PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

SEMANTIC LANGUAGE MODELS WITH DEEP NEURAL NETWORKS

Ali Orkan Bayer

Advisor:

Prof. Giuseppe Riccardi University of Trento

Examining Committee:

Prof. Marco Gori	University of Siena
Prof. Murat Saraçlar	Boğaziçi University
Prof. Marco Baroni	University of Trento
Prof. Renato De Mori	McGill University

April 2015

Abstract

Spoken language systems (SLS) communicate with users in natural language through speech. There are two main problems related to processing the spoken input in SLS. The first one is automatic speech recognition (ASR) which recognizes what the user says. The second one is spoken language understanding (SLU) which understands what the user means. We focus on the language model (LM) component of SLS. LMs constrain the search space that is used in the search for the best hypothesis. Therefore, they play a crucial role in the performance of SLS.

It has long been discussed that an improvement in the recognition performance does not necessarily yield a better understanding performance. Therefore, optimization of LMs for the understanding performance is crucial. In addition, long-range dependencies in languages are hard to handle with statistical language models. These two problems are addressed in this thesis.

We investigate two different LM structures. The first LM that we investigate enable SLS to understand better what they recognize by searching the ASR hypotheses for the best understanding performance. We refer to these models as joint LMs. They use lexical and semantic units jointly in the LM. The second LM structure uses the semantic context of an utterance, which can also be described as "what the system understands", to search for a better hypothesis that improves the recognition and the understanding performance. We refer to these models as semantic LMs

(SELMs). SELMs use features that are based on a well established theory of lexical semantics, namely the theory of frame semantics. They incorporate the semantic features which are extracted from the ASR hypothesis into the LM and handle long-range dependencies by using the semantic relationships between words and semantic context. ASR noise is propagated to the semantic features, to suppress this noise we introduce the use of deep semantic encodings for semantic feature extraction. In this way, SELMs optimize both the recognition and the understanding performance.

Keywords

[Language Models, Automatic Speech Recognition, Spoken Language Understanding, Neural Networks, Deep Autoencoders]

Acknowledgments

I would like to thank my advisor Professor Giuseppe Riccardi, this work would not be possible without his guidance and support.

I would like to thank all my colleagues for creating a lovely work environment with their existence and friendship. Special thanks goes to Carmelo Ferrante for his endless help regarding any issue I faced up to, especially for finding me a place to live. Evgeny Stepanov for all the chats, discussions, his support and ideas, and also for lots of free smoke. Arindam Ghosh for all the support he has given and all the fun we had. Darién Miranda for his young energy, to Belén Agüeras for her endless smile, to Katalin Jászkuti for supporting me whenever I feel down.

I would like to thank all my friends who was part of this journey. Sinan Mutlu, the first person I met in Trento, was always my support for all the bad times I had, with the huge heart he has. Gözde Özbal who played a role for me starting this journey always supported me. Begüm Demir always told me to keep calm and she found a way to relieve me everytime I was stressed. Serra Sinem Tekiroğlu was another support with her positive attitude. Azad Abad was always a good friend when I was bored of everything. Saameh Golzadeh for listening to my endless problems without even complaining and I would like to thank Ekaterina Panina for making me keep my hope in people.

I owe everything to my family. My grandmother Ruhat Kütükbaşı and my parents Hülya Bayer and Ersin Bayer have always shared everything

they with me and always directed me in the correct direction. I am pretty sure that I would not be able to come to this point in my life without their support. My brother Ertan Bayer and my sister-in-law Ayça Bayer; although they were far away, I always felt their support next to me.

I would like to thank Ayça Müge Sevinç for always being right by my side since we have entered the world of academic research together more than a decade ago. I would not be the person I am today if I had not met her. I appreciate her understanding and help during the hard times of this Ph.D. and thank her again for proof reading this thesis.

To my grandmother, Ruhat Kütükbaşı
who always wanted to see me as a doctor.

To my parents, Hülya Bayer and Ersin Bayer

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."

Alan Turing

Alan Turing, after the invention of digital computers brought the possibility of thinking machines [124] to the agenda. He designed a game called "imitation game", in which a human interrogator asks questions to two players, a machine and a human, to find out which one is the human and which one is the machine. Turing designed the game so that all the information that is passed between the interrogator and the players is through typing. He must have set the communication in this way, probably because at the time, machines communicating through natural speech were seemed infeasible. However, 65 years after Turing designed the imitation game, we can talk about *intelligent* systems that can recognize and understand human speech.

Still far from being *intelligent* in the sense Turing implies[1], these ma-

[1] A state-of-the-art spoken dialog system may not be able deceive a human, since by putting a little effort one can easily break a spoken dialog system.

chines started to take an important role in our lives with the increasing use of mobile devices. Among these systems, intelligent personal assistants may be among the most widely used applications, which can communicate through speech. Also, voice search is very satisfactory and efficient, especially on mobile phones where typing is more error prone and cumbersome.

This thesis concentrates on the two modules of spoken dialogue systems (SDS) and spoken language systems (SLS): the automatic speech recognition (ASR) and the spoken language understanding (SLU). A diagram of a typical SDS system is given in Figure 1.1 [89].

