LUNA Italian corpus, which is a small and simple task. This explains results obtained with the Syntactic Tree Kernel (STK) and MT approach in Table 6.6: the STK is more rigid than the other kernels, since it doesn't split tree children at each node. So using MT, that provides very accurate hypotheses on such simple task, allows finding eventually the most significant features. These conditions are specific for this task, using the STK kernel and, additionally, using the semantic structure shown in Figure 4.2(a).

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 6.8: CER of SVMs using STK, PTK and SK on the LUNA Italian corpus (manual transcriptions). The Baselines, FST and SVMs alone, show a CER of **23.2%** and **21.0%**, respectively.

6.3.2 Comparison of Kernels and Semantic Structures

Table 6.8 shows the accuracy of different kernels applied to the different semantic structures described in Section 4.3: The simple "*FLAT*" semantic tree, the "*MULTILEVEL*" and "*FEATURES*" semantic trees. These experiments were conducted using the approach "*ST*" for training and the strategies "*C*" for pairs generation, that have shown better results, on average, in the preliminar studies. Additionally, for these experiments we have optimized kernel parameters, i.e. vertical decay factor for Syntactic Tree Kernel and Partial Tree Kernel and sub-sequences length decay factor for Partial Tree Kernel and String Kernel (see Section 3.3.4). Such results are reported also in our previous work [28] and, like results in the previous section, are obtained using the first version of the LUNA Italian corpus. We exploit this outcomes to motivate our choice of the best combination of kernels and structures to be used for the comparison against the state-of-the-art. The dash symbol appears where the structure cannot be applied to the corresponding kernel. Note that for this task, re-rankers significantly improve the baseline results, i.e. 23.2% (CER for FST) and 21.0% (CER for SVMs). 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 rather different results depending on the used kernel. We can see in Table 6.8 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 mentioned earlier, and so structures like "*MULTILEVEL*" and "*FEATURES*" result too rigid and prevent STK to be effective. In contrast, the structure "*FLAT*" is rigid as well, but since it is very simple and has only one level of nodes it can capture the most meaningful features.

Not all the results using different kernels and structures are reported for all corpora,

our studies in this direction were conducted only on the Italian and on the French corpora. It's worth underline that as MEDIA is a more complex task (42 concepts in LUNA, 99 in MEDIA) and it is a noticeable larger corpus, the more complex structures are also more effective to capture word-concept dependencies.

Table 6.8 shows that a String Kernel with the structure "*SK1*" is the most effective with a CER of 16.2%. Nevertheless the String Kernel is also the most complex kernel and its computational complexity prevents using it on big corpora, e.g. MEDIA and the Polish corpora. For this reasons, we have adopted the Partial Tree Kernel (PTK) with the most complex tree structure "*FEATURES*" in all the following re-ranking experiments: this is the best trade-off between accuracy (i.e. a CER of 18.4%) and computational complexity [2].

All the results reported so far made up our preliminar studies in order to understand the performance of the re-ranking models as a function of training approaches, pairs generation strategies, semantic structures and kernel function adopted. All the following results have been performed using the best settings found in the preliminar studies:

- "*Split Training*" (ST) training approach
- "*C*" strategy for pairs generation
- Partial Tree Kernel (PTK) as kernel function
- "FEATURES" semantic tree structure

Additionally, all results reported from now on are obtained on official version of corpora, so they can be compared with state-of-the-art results that will be given along the discussion.

6.3.3 Cross-Corpora Results Comparison

In this section we compare our joint models across different corpora, i.e. ATIS, MEDIA and the LUNA Italian corpus, respectively.

In Table 6.9 are reported the results on the ATIS corpus. Given the characteristics of this corpus discussed in Section 6.1.1, all models are very accurate, even using automatic transcriptions coming from an ASR system. Nevertheless is worth discussing some

[2]Some other experiments have been performed on MEDIA for semantic structures and kernels comparison that have confirmed that this choice is the most effective, we have not reported these results since on MEDIA, in some cases, experiments have not been possible for resource request problems, our choice takes also these aspects into account

Model	$ATIS_{text}$ **(CER)**	$ATIS_{speech}$ **(CER)**
FST	6.7%	13.5%
SVM	6.9%	13.8%
CRF	7.1%	14.0%
FST+RR (PTK)	**6.2**%	**13.2**%

Table 6.9: Results of SLU experiments on the ATIS Corpus using manual ($ATIS_{text}$) and automatic transcriptions ($ATIS_{speech}$), in the latter case Word Error Rate (WER) of the ASR is 10.4%.

ATIS concepts	**counts**
ArrivalCity	5043
DepartureCity	5018
DepartureDate.day_name	1100
AirlineName	802
DepartureTime.period_of_day	683
DepartDate.day_number	450
DepartDate.month_name	435
DepartTime.time	426
RoundTrip	421
DepartTime.time_relative	387

Table 6.10: Top most occurring concepts in the ATIS corpus.

interesting outcomes. The errors made on the ATIS test set are caused by an unbalanced amount of instances of concepts. In Table 6.10 unigram counts of concepts in the ATIS corpus are reported. As it can be seen from this counts, the concepts **DepartureCity**, **ArrivalCity** and **DepartureDate.day_name** are by far the most frequent (57.7% of the total counts). This means, for instance, that the models are strongly biased to annotate a city as *Departure* or *Arrival* city, even if the context "suggests" another concept. Note that the re-ranking model, FST+RR (PTK), even in this situation, can improve individual systems. The improvement is only .5% points on manual transcriptions, with respect to the re-ranked FST model, since in any case the percentage of such errors with respect to the total number of reference concepts is very small.

Experimens conducted on ATIS had the goal of comparing our models with previous state-of-the-art approaches, e.g. [41]. In particular, the Hidden Vector State model described in [41] is compared with a FST model comparable, in terms of model characteristics, with our FST baseline model. Nevertheless, we have explained how the FST model

Model	MEDIA(CER)		LUNA IT(CER)	
	Attr	Attr+Val	Attr	Attr+Val
FST	14.2%	17.0%	24.4%	27.4%
SVM	13.4%	15.9%	25.3%	27.1%
CRF	**11.7**%	**14.2**%	**21.3**%	**23.5**%
FST+RR	11.9%	14.6%	**21.3**%	23.7%

Table 6.11: Results of SLU experiments on the MEDIA and the Italian LUNA test sets on manual transcriptions for both attribute names (Attr) and attribute values (Attr+Val)

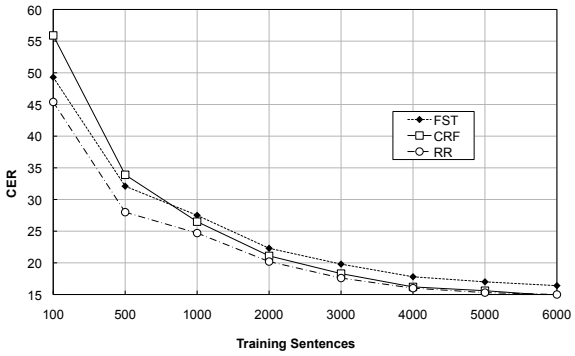
Model	MEDIA(CER)		LUNA IT(CER)	
	Attr	Attr+Val	Attr	Attr+Val
FST	28.9%	33.6%	36.4%	39.9%
SVM	25.8%	29.7%	34.0%	36.7%
CRF	**24.3**%	**28.2**%	**31.0**%	**34.2**%
FST+RR	25.4%	29.9%	32.7%	36.2%

Table 6.12: Results of SLU experiments on the MEDIA and the Italian LUNA test sets on automatic transcriptions for both attribute names (Attr) and attribute values (Attr+Val) extraction (ASR WER is 31.4% for MEDIA and 27.0% for LUNA IT)

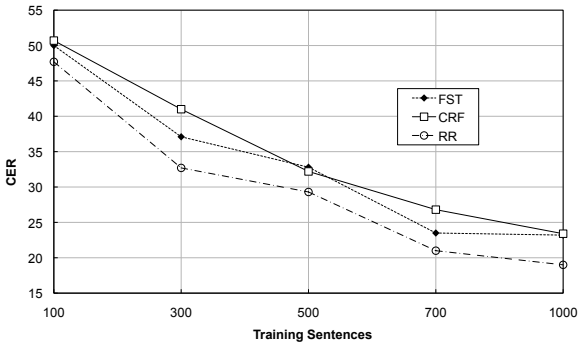
we use is different from the one used in [81]. Even without the improvements described in Section 3.3.1, with our SFST model we achieve a F-1 measure of 92.68%, which is already fairly higher than the result achieved in [41] (89.28%). Beyond this comparison on ATIS, the remainder of our experiments are performed on the other three corpora considered in this work.

Table 6.11 shows the results of the SLU experiments on the MEDIA and Italian LUNA test sets using the manual transcriptions of spoken sentences. Note that on a big corpus like MEDIA, the baseline models (FST and CRF) can be accurately learned thus less errors can be "corrected" (this is the effect of re-ranking from another point of view). As a consequence, the re-ranking approach does not outperforms CRF, that results the best model, but it still improves the FST's baseline of 2.3% points (16.2% of relative improvement).

The behavior of these models doesn't change on automatic transcriptions from ASR, shown in Table 6.12, where CRF shows a better robustness to the noisy speech input. In any case a good performance is obtained again also with the re-ranking model, which improves significantly the FST model baseline in all cases. For example, on the MEDIA corpus, for attribute-value extraction, the re-ranking model improves the baseline of 3.7%



(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 6.2: Learning curves on MEDIA and LUNA corpora using FST, CRF and RR on the FST hypotheses

points (13.1% relative improvement).