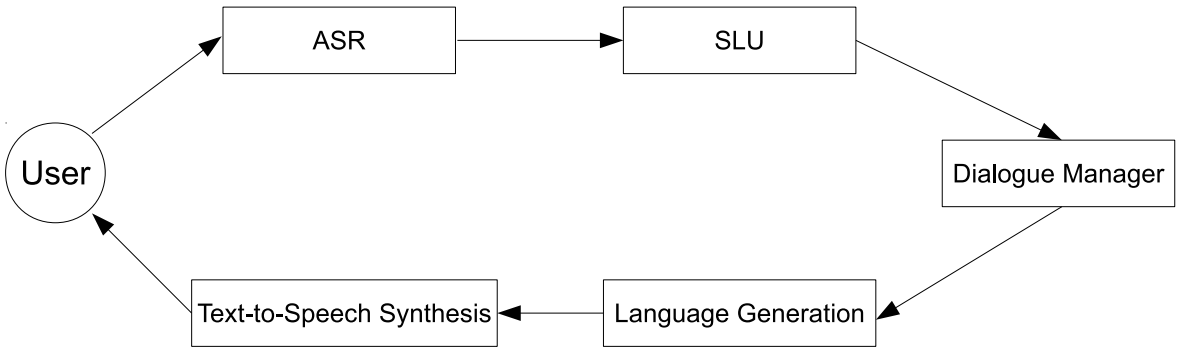

Figure 1.1: A typical spoken dialogue system. The typical system uses a cascaded architecture.

ASR is the problem of recognizing what words are uttered by the user. ASR systems base their hypotheses on two different information sources which work in combination: acoustic models and language models. Acoustic models focus on the recognition of the speech sound sequences, and language models (LMs) estimate the probability of word sequences. This thesis focuses on the LM component of the ASR module.

Although a system that recognizes each word the user utters can be considered to be successful, a system that is able to communicate through speech also needs to understand what the user means. Therefore, spoken language understanding (SLU) plays a crucial role in SLS. SLU is the main

component that understands what the user expects from the system. In this way, the system could act accordingly to satisfy user's needs.

1.1 Motivation

Most often the two problems, ASR and SLU are approached separately when building SLS as shown in the cascaded architecture in Figure 1.1. First an ASR module is trained and it is optimized for the recognition performance. Then an SLU module that uses the output of the ASR is trained and optimized for the understanding performance. However, as it has been pointed out many times, the hypothesis that gives a better recognition performance does not always yield a better understanding performance [108, 127, 38]. If the end goal of SLS is to understand what the user means and respond accordingly, both modules can be optimized jointly such that the system "understands better what it recognizes" and "recognizes better what it understands".

We believe that LMs that satisfy lexical and semantic constraints jointly would have a better joint performance of recognition and understanding. Therefore, we focus on joint LMs which are constrained by lexical and semantic structures at the same time.

1.2 Contribution of the Thesis

This thesis aims at incorporating semantic features into LMs for a better joint performance of recognition and understanding, and for handling long-range dependencies. We use neural network LMs that use distributed representations of words. In this manner, they estimate word probabilities better. We investigate two different LM structures. The first model is the "joint LM" that uses word-concept pairs as the units of language modeling.

We show how to train and optimize joint LMs for recognition and understanding. The second model is the "semantic LM" that uses features that are based on the theory of frame semantics [46]. We train semantic LMs by using features extracted from the semantic context. Also, we introduce the use of deep autoencoders to extract more robust semantic features.

This thesis makes the following contributions:

- Training of joint LMs by using lexical and semantic information and optimizing LMs for recognition and understanding.
- On-line adaptation of joint LMs for improving the understanding performance.
- Application of joint LMs to cross-language SLU porting.
- Training semantic LMs by exploiting the theory of frame semantics.
- Noisy representation of semantic (linguistic) scene by means of deep autoencoders.

1.3 Publications Relevant to the Thesis

The following publications are relevant to this thesis. They are revised and extended in the preparation of the thesis.

- Bayer A.O. and Riccardi G., "Joint Language Models for Automatic Speech Recognition and Understanding", December 2 - 5, 2012, IEEE Workshop on Spoken Language Technology (SLT 2012), Miami, Florida, USA. [5]
- Bayer A. O. and Riccardi G., "On-line Adaptation of Semantic Models for Spoken Language Understanding", December 8 - 12, 2013, IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2013), Olomouc, Czech Republic. [6]

- Stepanov E., Kashkarev I., Bayer A. O., Riccardi G. and Ghosh A., "Language Style and Domain Adaptation for Cross-Language Porting", December 8 - 12, 2013, IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2013), Olomouc, Czech Republic. [119]

- Bayer A.O. and Riccardi G., "Semantic Language Models for Automatic Speech Recognition", December 7 - 10, 2014, IEEE Workshop on Spoken Language Technology (SLT 2014), South Lake Tahoe, USA. [7]

- Bayer A.O. and Riccardi G., "Deep Semantic Encodings for Language Modeling", September 6 - 10, 2015, Interspeech 2015, Dresden, Germany. [8]

1.4 Structure of the Thesis

This thesis is structured as follows.