6.3.4 Impact of Training Data Size

The different performance of the re-ranking model on the LUNA and MEDIA corpora, in terms of achieving the state-of-the-art, is due partially to the task complexity and partially to the heavy optimization of CRFs on MEDIA corpus (see [39, 38]). The re-ranking model can be relevantly improved as well with respect to several aspects, e.g. by applying parameter optimization, by designing new structural features, improving the re-ranked model baseline etc.

Moreover, with the settings described above, the re-ranking models achieve the highest accuracy for automatic concept annotation when a small data set like the Italian one is used. To show this, in figures 6.2(a) and 6.2(b), we report the learning curves obtained using 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, so more appropriate for this expensive experiment on MEDIA. The second plot refers to the re-ranker accuracy using SK applied to *SK1* structure, which is the best on the Italian corpus and still can be applied on such small corpus.

In these figures, the FST baseline performance is compared with re-ranking (RR) and a Conditional Random Field (CRF) model. The above curves clearly shows that for small datasets the 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, e.g. STK. In particular STK is improved by $18.5 - 16.2 = 2.3$ points (Table 6.8).
- Although the training data is small, the re-rankers based on kernels appear to be very effective. This may also open a room of discussion about the burden of annotating large amount of data.

6.3.5 Impact of Re-Rank Selection and Hypotheses Selection Criteria

Results described in this section aims at showing the benefits of the improvements that we have described in Chapter 5: Re-Rank Selection (RRS), that applies confidence scores to choose a posteriori between the best hypothesis provided by the baseline models and the one provided by the re-ranking model, and the hypotheses selection criteria, that allow selecting hypotheses kept among the m-best to be re-ranked depending on an inconsistency metric.

Results obtained applying RRS strategy are shown in Table 6.13. These results show that the applied strategy always brings some improvements, more on Speech Input (.8% points in the best case, for attribute name&values extraction on the Italian test set) than Text Input (.4% points in the best case on MEDIA test set for attribute name&values extraction). Note that the improvements, in particular on MEDIA, even if small, are still significant since both baseline and re-ranking models are already very accurate. It is important to report that effective re-ranking models, using also the score provided by the re-ranked model, were studied in [14]. There is a big difference between our models and the approach proposed in [14]. In the latter, the score is used directly as feature in training phase, so that to learn correlation between features contained in the hypotheses and the corresponding probability provided by the baseline model. In our work we use this score in a post-processing step, combined also with re-ranker score, to learn thresholds and decide when it is more likely that the best hypothesis provided by the baseline model is more correct than the best hypothesis provided by the re-ranking model. We have run some experiments using also the approach described in [14]. The resulting models are more accurate than the "normal" re-ranking models, but significantly less than models resulting when applying our RRS approach.

	Text Input(CER)				**Speech Input(CER)**			
	MEDIA		LUNA-IT		MEDIA		LUNA-IT	
Model	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
FST	14.2%	17.0%	24.4%	27.4%	28.9%	33.6%	36.4%	39.9%
SVM	13.4%	15.9%	25.3%	27.1%	25.8%	29.7%	34.0%	36.7%
CRF	**11.7**%	**14.2**%	21.3%	**23.5**%	**24.3**%	**28.2**%	**31.0**%	**34.2**%
FST+RR	11.9%	14.6%	21.3%	23.7%	25.4%	29.9%	32.6%	36.2%
FST+RRS	**11.7**%	**14.2**%	**21.0**%	**23.5**%	25.0%	29.2%	31.8%	35.5%

Table 6.13: Results of SLU experiments on the MEDIA and the Italian LUNA test sets on manual and automatic transcriptions for both attribute names (Attr) and attribute values (Attr+Val) using also the Re-Rank Selection strategy for the re-ranking model (RRS)

Applying the new Hypotheses Selection Criterion (HSC) described in Chapter 5 brings further improvements on the re-ranking model performance. A very important point in using the HSC solution is that it allows to re-rank hypotheses generated by any kind of SLU model. Re-ranking models described so far were based only on the SFST model, i.e. a generative model. Models with different characteristics, e.g. CRF, a discriminative model, are less suitable for re-ranking since they learn a very skewed distribution fitting the training data. Hypotheses generated with these models are much less various than those that can be produced using a generative model. Since re-ranking needs a relatively high number of different hypotheses in order to learn all the meaningful features, a discriminative model cannot provide such variaty of hypotheses thus affecting the re-ranking performance. Applying the new hypotheses selection criterion, as explained in Section 5.2, enlarge remarkably the variety of hypotheses kept in the m-best list and brings in the list possibly more correct hypotheses. Thanks to this selection criterion, we can apply re-ranking also to discriminative models. The model we chose is CRF, which has shown very good performances in several SLU tasks [81, 38].

Results are depicted in Table 6.14 and they show that the new Hypotheses Selection Criterion brings on average larger improvements than what it can be reached with only RRS strategy. For example on the Italian test set, re-ranking FST hypotheses ("FST+HSC" in the table), we can gain 1.6% points on Speech Input for attribute name&values extraction (Attr+Val), while on MEDIA the best gain is 1.2% points, again on Speech Input and on "*Attr+Val*" extraction. Even larger gains can be achieved re-ranking CRF hypotheses ("CRF+HSC"), especially on Speech Input. For example, on the Italian test set, 2% points reduction is achieved on both attribute name (Attr) and

	Text Input(CER)				Speech Input(CER)			
	MEDIA		LUNA-IT		MEDIA		LUNA-IT	
Model	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
FST	14.2%	17.0%	24.4%	27.4%	28.9%	33.6%	36.4%	39.9%
SVM	13.4%	15.9%	25.3%	27.1%	25.8%	29.7%	34.0%	36.7%
CRF	11.7%	14.2%	21.3%	23.5%	24.3%	28.2%	31.0%	34.2%
FST+RR	11.9%	14.6%	21.3%	23.7%	25.4%	29.9%	32.6%	36.2%
FST+HSC	11.5%	14.0%	20.7%	22.8%	24.9%	28.7%	31.5%	34.6%
CRF+HSC	**11.2**%	**13.8**%	**19.9**%	**21.9**%	**22.9**%	**27.2**%	**29.0**%	**32.2**%

Table 6.14: Results of SLU experiments on the MEDIA and the Italian LUNA test sets on manual and automatic transcriptions for both attribute (Attr) names and attribute values (Attr+Val) using also an improved Hypotheses Selection Criterion (HSC)

attribute name&value extraction (Attr+Val), i.e. 6.5% and 5.8% relative improvement, respectively. Relative improvements yielded by CRF re-ranking on MEDIA are smaller, due to the fact that CRF baseline is already very accurate on such large corpus.

Combining RRS and HSC together provides further and significant improvements. This is someway intuitive since, RRS thresholds are trained on the development set re-ranking output, which contains more correct hypotheses when using also HSC. Results for these experiments are shown in Table 6.15, where FST and CRF Re-ranking using RRS and HSC have been shortened as "FST+*Imp.*" and "CRF+*Imp.*", standing for Improvements (over the re-ranking model).

The most significant improvements with respect to the baseline model are achieved with FST Re-ranking ("FST+*Imp.*"): 18.8% (from 17.0% to 13.8%) relative improvement on Text Input and attribute name&values extraction for MEDIA; 16.1% (from 33.6% to 28.2%) relative improvement on Speech Input and attribute name&values extraction again for MEDIA; 21.5% and 14.8% relative improvement on the same tasks for the Italian corpus.

Relative improvements using CRF Re-ranking ("CRF+*Imp.*") are smaller but, with absolute CER of 22.7% and 26.3% on Speech Input for the MEDIA corpus, we provide new State-Of-The-Art results for this task (compared with [38]).

6.3.6 Robustness with Respect to Re-ranked Model Baseline

The small improvements for the SFST model described in Section 3.3.1 have been implemented during the research work described in this document. This means that we have

	Text Input(CER)				**Speech Input(CER)**			
	MEDIA		LUNA-IT		MEDIA		LUNA-IT	
Model	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
FST	14.2%	17.0%	24.4%	27.4%	28.9%	33.6%	36.4%	39.9%
SVM	13.4%	15.9%	25.3%	27.1%	25.8%	29.7%	34.0%	36.7%
CRF	11.7%	14.2%	21.3%	23.5%	24.3%	28.2%	31.0%	34.2%
FST+RR	11.9%	14.6%	21.3%	23.7%	25.4%	29.9%	32.6%	36.2%
FST+*Imp.*	11.3%	13.8%	19.2%	21.5%	24.5%	28.2%	30.7%	34.0%
CRF+*Imp.*	**11.1**%	**13.2**%	**19.0**%	**21.1**%	**22.7**%	**26.3**%	**28.3**%	**31.4**%

Table 6.15: Results of SLU experiments on the MEDIA and the Italian LUNA test sets on manual and automatic transcriptions for both attribute (Attr) names and attribute values (Attr+Val) using an improved Hypotheses Selection Criterion together with the Re-Rank Selection strategy (*Imp.* stands for improvements for the re-ranking model)

been able to perform re-ranking experiments on two different version of the same baseline model. As consequence we had the opportunity to test another important feature of our joint models: robustness with respect to the re-ranked individual model baseline performance. We perfomed exactly the same set of re-ranking experiments using also the improved version of the SFST model. Results are shown in Table 6.16. The improved SFST model is named FST_{++} in this table, so the corresponding re-ranking model is FST_{++}+*Imp.*. Results are given together with those achieved with the "traditional" SFST model for comparison. It can be noticed that FST_{++} has a significant improvement over the previous SFST model, especially on the Italian corpus, with a relative improvement of 15.7% on Text Input and 6.8% on Speech Input (both on Attr+Val).