Chapter 2 gives an overview of the ASR and SLU modules and discusses the problems that are addressed in this thesis.

Chapter 3 describes the details of statistical language modeling. The chapter starts with the introduction of the n-gram models and also presents the advanced LM models that uses neural network architectures.

Chapter 4 introduces the problem of spoken language understanding and discusses meaning representations. In addition, it presents computational models for spoken language understanding.

Chapter 5 describes the computational models that are used in this thesis. This chapter is devoted to neural network architectures.

Chapter 6 presents the corpora that is used in the experimental work and gives the baseline models and their performance on these corpora.

Chapter 7 describes the training of joint LMs and their optimization for recognition and understanding. Also, on-line adaptation of these semantic models are presented. Finally, applications of joint LMs to cross-language SLU porting is described.

Chapter 8 introduces semantic LMs that are based on the theory of frame semantics. This chapter presents which semantic features are extracted and how these features are integrated to LMs to obtain an acceptable performance.

Chapter 9 presents the use of deep autoencoders for high-level semantic encodings that can be used by semantic LMs.

Chapter 10 concludes this thesis by discussing the contributions and possible future work.

Chapter 2

Background and Relevant Problems

Using speech as the medium of human-computer interaction enables users to communicate easily with computer systems. However, this introduces many research challenges for building spoken language systems (SLS) that function successfully. One of the main problems related to SLS is the *recognition* of what the user says, namely, automatic speech recognition (ASR). Another main problem is the *understanding* of what is being meant. Although these two problems may seem like two separate problems, in this thesis, we show that both *recognition* and *understanding* may benefit from each other and can be approached jointly.

2.1 Automatic Speech Recognition

The *recognition* problem aims at correctly transcribing every utterance it encounters. Therefore, the main goal of an ASR system is to increase the transcription performance of each word the user utters. The first automatic speech recognizer were built in 1950s for the recognition of digits that were spoken by a single individual [33]. Therefore, it was a speaker independent system. This first device could be adapted to another individual by performing manual analysis on that individual's speech, and it could recognize digits by using spectral resonances of the vowels. The foundation

of statistical speech recognition, which the current systems are based on, was introduced in 1970s by Jelinek [67] and Baker [4].

The statistical speech recognition approach models the speech recognition problem as follows [68]. A recognizer hypothesis is a string of words, $\mathbf{W}$, which are drawn from a finite vocabulary based on some acoustic evidence $\mathbf{A}$. The word string $\hat{\mathbf{W}}$, that maximizes the conditional probability of word strings given that acoustic evidence, $P(\mathbf{W}|\mathbf{A})$ is the hypothesis that the recognizer searches for:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{A})$$

Using the Bayes' formula of probability theory, the above equation can be written as:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \frac{P(\mathbf{W})P(\mathbf{A}|\mathbf{W})}{P(\mathbf{A})} \tag{2.1}$$

The denominator in equation 2.1 can be ignored during maximization for a given input utterance. Therefore, a statistical speech recognition system uses two knowledge sources:

1. $P(\mathbf{W})$: the probability of a word sequence in a language and it is computed by a "language model"

2. $P(\mathbf{A}|\mathbf{W})$: the probability of the acoustic observations given a word sequence which is computed by an "acoustic model".

Briefly, an ASR system bases its hypothesis on the information coming from the "language model" and the "acoustic model". This thesis focuses on the "language model" and uses the traditional approaches for the "acoustic model".

2.1.1 Acoustic Model

Acoustic models are not within the scope of this thesis, therefore they will not be treated in detail. A good overview of traditional acoustic modeling can be found in [135]. Traditionally, acoustic models are based on hidden Markov models (HMMs) that model each phoneme in the language. The phonemes can also be modeled considering the preceding and the succeeding phonetic context, in which case they are called "context-dependent phonemes".

The current improvements in training deep neural networks have been resulted in significant improvements also in acoustic modeling. An overview about acoustic models based on deep neural networks can be found in [57, 55]. However, in this thesis we employ the traditional HMM acoustic models with more recent techniques that Kaldi [105] speech recognition toolkit provides.

2.1.2 Language Model

Language models (LMs) estimate the probability of occurrences of word sequences in that language. Therefore, they play an important role during the search for the most likely hypothesis. Chapter 3 gives a detailed overview of statistical language modeling. In this section we focus mainly on the problems related to the current approaches, and our approach to language modeling.

Currently, the most widely used LMs are n-grams, because they are simple models with good baseline performance. N-grams estimate probabilities of words by simply counting them in a fixed history, i.e. they assume that the probability of a word is only dependent on the previous $n-1$ words [109]. In n-grams, the probability of a word, w_i given a history h_i is estimated by the equation 2.2.

$$P(w_i|h_i) \approx p(w_i|w_{i-n+1}, \cdots, w_{i-1}) \tag{2.2}$$

In this respect, n-gram LMs are based on limited size of words occurring together. They treat words as a sequence of symbols and do not exploit the fact that they model a language [109]. Also, one of most important weaknesses of n-grams is the assumption that a word only depends on previous $n-1$ words and it is independent of the other words [109]. Therefore it is trivial that n-grams fail to capture long-range dependencies that occur naturally in language.