Comparing the re-ranking results obtained with the two SFST models, we can see that RRS combined with HSC provide a great robustness with respect to the SFST baseline, in the sense that final results are only slightly different from those previously discussed, despite the relatively high improvements achieved with the FST_{++} model. The best improvements are again on the Italian corpus (since it is smaller it leaves larger margin for improvements), with 4.1% relative improvement on Speech Input and for attribute name& values extraction. Improvements are much less significant on the larger MEDIA corpus.

The improved SFST with Re-ranking based on RRS and HSC, as well as a model of CRF re-ranking, have been applied also to the Polish corpus, introduced in Section 6.1. Results for these experiments are depicted in Table 6.17. As mentioned earlier while introducing the task, Polish is the most difficult language among those considered in

	Text Input(CER)				**Speech Input(CER)**			
	MEDIA		LUNA-IT		MEDIA		LUNA-IT	
Model	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
FST	14.2%	17.0%	24.4%	27.4%	28.9%	33.6%	36.4%	39.9%
FST_{++}	13.4%	16.2%	20.1%	23.1%	28.3%	32.9%	33.3%	37.2%
SVM	13.4%	15.9%	25.3%	27.1%	25.8%	29.7%	34.0%	36.7%
CRF	11.7%	14.2%	21.3%	23.5%	24.3%	28.2%	31.0%	34.2%
FST+RR	11.9%	14.6%	21.3%	23.7%	25.4%	29.9%	32.6%	36.2%
FST+*Imp.*	11.3%	13.8%	19.2%	21.5%	24.5%	28.2%	30.7%	34.0%
FST_{++}+*Imp.*	**11.1**%	13.3%	**18.3**%	**20.9**%	24.3%	27.8%	29.2%	32.6%
CRF+*Imp.*	**11.1**%	**13.2**%	19.0%	21.1%	**22.7**%	**26.3**%	**28.3**%	**31.4**%

Table 6.16: Results of SLU experiments on the MEDIA and the Italian LUNA test sets on manual and automatic transcriptions for both attribute (Attr) names and attribute values (Attr+Val) using an improved version of the FST model, FST_{++}, and the improvements for the re-ranking model described in Chapter 5 (*Imp.* stands for improvements for the re-ranking model)

our work. Additinal complexity is introduced by the task itself that, with roughly 200 concepts to be recognized, is by far the most difficult task.

These aspects affect models performances, especially on Speech Input where the WER of the ASR is the highest among the corpora used in our experiments (38.9%). As consequence, error rates on speech input for this task are very high, no model can go below 50%. On text input, the joint models based on discriminative re-ranking are again the most accurate, with a relative improvement of 11.1% on attribute name&values extraction using FST Re-ranking approach ("FST_{++}+*Imp.*").

The re-ranking model based on the new version of the SFST model, FST_{++}, and on the RRS and HSC strategies results in our best joint model using FST. As mentioned at the beginning of this chapter, the FST model is suitable to be used with ASR word lattices, since the latter can be easily encoded as SFST. Applying the re-ranking model to word lattices requires a slightly different processing procedure with respect to the one we adopt when processing ASR 1-Best. In the latter case, the ASR 1-Best is converted into a SFST representation and then processed as a normal text sentence, as explained in Section 3.3.1. When processing lattices the following steps are performed:

1. The lattice is converted from HTK[3] format to AT&T FSM format, using acoustic

[3]http://labrosa.ee.columbia.edu/doc/HTKBook21/node257.html

POLISH	**Text Input(CER)**		**Speech Input(CER)**	
Model	Attr	Attr+Val	Attr	Attr+Val
FST_{++}	21.9%	27.1%	57.9%	64.0%
SVM	27.3%	31.2%	58.1%	61.5%
CRF	21.9%	25.9%	56.7%	60.3%
FST_{++}+*Imp.*	**19.5**%	**24.1**%	56.5%	61.3%
CRF+*Imp.*	21.3%	24.5%	**55.7**%	**59.1**%

Table 6.17: Results of SLU experiments on the POLISH corpus on manual (Text Input) and automatic (Speech Input) transcriptions for both attribute names (Attr) and attribute values (Attr+Val) extraction (ASR WER is 38.9%)

LUNA-IT	**Speech 1-Best Input(CER)**		**Speech Lattice Input(CER)**	
Model	Attr	Attr+Val	Attr	Attr+Val
FST_{++}	33.3%	37.2%	36.6%	41.2%
FST_{++}+*Imp.*	29.2%	32.6%	32.6%	36.3%

Table 6.18: Results of SLU experiments on the Italian corpus using ASR 1-Best and ASR Lattices as input. The Oracle Error Rate over words of lattices is 22.8%

score as weight.

2. The lattice is rescored with a stochastic language model trained on the training set of the corpus (the language model scale factor is 11, optimized on the development set)

3. The lattice, that is representend as a SFST, is composed with SFSTs constituting the SLU model described in Section 3.3.1 (the scale factor of the stochastic conceptual language model is 4, optimized on the development set).

4. From the resulting SFST, up to 1.000 hypotheses are generated, evaluated with the semantic inconsistency metric, re-sorted with respect to the inconsistency and the m-best are kept for re-ranking.

The results obtained on ASR lattices are shown in Table 6.18, compared with results obtained using ASR 1-Best. The latter results are significantly better than results on lattices (between 3 and 4 percent points). This is someway surprisingly, since WER on ASR 1-best is worst than Oracle Error Rate of lattices (27% and 22.8% respectively). Nevertheless working on lattices introduces a lot of noise, the re-ranking approach based on SFST is affected by noise more than other approaches, as it can be seen comparing

FST and CRF re-ranking results. As a consequence, even applying HSC doesn't allow improving results obtained using ASR 1-best as input. Further studies of how to better constrain hypotheses generated from lattices are needed, the oracle error rate motivates this work. Some studies are in progress, but this is left as a future work.

6.3.7 Impact of the M-Best List Size

An important point that should be considered when using re-ranking models is the quality of hypotheses used for training. RRS is a post-processing step, so it doesn't impact on the training phase. The Hypotheses Selection Criteria affect training since allows to select different hypotheses, possibly more correct, that will be used to train the re-ranker. The best hypothesis in each pool generated from the same input sentence is selected to build pairs, so that to provide examples of meaningful features, useful to capture the most significant word-concept dependencies. Beyond this, no further constraint is used in training phase. The best hypothesis taken from each pool can contain errors, i.e. is not an oracle. This means that the re-ranking model is learned using also features coming from wrong concepts tagging.

In general, it is important to provide these features since, also at classification phase, the best hypothesis provided by the baseline model for a sentence can be not an oracle. Although, it is interesting to put additional constraints on the quality of hypotheses used for training and learning a re-ranking model with these settings. As a general approach, we can keep only those pool of hypotheses where the best has an error rate lower than a given threshold. As a particular case, we can keep pool of hypotheses only if the best is an oracle.

We have run some experiments using these settings. Since not for all sentences can be provided an oracle interpretation, and hypotheses generated from these sentences are not kept, the coverage in terms of words and features is worst, roughly 30% less than using all hypotheses. In order to overcome this problem, at least to augment the coverage if terms of tokens, we modified the "FEATURES" semantic structure (Figure 4.3) using also prefixes of words in the leaves. The resulting models, since less training instances are provided, are roughly 30% smaller in terms of support vectors, up to 7 times faster in training phase and up to 4 times faster in classification phase. This allows to enlarge the test set size, keeping more than 10-best hypotheses to be re-ranked in classification phase. We have tried some experiments with an exponentially encreasing number of hypotheses, i.e. 10, 16, 32 and 64.

Results for these experiments are reported in Table 6.19 for the Italian corpus and in

LUNA-IT	FST_{++}+*Imp.*				**CRF**+*Imp.*			
	Text Input		Speech Input		Text Input		Speech Input	
n-best size	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
10-best	18.5%	21.0%	29.0%	32.4%	19.8%	21.5%	28.7%	31.9%
16-best	18.1%	20.5%	28.7%	32.2%	20.1%	21.8%	28.7%	32.0%
32-best	18.2%	20.8%	28.8%	32.4%	20.2%	21.9%	**28.5**%	**31.9**%
64-best	**17.9**%	**20.3**%	28.7%	32.2%	20.4%	22.1%	28.8%	32.2%

Table 6.19: Results of SLU experiments on the Italian LUNA test set on manual and automatic transcriptions for both attribute (Attr) names and attribute values (Attr+Val) as a function of the m-best list size, from 10 to 64

Table 6.20 for the French MEDIA corpus. As it was expected given the worst coverage in terms of tokens, results on Text Input using 10-best hypotheses are slightly worst than those obtained using the same number of hypotheses in classification phase and keeping all the hypotheses for training phase. Nevertheless, using a FST Re-ranking model ("FST_{++}+*Imp.*") on the Italian corpus, as the test set size encreases the performance improves previous re-ranking models. Additionally, on Speech Input these model is more robust. Even when using 10-best hypotheses for classification, despite the worst coverage in training phase, the results are slightly better than those shown in Table 6.16, improving further when using more than 10-best hypotheses: the best result, taking both Text and Speech Input into account, is obtained using 64-best hypotheses, leading to an average absolute improvement of .5% points. The same model on MEDIA has a different behavior. On Text Input the performance of previous models is never reached, but this is achieved on Speech Input using 16-best hypotheses.

Different interpretations hold for the CRF based re-ranker ("CRF+*Imp.*"). Results are basically always the same, with very small variations. The only good point is that the same performance as previous models can be reached on Speech Input, but with a smaller and faster model. Results can be explained by the lack of coverage in terms of tokens, by which CRF re-ranking is more affected.

Beyond some good achievement yielded by these models, a very important point can be concluded from the results described in this section: important information for learning accurate re-ranking models are carried also by wrong hypotheses, which provide negative features allowing a better discrimination for positive ones.

All the best results discussed so far, taking all the experiments into account, are summarized in Table 6.21. These results show an impressive reduction of error rates achieved during our research work. From the first SFST model, reaching on Speech Input

MEDIA	FST_{++}+*Imp.*				**CRF**+*Imp.*			
	Text Input		Speech Input		Text Input		Speech Input	
n-best size	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
10-best	**11.5**%	13.5%	24.6%	28.1%	11.5%	13.4%	23.0%	26.4%
16-best	11.5%	13.6%	24.3%	27.9%	**11.5**%	**13.4**%	**22.7**%	**26.3**%
32-best	11.5%	13.7%	24.5%	28.3%	11.5%	13.5%	22.9%	26.3%
64-best	11.5%	13.7%	24.5%	28.4%	11.5%	13.5%	22.9%	26.4%

Table 6.20: Results of SLU experiments on the MEDIA test set on manual and automatic transcriptions for both attribute (Attr) names and attribute values (Attr+Val) as a function of the m-best list size, from 10 to 64

	Text Input(CER)				**Speech Input(CER)**			
	MEDIA		LUNA-IT		MEDIA		LUNA-IT	
Model	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
FST	14.2%	17.0%	24.4%	27.4%	28.9%	33.6%	36.4%	39.9%
FST_{++}	13.4%	16.2%	20.1%	23.1%	28.3%	32.9%	33.3%	37.2%
CRF	11.7%	14.2%	21.3%	23.5%	24.3%	28.2%	31.0%	34.2%
FST_{++}+*Imp.*	**11.1**%	13.3%	**17.9**%	**20.3**%	24.3%	27.8%	28.7%	32.2%
CRF+*Imp.*	**11.1**%	**13.2**%	19.0%	21.1%	**22.7**%	**26.3**%	**28.3**%	**31.4**%

Table 6.21: Overall improvement achieved during our research work, FST and CRF baselines are shown together with improvement of the FST baseline (FST_{++}) and results achieved with the best joint models, FST_{++}+*Imp.* and CRF+*Imp.*, taking all improvements into account.

28.9% 33.6% 36.4% 39.9% concept error rates on MEDIA and Italian corpora, respectively, to the best re-ranking models described in this work, with concept error rates 22.7% 26.3% 28.3% 31.4% on the same tasks, resulting in relative improvements of 21.5%, 21.7%, 22.3% and 21.3%.

In order to show how much results can be further improved, we provide Oracle Error Rates (OER) of our joint models, for both attribute name and attribute name&values extraction, on both Text and Speech input and for all corpora. The Oracle Error Rate in this case is computed over hypotheses, i.e. the best hypothesis is taken among the m-best used in classification. This is different from how oracle is usually computed, i.e. aligning a graph with the reference, as it was done for OER over words computed on ASR lattices in previous section.

In Table 6.22 oracle error rates are given on the Italian corpus, over increasing number

LUNA-IT	FST_{++}+*Imp.*				**CRF**+*Imp.*			
	Text Input		Speech Input		Text Input		Speech Input	
n-best size	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
10-best	11.5%	13.3%	21.5%	24.2%	13.7%	15.0%	21.0%	23.1%
16-best	10.7%	12.6%	20.6%	23.6%	12.9%	14.1%	20.2%	22.3%
32-best	9.7%	11.6%	19.0%	22.0%	11.8%	12.9%	18.7%	20.6%
64-best	9.0%	10.8%	18.3%	21.2%	10.7%	11.8%	17.9%	19.8%

Table 6.22: Oracle Error Rates (OER) over increasing n-best list size for the Italian test set on both Text and Speech input and for both attribute name (Attr) and attribute value extarction (Attr+Val)

MEDIA	FST_{++}+*Imp.*				**CRF**+*Imp.*			
	Text Input		Speech Input		Text Input		Speech Input	
n-best size	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val	Attr	Attr+Val
10-best	6.2%	7.9%	18.8%	21.7%	6.6%	8.1%	18.0%	20.6%
16-best	5.8%	7.4%	18.2%	21.0%	6.2%	7.6%	17.4%	20.0%
32-best	5.5%	7.1%	17.6%	20.3%	5.8%	7.2%	16.8%	19.3%
64-best	5.1%	6.5%	17.1%	19.7%	5.5%	6.8%	16.3%	18.7%

Table 6.23: Oracle Error Rates (OER) over increasing n-best list size for the French MEDIA test set on both Text and Speech input and for both attribute name (Attr) and attribute value extarction (Attr+Val)

of hypotheses. In the same way, we provide OER on MEDIA in Table 6.23. Oracles on the Polish corpus are depicted in Table 6.24, only using 10-best hypotheses for classification. Finally, SLU oracles for lattices of the Italian test set are reported in Table 6.25.

Despite the good results achieved, the oracles in these tables show that further large improvements can be achieved, a lot of work is still needed to reach satisfactory results. In particular, Table 6.25 shows that oracles on lattices are slightly worst than oracles on ASR 1-best, this means that more hypotheses should be generated in order to capture the most correct in the m-best list. Combining the ASR lattice word space with SLU model semantic space results in a huge SLU search space. Eventually 1.000 hypotheses are not sufficient to include the most correct ones. Another explenation is that a more restrictive semantic inconsistency metric should be defined on such noisy input.

POLISH	**Text Input(CER)**		**Speech Input(CER)**	
Model	Attr	Attr+Val	Attr	Attr+Val
FST Re-ranking	11.2%	15.3%	48.3%	52.3%
CRF Re-ranking	13.3%	15.9%	45.4%	47.4%

Table 6.24: Oracle Error Rates (OER) on the POLISH corpus test set on both Text and Speech input for attribute name (Attr) and attribute value (Attr+Val) extraction for the two re-ranking models described in this work, "*FST Re-ranking*" and "*CRF Re-ranking*"

LUNA-IT	**Speech 1-Best Input(CER)**		**Speech Lattice Input(CER)**	
n-best size	Attr	Attr+Val	Attr	Attr+Val
10-best	21.5%	24.2%	22.1%	24.9%
16-best	20.6%	23.6%	21.8%	24.6%
32-best	19.0%	22.0%	19.9%	22.7%
64-best	18.3%	21.2%	18.9%	21.4%

Table 6.25: Oracle Error Rates (OER) on the Italian test set using ASR 1-best (Speech 1-Best Input) and ASR lattice (Speech Lattice Input) speech input.

6.3.8 Comparison with State-Of-The-Art

As a final models comparison, needed to show where our work is placed with respect to the state-of-the-art, we have reported performances of all the best SLU models used in the last decade for SLU. This comparison is made on the MEDIA task since it is the only corpus on which all the reported models have been tested. In the comparison are included all the models described in Section 3.3, in particular we provide results for the Stochastic Finite State Transducer model (FST), for the SVM baseline model (SVM), for the CRF baseline model ($CRF_{baseline}$) used in our joint models based on discriminative re-ranking (FST Re-ranking and CRF Re-ranking). Additionally we have taken results from [38] for the best CRF model (CRF_{SOA}) on MEDIA in the literature, for the Maximum Entropy model (MEMM, called "log-lin" in [38]), for the Statistical Machine Translation (SMT) model and for the models combination approach based on ROVER (defined in [34]). Finally we have taken results from [55] for the Dynamic Bayesian Network model (DBN). All these results are listed in Table 6.26.

These results show that our joint models, in particular CRF Re-ranking, result to be the best models for SLU on this task. We are aware that several results among those depicted in Table 6.26 have been significantly improved. Nevertheless, most results shown in this work are still state-of-the-art.

MEDIA	Text Input		Speech Input	
	Att	Att+Val	Att	Att+Val
CRF_{SOA}	11.8	16.2	24.6	29.8
$CRF_{baseline}$	11.7	14.2	24.3	28.2
SVM	13.4	15.9	25.8	29.7
MEMM	15.0	19.3	27.8	33.5
FST	13.4	16.2	28.3	32.9
SMT	19.2	23.3	29.2	35.2
DBN	15.5	17.4	29.1	32.8
weighted ROVER	**11.0**	15.0	23.8	28.9
FST Re-ranking	11.1	13.3	24.3	27.8
CRF Re-ranking	11.1	**13.2**	**22.7**	**26.3**

Table 6.26: Comparison of results achieved with the best joint models described in this work and the best State-Of-The-Art models used in the last decade for SLU. The comparison is made on MEDIA, since results are available for all models only on this corpus. For CRF are shown both the best result achieved in the literature (CRF_{SOA}) and our baseline $CRF_{baseline}$ obtained using the tool CRF++ (http://crfpp.sourceforge.net/)

6.4 Open Issues

There are two open issues that we have not been able to exaustively investigate during our research work:

- Semantic Composition
- Context Sensitive Validation

The first deals with the composition of semantic constituents annotated by the SLU model to build more complex semantic structures. In the LUNA project the only annotated semantic structures are frames like in the FrameNet corpus [3]. This impose using frames for the semantic composition module. Nevertheless, as we stated in Section 3.2, semantic roles are instantiated by words realizing syntactic constituents, as well as semantic constituents used for SLU tasks. In other words semantic roles are different names for the same information. As consequence, we don't expect to achieve improvements using an integrated representation based on both semantic roles and semantic constituents coming from a specific SLU task. It is more intuitive instead, using semantic roles annotation as an additional features in the re-ranking models. This can be done by simply using a

tree representation of frames like the one in Figure 4.3, together with our semantic trees. Each instance for the re-ranking model is thus made of four different structures, two for each hypothesis: the semantic tree used in all previous models plus the tree representation of the frames instantiated in the corresponding sentence. With such instances the generalized form of the re-ranking kernel is used, as explained in Section 4.4.