Bellegarda [10] refers to this problem as the *locality problem* and explains the problem with the following example. Consider that the LM is trying to predict the word *"fell"* from the word *"stocks"* in the following sentences:

stocks fell sharply as a result of the announcement (2.3)

stocks, as a result of the announcement, sharply fell (2.4)

In Sentence 2.3 *"fell"* can be predicted with a bi-gram LM. However, in 2.4 a 9-gram model would be needed for this prediction which is not feasible. In addition, one can embed more words in between the commas in 2.4 without making the sentence ungrammatical, in this case even a 9-gram would not be sufficient. Hence, models that are based on fixed histories cannot handle long-range dependencies.

The *locality problem* can be solved by extending the span of the LM to handle the long-range dependencies of the language (*span extension*). There are two linguistically motivated approaches to extend the span. The first approach is to use the structural information, i.e. the syntax and the second one is to use the semantic information.

The attempts to incorporate syntactic structure into LM started with the usage of hand-written context free grammars (CFGs), which are used

to parse the word lattice that is generated by the ASR [27]. In [137], LMs are trained by generating syntactically plausible sentences by using a natural language component. Jurafsky et al. [71] use stochastic CFGs (SCFGs) to extend the corpus for training and interpolates SCFG probabilities with bi-gram probabilities. Chelba et al. [24] use a dependency grammar framework with maximum entropy models to constrain word prediction by the linguistically related words in the past. The most important instance of LMs that use syntactic structure is presented in [25]. This model incrementally parses hypothesis in a left-to-right manner and assigns a probability to a word considering its parses. However, structured LMs rely on syntactic parsers, therefore, affected by the errors made by the parser [9].

The problem of handling long-range semantic dependencies is approached by *trigger pairs* in Lau et al. [82]. In this approach, significantly correlated word sequences are considered as trigger pairs and the occurrence of a triggering sequence causes the probability estimate of the triggered sequence to change. The trigger pairs are modeled by means of feature functions of maximum entropy models. The identification of trigger pairs is an issue and in [110] it has been shown that self triggers are powerful and robust. Bellegarda [10, 11] applies latent semantic analysis (LSA) to LMs for handling long-range semantic dependencies. LSA [35, 16] is used as an indexing mechanism in information retrieval, where the span of the history is a document which is larger than a sentence.

In this thesis, we consider incorporating semantic relations in LM by using the theory of frame semantics [46]. This approach applies a linguistic semantic theory to language modeling and constructs a linguistic scene by using the information coming from frames evoked in the utterance.

Another problem related to n-gram LMs is the *curse of dimensionality* [14]. In traditional LMs words are treated as discrete variables. As Bengio et al. [14] point out, curse of dimensionality would be an issue

when one wants to model a joint distribution between many words that are represented by discrete random variables. In addition, this kind of representation never considers the similarity between words [14]. For a traditional LM a *cat* to a *dog* is no more similar than a *cat* to a *room*. One of the solutions to this problem is to represent words in a continuous space, such that each word is represented with an m dimensional vector and these vectors are more closer to each other as the words they represent are more similar. As proposed by Bengio et al. [14], in this approach, each word is associated with a distributed word feature vector and joint probabilities of word sequences are expressed in terms of these feature vectors.

The problem of "curse of dimensionality" is solved by using distributed word representations, where each word is represented by a feature vector [14] on a continuous space. Bengio et al. [14] introduce how LMs that uses distributed representations are modeled, and how these representations are learned by a feed-forward neural network. They have shown significant improvements in perplexity. Schwenk [116] uses neural network LMs for re-scoring multiple ASR hypotheses. Distributed representations have also attracted attention and have been applied to various natural language processing problems [28], in which they were also called "word embeddings".

Another significant improvement in the recognition performance is obtained by the use of recurrent neural networks (RNNs) [96, 80, 94]. RNNs use recurrent connections, which resemble a short-time memory that remembers the state of the model, in this way they can model theoretically infinite histories i.e. long-range dependencies.

2.1.3 Evaluation of ASR Systems

ASR systems conduct a search on the possible hypothesis by using acoustic models and LMs. This search is also referred to as *decoding*. ASR

systems, rather than outputting a single hypothesis (1st-best hypothesis), can output multiple hypotheses by using lattices, or n-best lists extracted from these lattices. These lattices or n-best lists can then be re-scored by different models to improve the performance. The evaluation of ASR can be done over the 1st-best or over the *oracle hypothesis*, which is the best hypothesis in the lattice with respect to the reference transcription.

The statistical speech recognition uses the noisy channel approach. In this approach, the system may make three kinds of errors [2]:

1. : Deletions (D): The hypothesis of the system misses some of the words.