Since frames are information at a turn level, they can bring in the model explicit information of how to distinguish between different types of turns, like for example dialog acts. We have performed some experiments using these settings and we have found out that the performance is exactly the same of our best re-ranking models, i.e. using frame information doesn't provide additional information useful to improve the SLU task. Our a priori intuition still holds, but most probably, on a simple task like the one modeled by the Italian corpus, the model can learn implicitly differences between different kind of turns, without the need of additional information.

Given these findings, semantic composition in the Italian corpus is yielded by merging a posteriori the attribute value annotation produced by our SLU model with the frame information produced by the system described in [22], that has been implemented for the same Italian task.

Additionally we have tried some more experiments towards the intergration of semantic and syntactic features. We have followed the same approach used in many cases in the Semantic Role Labeling task, i.e. we have annotated syntactic parse trees with semantic constituents taken from the SLU model output. The annotation is performed percolating semantic tags from pre-terminal nodes to the node covering the entire surface form instantiating the concept. For example, using the same input sentence of Section 3.2:

> *"Buongiorno ho un problema con la stampante da stamattina non riesco piu' a stamapare"*

("*Good morning I have a problem with my printer since this morning I cannot print any more*") resulting in the following SLU interpretation:

> **null**{Buongiorno ho} **ProblemHardware.type**{un problema} **Peripheral.type**{con la stampante} **Time.relative**{da stamattina} **HardwareOperation.negate**{non} **null**{riesco piu' a} **HardwareOperation.operationType**{stampare}

the structure integrating syntactic and semantic features for this SLU interpretation is shown in figure 6.3. As for structures shown in Section 4.3, several different variants of this structure have been tested, e.g. adding also values extracted from surface forms, BIO-like

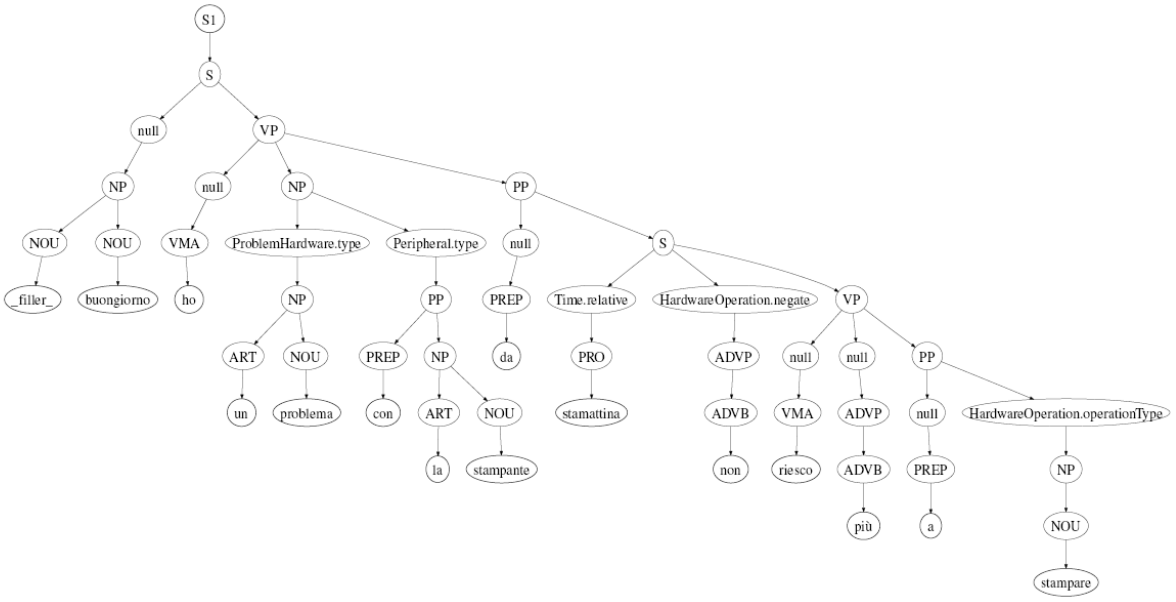


Figure 6.3: An example of structure intergrating syntactic and semantic features taken from the LUNA Italian corpus

markers etc. Even if they are interesting from an estetical point of view, the use of this structure didn't bring any improvement to the re-ranking models described in previous sections. This is due to the fact that the structure shown here are much more complex than those used previously. The higher level of generalization brought by POS tags and syntactic constituents doesn't provide additinal information, in contrast it can add ambiguity. Features capturing relevant information contain longest paths and in general bigger substructures, thus they are more sparse and more difficult to match in noisy input like speech transcriptions. Simpler structures like those shown in Section 4.3 provides more compact information about words-concepts dependiencies, where the surface form is the most important level and sufficient generalization is yielded by the simpler categories used in the SLU models. Further studies can be conducted applying sophisticated features selection techniques, e.g. [18], or reverse features engineering, like in [75].

Regarding Context Sensitive Validation (CSV), this has been integrated also in the system described in [22]. Since CSV is performed at dialog level, the context to validate the current interpretation must be provided by the dialog manager, this go beyond the purpose of this dissertation, where we have focused mainly on SLU models. Nevertheless is worth discussing some ideas that can be studied in future works.

Dialog context can be taken into account in SLU models exploiting again the generalized re-ranking kernel. The context in this case would be the interpretation of the

previous dialog turn, added as feature for the re-ranking model, like it has been done with frame information explained above. The training for such model is carried out in one step using two consecutive turns for each instance. Classification phase is performed in two steps using two different re-ranking models: the first is the "normal" classification step performed for all the experiments described in this chapter. The second is carried out with the model trained taking dialog context into account and using the best interpretation from the first step as context. Note that this approach can be generalized using more than one dialog turn as context, additionally also more structures for each turn can be used, the generalized re-ranking kernel can take many structures into account. Nevertheless, a trade-off between accuracy and computational cost should be studied as well.

Chapter 7

SLU in Spoken Dialog Systems

In this chapter we sescribe the Spoken Dialog System (SDS) we have developed during our research work. The system integrates the joint models for SLU based on discriminative re-ranking described in previous chapter. Our SDS is an evolution of a call routing application and it is based on the Italian corpus acquired in the European project LUNA: problem solving in the domain of hardware/software repairing (see Section 6.1).

The goal of a call routing application is to determine the type of the call from the user utterance and transfer the call to an appropriate destination. Even though at first glance it may seem that a dialog is not really necessary to accomplish this, it is an important part of such an application since it might not be possible to correctly guess the type of the call at the first turn. This might happen due to ambiguity of input or failure of the system to understand the user correctly [35]. Call routing has been investigated e.g. in [35, 13, 32]. Most of the work in call routing has focused on classifiers and their performance. None of the works in the field report evaluation of the effect of the dialog on this task. In this work we took also these aspects into account.

A call routing application, a Help Desk for hardware and software-related problems, was developed as the Italian Spoken Dialogue System Prototype for the LUNA Project. The goal of the system is to identify the type of the user problem as one belonging to one of the 10 possible scenarios, identifying problem classes. Upon successful completion of the dialogue, the call is forwarded to the appropriate human operator able to provide further assistance. The system also summarizes call-type and all the relevant information acquired during the dialog, e.g. acquiring the brand of hardware, turns out to be useful to give a positive feedback to the users on the successful understanding of the kind of hardware involved in the problem. An example dialogue (translated in English) can be seen on Figure 7.

SYSTEM: Welcome to LUNA. Good day, I am Paola. How may I help you?
USER: Eh, Sorry. I have a problem with the printer.

ASR	And I am sorry a problem with the printer		
SLU	**Concept**	**Value**	**Conf**
	conjugation	and	0.725
	problem	a_problem	0.731
	computer_componentHardware	with_the_printer	0.718
CTC	Label: C1_Printer_Problem; Confidence: 1		
DM	1 Infer subclass_C1_Printer_Problem		
	2 Inferred Class == CTC Label		
	3 CLASS LABEL AND PROBLEM STATEMENT		
	4 MOVE TO NEXT TURN		

SYSTEM: Can you please tell me what's the brand of your printer ?
... ...
SYSTEM: Thank you, wait in line.
An operator will assist you with your Lexmark printer problem!

Figure 7.1: Example dialogue translated to English

7.1 Dialogue System Architecture

A typical interaction with the system is initiated by a phone call that arrives at a telephony server. This routes the call to a VXML platform. Since the VXML standard is based on the web infrastructure, a VXML platform can issue HTTP requests that can be served by a web server just like any HTML page. This allows us to organize the processing modules of the dialogue system (SLU, DM, VXML generator) as web services that are invoked by the HTTP request. As a consequence, each system turn of a dialogue is a separate, *stateless* request. State information is passed through the dialog exploiting usual mechanisms of web applications, i.e. web sessions. Even more, all the information used in any system module are stored in a database, making it persistent the dialog state.

The architecture implements a 'fat pipeline' paradigm: each speech, language and DM module has access to the database for rescoring and modeling (e.g. time series intra and inter dialogues) and for retrieving at any moment all the dialog state information. At the implementation level, this balances a lightweight communication protocol downstream with data flowing laterally towards and from the database. Further details about dialog state persistency are described in [95].