2. : Insertions (I): The hypothesis of the system inserts some extra words.

3. : Substitutions (S): The hypothesis of the system misrecognizes some words and confuses them with similar words.

After the hypothesis of an ASR system is aligned with the reference transcription, these errors are computed and the overall error of the system, word error rate (WER), is calculated as in Equation 2.5, where N refers to the number of words in the reference transcription.

$$WER = \frac{D + I + S}{N} \tag{2.5}$$

2.2 Spoken Language Understanding

The *spoken language understanding* problem, on the other hand, aims at correctly interpreting what the utterance means. SLU has common goals with natural language understanding as both systems aim at obtaining a meaning representation for the user input which is in natural language.

However, in general it is a harder problem, because of the noisy nature of speech in terms of the ASR word error as well as the variability of the ungramaticality of the conversational speech. [34].

The term SLU defines a very broad research area, the possible set of problems that can be considered in the scope of SLU contains intent determination, spoken utterance classification, topic identification, and speech summarization. This thesis focuses on the semantic frame-based SLU, which is one of the main building blocks of spoken language systems. Semantic-frame-based SLU works on a task specific limited domain which is usually defined by an ontology and aims at correctly recognizing the frames evoked and correctly filling the slots of these frames [130].

As stated in Wang et al. [130], the studies on SLU were started in 1970s with the Defense Advanced Research Projects Agency (DARPA) tasks on speech understanding and resource management. The interest in SLU research accelerated with the evaluations on the Air Travel Information System (ATIS) which was also sponsored by DARPA in 1990s [106]. In the ATIS domain the SLU task is to respond to user queries about air travel information which are in spontaneous speech. The system responds to queries by converting these queries to SQL statements and by retrieving the inquired information. Two different approaches emerged from these studies, knowledge-based approaches and statistical approaches. Knowledge-based approaches are based on hand-crafted grammars and not in the scope of this thesis. On the contrary, we focus on statistical systems that can learn from annotated data which are more robust and scalable [130].

2.2.1 Meaning Representation

In semantic-frame-based SLU, the semantic information is represented by *semantic frames* [129]. Semantic frames are defined with respect to the domain of the application and each frame contains *slots* that needs to be

filled with the required information. For instance, the following sentence in the ATIS domain triggers two frames: *"Show me flights from Seattle to Boston"* [129].

In addition to the semantic frame representation, the meaning can also be represented as a sequence of *basic units*. [104] uses *keyword-value* pairs, (k_j, v_j), where keywords (k_j) represent the conceptual categories like *destination city* and values (v_j) are the words which were assigned to these categories like *Boston* in the above example. This representation is also called *attribute-value* pairs or *flat concept representation*. We adopt the attribute-value pair representation in the scope of this thesis.

The SLU models used in this thesis use the flat concept representation with *in/out/begin (IOB)* representation to handle the concepts mappings that span multiple words. In this approach, the concept tags have additional attributes; "-B" marks the beginning of the concept alignment, and "-I" marks the rest of the alignment. Figure 2.1 gives an example of this representation from the Italian LUNA corpus [41].

Buongiorno	io	ho	un	problema
null	null	null	HardwareProblem.type-B	HardwareProblem.type-B

con	la	stampante
Peripheral.type-B	Peripheral.type-I	Peripheral.type-I

Figure 2.1: The flat concept representation with (IOB) representation.

In this example, which means "Good morning, I have a problem with the printer", the "Peripheral.type" concept is aligned with the phrase "con la stampante".

2.2.2 Statistical SLU

The SLU problem can also be approached as a sequence labeling problem i.e. the problem of labeling a sequence of words with the correct concept tags. A comparison of various statistical models are given in [53]. The statistical approach can be a generative approach or a discriminative approach. In this thesis we have used the generative approaches that are based on stochastic finite state transducers (SFSTs) [107]. and discriminative approaches that are based on conditional random fields (CRFs) [81]. SFSTs model finite state transducers for making alignments between word sequences and concepts [107] by modeling the joint probability $P(W, C)$, where W represents the word sequence and C represents the concept sequence. CRFs, on the other hand, model the conditional probability $P(C|W)$ by log-linear models with feature functions. CRFs perform significantly better than SFSTs [53].

The current state-of-the-art uses neural network models that exploit the distributed word representations as in language modeling [122, 90, 133, 132, 131]. Using distributed word representation improves the performance of SLU in general.

2.2.3 Using Multiple Hypotheses

Usually SLU systems use a cascaded approach where the ASR 1st-best hypothesis is fed into the SLU module and the semantic representations are extracted from this single hypothesis. However, a system that uses multiple hypotheses for SLU would be more robust to ASR noise. Word confusion networks that are extracted from ASR lattice have been used in [54, 123]. A joint decoding algorithm is proposed for jointly performing recognition and understanding in [38]. Re-ranking models that re-ranks the multiple hypotheses of a generative SLU system by using support vector

machines is given in [40].

In this thesis, we employ a similar approach to [136]. We use joint neural network LMs that are trained on word-concept pairs (*joint LMs*) to re-score the output of a baseline SLU model that is built by SFSTs and CRFs. We also address the issue of adaptation of this joint model to the context of the utterance.