The overall architecture of the system is shown in Figure 7.2. The architecture is based on the Loquendo *VoxN@uta* Platform[1]. We adopted state-of-the-art ASR and TTS technologies provided by the Italian Loquendo partner of the LUNA project[2]. Since we used our joint models described in previous chapters, also the SLU module is state-of-the-

[1]http://www.loquendo.com/en/technology/voxnauta_platform.htm
[2]http://www.loquendo.com/en/

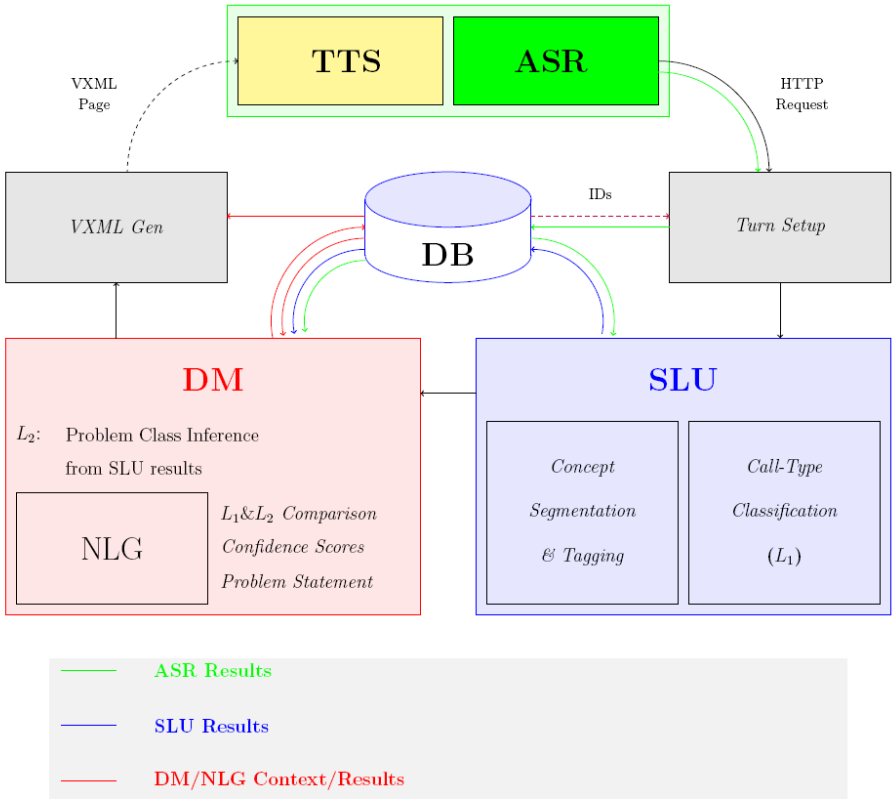


Figure 7.2: A general diagram of re-ranking framework showing the entire chain of processing, from speech input to the SLU interpretation

art. While for the other components we adopted tradictional solutions: a simple sentence classifier for call-type classification; a rule-based dialog manager, integrating also the language generation module, implemented as an expert system in CLIPS[3].

7.2 Spoken Language Understanding

The SLU Module of the SDS consists of two components operating in parallel. Concept tagging is complemented with user call-type classification, and the results of both components are passed to DM to decide the next move. Automatic concept tagging is performed exploiting the joint model based on discriminative re-ranking described in previous chapters.

[3]http://clipsrules.sourceforge.net/

7.2.1 Call-Type Classification

Since the goal of the system is to classify the call as belonging to one of the ten possible scenarios, SLU concept attribute values and segmentation are also complemented with user goal prediction. These user goals are extracted from the caller responses to the opening prompt. A BoosTexter [84] based utterance classifier is build to operate on ASR hypotheses in parallel to SLU concept segmentation. The class label provided by the classifier and its confidence are used further for the decision by the DM.

The call-type classifier was trained on first utterances of the LUNA Italian Wizard of Oz corpus. The first turn of each dialog in this corpus is the one where, in most cases, the user states clearly the problem. So it is the most meaningful to discriminate between the different problem classes. In the other turns few additional information are provided to the system, unless the user is asked to state again the problem. Due to the nature of the task - the classification of documents in the same domain - the most distinguishing features are hardware types such as keyboard, mouse, etc.; thus, unigram model had comparable performance to the bigram and trigram feature models. The best performing unigram model, having 93.8% accuracy on the test set, was chosen to be used for call-type classification within the spoken dialogue system prototype. The above performance is measured on manual transcriptions, since the model has been trained to be used in the SDS, evaluation on ASR output is a more reliable measure of the model accuracy, which turned out to be 90.8%.

The problem class identifier provided by the call-type classifier is used together with automatic concept annotation provided by the SLU joint model to decide next dialog move. With reference to Figure 7, as it can be seen the user states a printer problem. The statement is supported by an explicit proble statement ("*a_problem*") and by a "`hardware-component`" concept realized by the surface form "*with_the_printer*". For the same turn the classifier predict a "*Printer-Problem*". Since in this particular case, the SLU model and the call-type classifier agree on the same problem class, the system accept the prediction and move to the next dialog step.

7.3 Dialogue Management

Dialogue management follows an Information State Update approach [53]. First, the following information is retrieved from the database:

- ASR recognition results of last user turn;
- confidence and other thresholds;

- SLU concept attribute-value pairs;
- classifier results for the problem class if available (first turn only), which include problem class label and confidence of this label, as mentioned in section 7.2.1;
- all open questions for the current dialogue from the database;
- application information already provided by the user, including their grounding status ('explicitly-verified' by a clarification question, 'implicitly-accepted' by virtue of being above a heuristically set threshold, and 'under-verification' for ongoing clarification questions);
- user rejections in verification questions and 'noinput' / 'nomatch' events of the entire dialogue;
- 'noinput' and other events are summarized to produce counts that can serve as 'success metrics' of the ongoing dialogue.

Given this information, the DM employs a 'dialogue move engine' to determine the system action and response. This is accomplished using several sets of forward chaining inference rules: SLU rules match the user utterance to open questions. This may result in the decision to verify the application parameter in question (e.g. problem class, hardware brand), and the action is verbalized by language generation rules (which are part of the DM in this system). If the parameter is accepted, application dependent task rules determine the next parameter to be acquired, resulting in the generation of an appropriate request. Typical dialogue moves available to the system are those that are needed for the application domain, for example forward looking moves such as 'question-parameter', to acquire a new parameter, 'confirm-parameter', for parameter confirmation, and 'request-repeat', to ask again a paramter in case 'question-parameter' failure, and backward looking moves such as 'accept-parameter-implicitly' (by the system) or 'answer-question-parameter' (by the user). The dialogue is initially open to a wide range of user utterances in response to "How may I help you?" prompt and becomes more constrained afterwards.

7.4 Experiments and Results

The system was tested by 50 volunteer callers (3 calls each), and the collected calls were transcribed and annotated. We have selected a 100 dialogues subset (only dialogues containing all required metrics are included) that was used in assessing the system performance.

Metric	All	T1	T2	T3
Av. Dial. Dur. (sec)	40.29	36.54	42.93	45.90
Av. Turn. Dur. (sec)	7.90	8.08	7.87	7.40
Av. # of Turns	5.10	4.52	5.45	6.20
Av. # of Tasks	1	1	1	1

Table 7.1: General dialogue and utterance level metrics

7.4.1 General Dialogue Statistics

The average duration of the 100 dialogues is 40.29 seconds, with 5.10 turns per dialogue in average. In the scenarios covered by the prototype there is 1 task per dialogue by design (excluding transfer to the operator request, which was never encountered).

The dialogues were categorized with respect to the way the dialogue was ended:

- T1 – the call was routed with correct attribute values
- T2 – the call was routed with incorrect attribute values
- T3 – the call was transferred to the operator

As can be seen in Table 7.1, successful dialogues (T1) have the least number of turns on average and the shortest dialogue duration. Dialogues in T3 category are the opposite situation, they have the longest average duration and contain the highest average number of turns. The average turn duration exhibits the reverse tendency compared to average dialogue duration and number of turns. This observation is to be expected since successful dialogues contain fewer turns with more interaction. The general dialogue statistics confirm that the length of the dialogue is indicative of its success.

7.4.2 Task Success

The task is considered completed successfully in case it was routed to the appropriate destination, i.e. both problem class and the hardware brand attribute should be correct. However, from the point of view of the user, on a longer term, the task can be considered successful also in case of transfer to the operator. The task success values presented are given for both cases: T – success only refers to call routing with correct attributes; T* – success also includes transfer to the operator.

[95] measure task success as the ratio of completed tasks to tasks requested, such that the completed task is the requested task. Tradictional Precision, Recall and F-Measure on the other hand (P, R, $F1$ respectively), allow to measure task success in a way that

	P	R	F1	TSR
T	0.52	0.46	0.49	0.47
T*	0.57	0.56	0.56	0.57

Table 7.2: Task Success as Precision (P), Recall (R) and F-Measure (F1); and Task Success Rate (TSR)

also takes into account mismatches between requested and completed task types. The resulting overall task success in terms of P, R, & $F1$ and Task Success Rate is presented in Table 7.2.

Regarding task failure, in the majority of cases (28 out of 43, i.e. 65.12%; not shown in Table 7.2), a task was not completed successfully (call routings with wrong hardware brand attribute) because the system received a brand name that was not covered by the grammar. Misrecognition was not among the main reasons contributing to the 'failed' task category in case of routing with incorrect attributes or in case of transfer to the operator.

Overall, the task success could be greatly improved by increasing the coverage of the grammars used in the system. However, better ASR is not the only factor affecting successful call routing; classifier performance and dialogue also play an important role. In Section 7.4.4 we present an evaluation of the classifier in detecting call type as well as the performance increase we gain from dialogue.

7.4.3 Task Success as Perceived by the User

The user also responded to a questionnaire following the dialogue. One of the questions of this questionnaire, "How well do you think the system understood your problem", was intended to measure the caller subjective perception of the task success. The callers scored the system on a 1-5 Likert scale. The results of the mapping of these subjective evaluation scores to the three task completion types described in Section 7.4.1 are presented in Table 7.3. As can be seen, call routing with the correct attribute value was judged as better understanding (average score 4.44) compared to the other two task completion types (T2 – 3.20 and T3 – 1.70). Moreover, there is a significant correlation between user judgements and task completion types: Spearman $r(98) = 0.56, p < 0.05$.

7.4.4 Call-type Classification vs. Dialogue

Since our system performs call-type classification on ASR interpretations of user utterances, we can assess the performance gained by implementing the whole dialogue system

Score	T3	T2	T1	Total
5	0.10 (1)	0.27 (12)	0.63 (29)	0.42 (42)
4	0.00 (0)	0.11 (5)	0.22 (10)	0.15 (15)
3	0.10 (1)	0.30 (13)	0.11 (5)	0.19 (19)
2	0.10 (1)	0.18 (8)	0.04 (2)	0.11 (11)
1	0.70 (7)	0.14 (6)	0.00 (0)	0.13 (13)
# of Scores	0.10 (10)	0.44 (44)	0.46 (46)	1.00 (100)
Av. Score	**1.70**	**3.20**	**4.44**	**3.62**

Table 7.3: Subjective task success assessment normalized by task completion type (counts given in parentheses)

Problem Class	Classifier	Dialogue
C1 Printer	0.89	0.94
C5 Keyboard	0.67	1.00
C9 CD-ROM	0.94	1.00
...	**...**	**...**
Accuracy (all classes)	0.79	0.94
Weighted Av. Acc.	0.79	0.94

Table 7.4: Classifier vs. Dialogue performance (accuracy) on determining the problem class

pipeline versus just the call-type classifier. Table 7.4 provides the call-type classification accuracy for classifier and the dialogue system as a whole. The values for dialogue contain cases for all dialogue completion types in which the system was able to determine the problem class correctly. As can be seen from the table, there is an important improvement in performance on call-type classification when dialogue is used. Even though only 100 dialogues were used, the results are statistically significant: $\chi^2(1, N = 200) = 9.63, p < 0.05$.

To assess the performance we gain by using dialogue even further, we measured the oracle accuracy of the classifier. Dialogue performance on detecting call category (0.94) turned out to be higher than 6-best oracle accuracy of the classifier (0.93). This means that even an oracle that always chooses correctly from the 6 best hypotheses would not be better than our dialogue system.

Chapter 8

Conclusions

In this document we have described the study, the implementation and the evaluation of several models for Spoken Language Understanding. The problem has been modeled as semantic parsing, going from utterance transcriptions, both manual, using annotated corpora, and automatic, generated by Automatic Speech Recognizers, to semantic structures, based either on flat attrivute value representation or on semantic trees, designed for the specific tasks described in this work.

We proposed a solution integrating SLU models with different characteristics, i.e. Stochastic Finite State Transducers and Conditional Random Fields, via discriminative re-ranking of SLU hypotheses, based on Support Vector Machines and sophisticated kernel methods designed for Natural Language Processing tasks. We have integrated in our joint models two different strategies to select more correct hypotheses: Re-Rank Selection and a new Hypotheses Selection Criterion, which provide significant improvements.

The proposed solution results to be very effective, with respect to both indivudual SLU models and another models combination approach used in SLU based on ROVER. A comprehensive evaluation has been performed on four different corpora in four different languages, representing the most significant tasks for Spoken Language Understanding. Results show that our solution outperforms in most cases the best state-of-the-art models on the same tasks.

Additionally, we have performed an evaluation on the Italian task using ASR word lattices as input. The increased noise in the search space resulting from the combination of lattices with an SLU model, prevents the joint models to be effective, showing that additional information is needed to better assess SLU hypotheses kept for re-ranking.

The main characteristics of our joint models are the ability to put together characteristics of the combined models, representing the meaningful information in complex

structures which allow to capture arbitrary long-distance dependencies between words and concepts.

We have integrated the proposed models in a Spoken Dialog System in Italian for the domain of hardware/software problem solving. We also presented evaluation results of the system on task success, comparing objective and subjective assessments of the dialogs. We quantified the effect of dialogue on call routing and found a significant improvement compared to routing based on call classification alone.

There are several interesting aspects that should be addressed in order to give a better understanding and possibly a better model performance on the re-ranking approach:

- Dependency of performance from ASR Word Error Rate (WER). WER can be varieted using different scale factors for the language model and different insertion penalties in the ASR system.
- Dependency of performance from Lattices Oracle Error Rate.
- Dependency of performance from kernel structures: many structures have been designed and evaluated. Nevertheless there are further structures that can better capture meaningful features for the SLU tasks, e.g. integrated syntactic semantic structures, on which sophisticated features selection and features engineering tequiniques are needed.
- Dependency of performance on the Re-rank Selection training approach: thresholds training is very fast and relatively effective, but in some cases training a new linear classifier may result in more accurate final SLU result.

Oracle Error Rates of SLU models described in this work underline that a lot of work is still needed in order to reach satisfactory results in an absolute scale.

Chapter 9

Acknowledgements

Reaching the end of this Ph.D. required all my energy and all my motivation during more than three years. Even if at first I was completely lost, finally I succeded, I found a research line that is very complex but very interesting.

Although it is the completion of a very important step, completing a Ph.D. doesn't mean reaching the end, is just the opposite: it is the training before starting something else, scientific research in most cases. This is what I think to have learned and what I would like to do in the future.

There are many people I should thank for having reached this result. First my advisor, he always believed in me and gave me the opportunity to get "the highest study degree". I thank also my "unofficial co-advisor", Alessandro Moschitti, he helped me a lot to improve my technical skills.

A very special thank to my girlfriend, Anne, who has been always close to me, even more during the most difficult periods of my Ph.D. If we can compare the Ph.D. to a journey across the ocean, if we can compare difficulties and problems involved in a Ph.D. to a storm, then my girlfriend has been the lighthouse that guided me to reach the destination, without getting lost. Most of the energy and motivation I needed to complete successfully this experience came from our love, from our common goal to live together the rest of our lives.

Another special thank to all my family, especially my parents, they always gave me support. Since they don't speak English, I write the acknowledgement for them in Italian to make them understand:

"*Un ringraziamento speciale va ai miei genitori, che oltre ad avermi aiutato economi-*

*camente durante gli studi, mi son sempre stati vicini anche durante il dottorato. Tanta gratitudine va a mia madre, che è sempre stata per me come un' amica. Sono molto grato anche a mio padre, che, tra le altre cose, seppur involontariamente, mi ha dato l' ispirazione: Mio padre è un ottimo giardiniere, ed uno dei tanti giorni in cui sono andato con lui per aiutarlo, durante la mia adolescenza, mi disse: <<**è molto importante saper lavorare con gli alberi**>>. Aveva perfettamente ragione.*"

I want to thank also the people in my group in the LSI lab and in the ADAMACH project, many of them have been more than colleagues, they have been good friends. In particular Silvia Quarteroni and Alexei Ivanov.

The list of people to thank is very long, I cannot thank them all one to one, so I thank everybody I met during these 3.5 years of Ph.D.

Thank you everybody!

"Non importa quante anse ha un fiume, alla fine di un giusto corso ogni acqua trova il suo mare"

Bibliography

[1] Giuseppe Riccardi Alexandros Potamianos, Shrikanth Narayanan. Adaptive categorical understanding for spoken dialog systems. *IEEE Transactions on Speech and Audio*, 13(3):321–329, 2005.

[2] James F. Allen. *Natural Language Processing*. John Wiley and Sons Ltd. Chichester, UK, 2003.

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of COLING-ACL '98*, pages 86–90, 1998.

[4] Oliver Bender, Klaus Macherey, Franz-Josef Och, and Hermann Ney. Comparison of alignment templates and maximum entropy models for natural language understanding. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 11–18, Budapest, Hungary, April 2003.

[5] Steven Bethard, Zhiyong Lu, James H. Martin, and Lawrence Hunter. Semantic role labeling for protein transport predicates hacioglu. In *In HLT/NAACL-03*, 2003.

[6] Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 4–6, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[7] Enrico Bocchieri, Giuseppe Riccardi, and Jayanth Anantharaman. The 1994 att atis chronus recognizer, 1994.

[8] Hélène Bonneau-Maynard, Christelle Ayache, F. Bechet, A Denis, A Kuhn, Fabrice Lefèvre, D. Mostefa, M. Qugnard, S. Rosset, and J. Servan, S. Vilaneau. Results of the french evalda-media evaluation campaign for literal understanding. In *LREC*, pages 2054–2059, Genoa, Italy, May 2006.

[9] Hélène Bonneau-Maynard, Sophie Rosset, Christelle Ayache, Anne Kuhn, and Djamel Mostefa. Semantic annotation of the french MEDIA dialog corpus. In *ISCA Eurospeech*, Lisboa, Portugal, 2005.

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

[11] X. Carreras and Lluis Marquez. Introduction to the conll-2005 shared task: Semantic role labeling, 2005.

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

[13] J. Chu-Carroll and B. Carpenter. Dialogue management in vector-based call routing. In *Proc. of the Annual Meeting of the ACL*, Montreal, 1998.

[14] M. Collins and N. Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete structures, and the voted perceptron. In *Proceedings of the Association for Computational Linguistics*, pages 263–270, 2002.

[15] Michael Collins. Discriminative reranking for natural language parsing. In *ICML*, pages 175–182, 2000.

[16] Michael Collins. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the Association for Computational Linguistics*, pages 489–496, 2002.

[17] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press, 2001.