2.2.4 Cross-Language SLU porting

Cross-language porting is the problem of transferring the semantic knowledge obtained in one language (the *source language*) to a new language (the *target language*) when there is no semantic annotation available for the target language [65, 117]. Cross-language porting uses statistical machine translation (SMT) for translating and aligning the resources. The methodology is divided in two categories with respect to the direction of translation: *Test-on-Source* and *Test-on-Target* [64]. Test-on-Source uses SMT to translate the utterance to the source language and uses the SLU model in the source language to extract the semantic representation. Test-on-Target, on the other hand, first transfers the semantically annotated resource to the target language and builds an SLU model on the target language.

In this thesis, we apply joint LMs to cross-language SLU porting on the Test-on-Source setting. In this application, multiple hypotheses generated by the SMT system are re-scored by using the joint LM.

Chapter 3

Statistical Language Modeling

Statistical language models, which will be referred to as language models (LMs) in this thesis, are one of the knowledge sources ASR systems. LMs estimate the probability of utterances in a language by capturing the statistical regularities in that language.

The problem of estimating the probability of an utterance, $P(U)$ is formulated as given in [68]:

The transcription of an utterance U is denoted by W as a sequence of words which are elements of a finite vocabulary $\mathcal{V}$:

$$W = w_1, w_2, w_3, \ldots, w_n \quad \forall w_i \in \mathcal{V}$$

The probability of W is factorized as in Equation 3.1. The preceding words $(w_1, w_2, \ldots, w_{i-1})$ that the current word (w_i) is conditioned on are referred to as the *history*. Histories may be further mapped into equivalence classes. This mapping leads to an approximate but a simple probabilistic model which would otherwise suffer from data sparseness.

$$P(w_1, w_2, w_3, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_1, w_2, \ldots, w_{i-1}) \tag{3.1}$$

3.1 N-gram LMs

The idea of having equivalence classes on histories leads to the most commonly used LMs, n-gram LMs. In n-gram LMs, equivalence classes are constructed based on the most recent words in the history. Therefore, n-gram models use the most recent $n-1$ words to build the equivalence classes. Hence, n-gram LM is represented as in Equation 3.2.

$$P(w_1, w_2, w_3, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-n+1}, \ldots, w_{i-1}) \tag{3.2}$$

Each factor in Equation 3.2 estimates a probability of each word, w_i, in the utterance given a limited window size of $n-1$ words. As can be seen, n-gram LMs make the independence assumption that a word is only dependent on the previous $n-1$ words. These probabilities may be estimated by taking the relevant counts in the training corpus [52]. The maximum likelihood estimates of tri-grams are given as in Equation 3.3, where $C(x)$ denotes the count of the n-gram x.

$$P(w_i | w_{i-1}, w_{i-2}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \tag{3.3}$$

Estimating probabilities by just using the counts would not give good estimates and the estimates would be noisy. In addition, since some n-grams would not be observed in the training data they would be assigned zero probabilities. To improve the quality of estimation several, smoothing methods have been introduced and important ones are described in Section 3.2.

3.2 Smoothing

In this section, an overview of some of the important smoothing techniques are given, a more detailed overview of various smoothing techniques can be found in [26].

Katz [72] proposes the idea of redistributing the over-estimated observed n-gram probabilities to the n-grams that do not occur. This is based on Good-Turing formula [51]. Assume that n_r denotes the number of n-grams that occur in the corpus r times. The count is discounted to r^* or $disc(x)$, as seen in Equation 5.1.

$$disc(x) = r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{3.4}$$

Katz smoothing distributes the discounted amount to the n-grams that are never seen. Therefore, the discounted counts, r^*, are used when estimating the probabilities for seen n-grams. The probabilities for unseen n-grams are given by backing-off to a lower order n-gram. The Katz probability estimates are given in Equation 3.5, where $\alpha(x)$ is a normalization factor for obtaining a probability distribution.

$$P_{KATZ}(w_i|w_{i-n+1},\ldots,w_{i-1}) = \begin{cases} \frac{disc(w_{i-n+1},\ldots,w_{i-1},w_i)}{C(w_{i-n+1},\ldots,w_{i-1})} & \text{if } C(w_{i-n+1},\ldots,w_{i-1}) > 0 \\ \alpha(w_{i-n+1},\ldots,w_{i-1})P_{KATZ}(w_i|w_{i-n+2},\ldots,w_{i-1}) & \text{otherwise} \end{cases} \tag{3.5}$$

Kneser-Ney smoothing uses a different back-off distribution by considering the number of contexts a word occurs rather than the occurrence of that word [74]. Therefore, the back-off distribution for a word that occurs in higher number of contexts would be higher than the back-off distribution

for a word that occurs in lower number of contexts. Kneser-Ney smoothing uses a single discount, D, which is optimized over a held-out set. Kneser-Ney smoothing can be described by the following formula in Equation 3.6 for a bi-gram model [52], where $|\{v|C(vw_i) > 0\}|$ represents the number of distinct context w_i occurs in. The factor $\alpha(x)$ is a normalization factor for obtaining a probability distribution.

$$P_{KN}(w_i|w_{i-n+1}) = \begin{cases} \frac{C(w_{i-1},w_i)-D}{C(w_{i-1})} & \text{if } C(w_{i-1},w_i) > 0 \\ \alpha(w_{i-1})\frac{|\{v|C(vw_iy)>0\}|}{\sum_w |\{v|C(vw)>0\}|} & \text{otherwise} \end{cases} \quad (3.6)$$

Chen and Goodman [26] improve Kneser-Ney smoothing by interpolating higher order and lower order counts rather than backing-off to a lower order count when the higher order count is not found. Therefore, interpolated models combine higher order and lower order counts at every occasion. The interpolated Kneser-Ney smoothing for a bi-gram model is given in Equation 3.7.

$$P_{IKN}(w_i|w_{i-n+1}) = \frac{C(w_{i-1},w_i) - D}{C(w_{i-1})} + \lambda(w_i)\frac{|\{v|C(vw_iy) > 0\}|}{\sum_w |\{v|C(vw) > 0\}|} \quad (3.7)$$

Chen and Goodman [26] make further improvements to the interpolated Kneser-Ney smoothing by using multiple discounts; one for single counts, one for double counts, and one for triple or more counts. This is referred as the modified Kneser-Ney smoothing and shown to be the best smoothing technique for large data sets. In this thesis, we have used this smoothing technique when training n-gram models.

3.3 Class-Based LMs

It is possible to cluster words into classes with respect to their semantic or syntactic similarities. In this way, more reliable estimates of probabilities

can be calculated for unseen histories based on these classes [21]. The class-based n-grams use a partition function, which maps a word, w_i into a class, c_i and they are defined by Equation 3.8.

$$P(w_n|w_{n-1},\ldots,w_1) = P(w_n|c_n)P(c_n|c_{n-1},\ldots,c_1) \tag{3.8}$$

The clustering on words can be performed by using a knowledge base or by using data-driven techniques. The knowledge-based approaches may use part-of-speech (POS) tags or even hand-crafted classes like "DATE", "NAME", "COMPANY", etc. [66, 56]. Data-driven approaches cluster words to minimize the overall perplexity of the corpus by a greedy approach [21, 75]. It has been shown that data-driven approaches outperform classes based on POS tags [102].

Neural network LMs (NNLMs) may also use class-based factorization of the probability estimates. The most important reason for that is the training complexity of NNLMs, especially when the vocabulary size is very large [97]. In the NNLMs we have used in this thesis, we use word classes that are based on the unigram frequencies of the words [97], our main purpose to use classes is to reduce the computational complexity of NNLMs.

3.4 Maximum Entropy LMs

The principle of maximum entropy models is to find the most uniform distribution, i.e., the distribution that maximizes the entropy that is consistent with all the constraints that are imposed on the model [15]. The constraints are imposed on the model by defining binary *feature functions* or *features*, f_i. These features return 1 if a constraint is satisfied and 0 otherwise. Maximum entropy models are introduced to LM by Rosenfeld [110]. Maximum entropy LMs are in the form given by Equation 3.9 [52].

$$P(w_n|w_{n-1},\ldots,w_1) = \frac{exp(\sum_k \lambda_k f_k(w_n, w_{n-1},\ldots,w_1))}{z(w_{n-1},\ldots,w_1)} \tag{3.9}$$

The denominator is a normalization factor that is given by Equation 3.10.

$$z(w_{n-1},\ldots,w_1) = \sum_w exp(\sum_k \lambda_k f_k(w, w_{n-1},\ldots,w_1)) \tag{3.10}$$

The weights, λ_k, are learned by using a learning algorithm like generalized iterative scaling [31]. The features may represent n-grams, caches, and skipping models. Maximum entropy models are reported not to improve much with respect to comparable interpolated n-gram models [52]. And also they are reported to be very time consuming to train and test.

One of the contributions of maximum entropy LMs is the use of *trigger pairs* as features [82]. Trigger pairs are defined by a trigger function between words, therefore if a word sequence A is significantly correlated with a word sequence B, they constitute a trigger pair. Then, A is referred to as the *trigger* and B as the *triggered sequence*. Rosenfeld [110] reports that *self triggers* are very powerful and robust. Also trigger pairs of frequent words have more potential than the trigger pairs of infrequent words. Trigger pairs are determined by using the average mutual information between the trigger and the triggered sequence. Trigger pairs are very effective to handle the long-range dependencies.

Another type of maximum entropy LMs is the *whole sentence model given* in [111]. This model predicts the probability of the whole sentence by using feature functions over the entire sentence. However, the training of these models require sampling methods which makes the model very complex.

3.5 Structured LMs

Structured LMs aim at exploiting the syntactic structure of utterances to handle long-range dependencies better than a fixed context. The first attempts are based on using context free grammars (CFGs) [27, 137, 71]. The main contribution of structured LMs is started with Chelba et al. [24] in which a dependency grammar framework with maximum entropy models is used to constrain the word prediction by the linguistically related words in the past.