[18] Michael Collins and Terry Koo. Discriminative re-ranking for natural language parsing. *Computational Linguistic (CL)*, 31(1).

[19] Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. In *Computational Linguistics*, pages 175–182. Morgan Kaufmann, 2003.

[20] CoNLL-2000. Results of the CoNLL-2000 Shared Task on Chunking. http://www.cnts.ua.ac.be/conll2000/chunking/.

[21] Bonaventura Coppola, Alessandro Moschitti, and Daniele Pighin. Generalized framework for syntax-based relation mining. In *ICDM*, pages 153–162. IEEE Computer Society, 2008.

[22] Bonaventura Coppola, Alessandro Moschitti, and Giuseppe Riccardi. Shallow semantic parsing for spoken language understanding. In *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 85–88, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[23] Bonaventura Coppola, Alessandro Moschitti, and Sara Tonelli Giuseppe Riccardi. Automatic framenet-based annotation of conversational speech. In *Proceedins of the IEEE Workshop on Human Language Technology*, 2008.

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

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

[26] D.A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. Expanding the scope of the atis task: the atis-3 corpus. In *Proceedings of Human Language Technologies*, page 4348, 1994.

[27] Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. Concept segmentation and labeling for conversational speech. In *Interspeech*, Brighton, U.K., 2009.

[28] Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. Re-ranking models based on small training data for spoken language understanding. In *Conference of Empirical Methods for Natural Language Processing*, pages 11–18, Singapore,Singapore, August 2009.

[29] Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. Re-ranking models for spoken language understanding. In *Conference of the European Chapter of the Association of Computational Linguistics*, pages 202–210, Athens,Greece, April 2009.

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

[31] Marco Dinarelli, Evgeny Stepanov, Sebastian Varges, and Giuseppe Riccardi. The luna spoken dialog system: Beyond utterance classification. In *International Conference on Acoustic, Speech and Signal Processing*, Dallas, Texeas, U.S.A., 2010.

[32] K. Evanini, D. Suenderman, and R. Pieraccini. Call classification for automated troubleshooting on large corpora. In *Proc. of ASRU 2007*, Kyoto, Japan, 2007.

[33] Charles J. Fillmore. The case for case. *Universals in Linguistic Theory*, 88(1), 1968.

[34] J. G. Fiscus. A post-processing system to yield reduced word error rates: Recogniser output voting error reduction (ROVER). In *Proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 347–352, Santa Barbara, CA, December 1997.

[35] A. L. Gorin, G. Riccardi, and J. H. Wright. How may i help you? *Speech Commun.*, 23(1-2):113–127, 1997.

[36] Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tur, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. The att spoken language understanding system. *IEEE Transaction on Audio, Speech and Language Processing*, 14(1):213–222, 2006.

[37] Stefan Hahn, Patrick Lehnen, Georg Heigold, and Hermann Ney. Optimizing crfs for slu tasks in various languages using modified training criteria. In *Interspeech*, Brighton, U.K., 2009.

[38] Stefan Hahn, Patrick Lehnen, and Hermann Ney. System combination for spoken language understanding. In *Interspeech*, pages 236–239, Brisbane, Australia, 2008.

[39] Stefan Hahn, Patrick Lehnen, Christian Raymond, and Hermann Ney. A comparison of various methods for concept tagging for spoken language understanding. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco, May 2008.

[40] Timothy J. Hazen, Theresa Burianek, Joseph Polifroni, and Stephanie Seneff. Recognition confidence scoring for use in speech understanding systems. In *Computer Speech and Language*, pages 49–67, 2000.

[41] Y. He and S. Young. Semantic processing using the hidden vector state model. *Computer Speech and Language*, 19:85–106, 2005.

[42] Gang Ji and Jeff Bilmes. Backoff model training using partially observed data: Application to dialog act tagging. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 280–287, New York City, USA, June 2006. Association for Computational Linguistics.

[43] Richard Johansson and Pierre Nugues. Dependency-based semantic role labeling of PropBank. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 69–78, 2008.

[44] Richard Johansson and Pierre Nugues. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom, 2008.

[45] Richard Johansson and Pierre Nugues. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 393–400, Manchester, United Kingdom, August 18-22 2008.

[46] Sameer Pradhan Kadri, Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. Semantic role parsing: Adding semantic structure to unstructured text. In *In ICDM*, pages 629–632, 2003.

[47] Taku Kudo. Crf++ toolkit. http://crfpp.sourceforge.net/, 2005.

[48] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

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

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

[51] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 230–237, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[52] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289, Williamstown, MA, USA, June 2001.

[53] S. Larsson and D. Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6(3-4):323–340, 2000.

[54] Fabrice Lefèvre. A dbn-based multi-level stochastic spoken language understanding system. In *IEEE Spoken Language Technology Workshop*, pages 82–85, December 2006.

[55] Fabrice Lefèvre. Dynamic bayesian networks and discriminative classifiers for multi-stage semantic interpretation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 4, pages 13–16, Honolulu, HI, USA, April 2007.

[56] Patrick Lehnen, Stefan Hahn, Hermann Ney, and Agnieszka Mykowiecka. Large-scale polish slu. In *Interspeech*, 2009.

[57] E. Levin and R. Pieraccini. Concept-based spontaneous speech understanding system. In *4th European Conf. on Speech Communication and Technology (EUROSPEECH)*, volume 2, pages 555–558, Madrid, Spain, September 1995.

[58] Klaus Macherey, Oliver Bender, and Hermann Ney. Applications of statistical machine translation approaches to spoken language understanding. *Computer Speech and Language*, 17(4):803–818, 2009.

[59] Krzysztof Marasek and Ryszard Gubrynowicz. Design and Data Collection for Spoken Polish Dialogs Database. In *Proc. of the Sixth Int. Conf. on Language Resources and Evaluation (LREC)*, Marrakech, Morocco, May 2008.

[60] Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. Semantic role labeling: an introduction to the special issue. *Comput. Linguist.*, 34(2):145–159, 2008.

[61] Arne Mauser, Richard Zens, Evgeny Matusov, Saša Hasan, and Hermann Ney. The RWTH statistical machine translation system for the IWSLT 2006 evaluation. In *International Workshop on Spoken Language Translation*, pages 103–110, Kyoto, Japan, November 2006. Best Paper Award.

[62] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[63] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer, Speech and Language*, 16(1):69–88, 2002.

[64] Renato De Mori. *Spoken Dialogues with Computers*. Academic Press, Inc., Orlando, FL, USA, 1997.

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

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

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

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

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

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

[71] Agnieszka Mykowiecka, Krzysztof Marasek, Magorzata Marciniak, Joanna Rabiega-Winiewska, and Ryszard Gubrynowicz. Annotated corpus of Polish spoken dialogues. In *Human Language Technology. Challenges of the Information Society. Third Language and Technology Conference, LTC 2007, Poznan, Poland, October 5-7, 2007, Revised Selected Papers, LNCS 5603*, pages 50–62. Springer, 2009.

[72] NIST. Speech recognition scoring toolkit (SCTK). http://www.nist.gov/speech/tools/.

[73] F.J. Och. Yet another small maxent toolkit. http://www-i6.informatik.rwth-aachen.de/web/Software/YASMET.html, 2002.

[74] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, 31(1):71–106, 2005.

[75] Daniele Pighin and Alessandro Moschitti. Reverse engineering of tree kernel feature spaces. In *EMNLP09: Empirical Methods of Natural Language Processing*, Singapore, 08/2009 2009.

[76] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Proceedings of the Biennial Conference on Numerical Analysis*, pages 144–157, Dundee, UK, June 1977. Lecture Notes in Mathematics, Vol. 630. Berlin, Heidelberg, New York: Springer 1978.

[77] S. Quarteroni, G. Riccardi, and M. Dinarelli. What's in an ontology for spoken language understanding. In *Proc. of Interspeech 2009*, Brighton, U.K., 2009.

[78] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[79] Lance Ramshaw and Mitchell Marcus. Text chunking using transformationbased learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 84–94, Cambridge, MA, USA, June 1995.

[80] Christian Raymond, Frdric Bchet, Renato De Mori, and Graldine Damnati. On the use of finite state transducers for semantic interpretation. *Speech Communication*, 48(3-4):288–304, March-April 2006.

[81] Christian Raymond and Giuseppe Riccardi. Generative and discriminative algorithms for spoken language understanding. In *Interspeech*, pages 1605–1608, Antwerp, Belgium, August 2007.

[82] Ruhi Sarikaya, Yuqing Gao, Michael Picheny, and Hakan Erdogan. Semantic confidence measurement for spoken dialogue systems. *IEEE Trans. on SAP*, 13:534–545, 2005.

[83] Roger C. Schank and Robert P. Abelson. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* Psychology Press, Oxford, England, 1977.

[84] R.E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[85] Stephanie Seneff. Tina: a natural language system for spoken language applications. *Comput. Linguist.*, 18(1):61–86, 1992.

[86] Christophe Servan, Christian Raymond, Frdric Bchet, and Pascal Nocra. Conceptual decoding from word lattices: application to the spoken dialogue corpus media. In *Proceedings of the International Conference on Spoken Language Processing*, Pittsburgh, USA, 2006.

[87] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

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

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

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

[91] Hsuan tien Lin, Chih jen Lin, and Ruby C. Weng. A note on platts probabilistic outputs for support vector machines. Technical report, 2003.

[92] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

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

[94] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[95] S. Varges, G. Riccardi, and S. Quarteroni. Persistent information state in a data-centric architecture. In *Proc. of SIGDial 2008*, Columbus, USA, 2008.

[96] Geoffrey Zweig and Stuart Russell. Speech recognition with dynamic bayesian networks. Technical report, 1998.