The most notable work in structured modeling is the work presented by Chelba and Jelinek [25]. This model uses incremental parsing to assign probabilities to words and parses in a left-to-right manner. This LM creates a binary branching parse of the sentence with its POS tag assignments, head-word annotation, and non-terminal label; and it assigns a probability for "word sequence and parse" pairs. This LM consists of three modules, "word-predictor" which predicts the next word, "tagger" which predicts the POS tag of the next word, and "constructor" which grows the existing binary branching parse tree. Since the word prediction is based on the current parse of the sentence, it takes the syntactic structure into consideration. It is reported to improve perplexity and word error rates, however, the major drawback of structured models is the fact that they depend on the performance of the syntactic parser. In addition, for spoken language, which contains hesitations and repetitions, the performance of a syntactic parser would be questionable. The structured LMs are also modeled by neural networks [45, 44], which benefit from the distributed representations. The neural networks model the "word-predictor" module and take word and non-terminal tag features as input. They are reported to improve perplexity and word error rate significantly.

3.6 Semantic LMs

Semantic LMs consider the semantic dependencies in the language. One of the approaches that incorporates semantic information into LM is the *topic model* [49, 114]. The topics may be selected from a hand-crafted set or can be learned by data-driven approaches. Topic LMs model the probability of a word in a topic and do not consider the local structure of the language. Therefore they are combined with n-gram models.

Trigger pairs as introduced in Section 3.4 may also be considered as a form of a semantic LM since they consider the significant correlations between trigger pairs. The performance of trigger pairs are dependent mostly on the selection of pairs.

Bellegarda [10, 11] uses *latent semantic analysis* (LSA) to extend the trigger pair approach. LSA [35, 16] is used as an indexing mechanism in information retrieval, it maps the discrete space of words and documents[1] to the same continuous space. Therefore, each word and document is represented as a vector in this space. A word-document matrix, in which each column represents a document and each row represents a word, is constructed by populating the matrix values by normalized counts of the words in the documents. The normalization is done with respect to the number of documents in which the word occurs. Because of computational requirements, singular value decomposition is applied to this matrix. The final representation conceptually represents each word and document as a linear combination of abstract concepts, which is very similar to the distributed representations the neural network LMs use. At the final step, LMs are modeled over the LSA history of the word. Combining LSA with n-gram models resulted in significant improvements in perplexity and word error rate [10, 11].

[1] A document refers to a group of sentences that are semantically related, i.e., have the same topic.

3.7 Neural Network LMs

N-gram LMs represent the words on a discrete space where it is not possible to model any relationships between words, regarding their position in this space. Neural network LMs (NNLMs) address this issue which suffers from curse of dimensionality. During training, NNLMs learn distributed representations, i.e., continuous valued vectors of words. Hence, NNLMs, associate a distributed representation (also known as a *word embedding*) with each word in the vocabulary and model the joint probability of word sequences over these vectors [14].

NNLMs are first started to be used in [14] and they were reported to reduce the perplexity. The first application of NNLMs to ASR is presented in [116] which is reported to drop the WER by linear interpolating NNLMs with back-off n-gram LMs. The detailed structure of NNLMs and their training procedure is presented in Chapter 5. The structure of feed-forward NNLMs is given in Figure 5.3 [14].

The input to feed-forward NNLMs is given as 1-of-n encoding, in which the index of the word is set to 1 and the rest is set to 0. The words are mapped to a continuous space by using a shared matrix, the projection matrix. The projection layer concatenates word vectors and it is fully-connected to the hidden layer. The hidden layer uses a non-linear activation function (the sigmoid function or *tanh*) and it is fully-connected to the output layer. The output layer, outputs a probability estimate for every word in the vocabulary by using a softmax function. The main complexity comes from the connections between the hidden layer and the output layer. Therefore, the vocabulary size plays a crucial rule in the complexity of NNLMs.

The complexity of NNLMs can be handled by using a shortlist of words [115]. In this approach, a shortlist is created from the most frequent

words and NNLMs are trained only for this shortlist. During prediction, NNLMs are used only for predicting probabilities of the words that are in the shortlist, the probabilities for the other words are estimated by using n-gram models. The hierarchical NNLM [101, 100] adopts a binary clustering of the words at the output layer to reduce the computational complexity. Structured output layer NNLMs [83, 84] use another tree representation at the output layer. In this approach, all words except a shortlist of words are clustered based on the distributed representations learned at the projection layer. A class-based output layer is presented in [97]. In this approach, word probabilities are factorized into class membership probabilities and class probabilities. The clustering is done based on the unigram frequencies of the words. All clustering techniques degrade the performance of NNLMs, however, for large vocabularies they make NNLMs feasible. In this thesis, we use the class-based output approach.

Feed-forward NNLMs are based on fixed histories, therefore they also suffer from the problems related to fixed histories. Recurrent NNLMs (RNNLMs) [96, 93] overcome this problem by using recurrent connections, which represent the state of the network. This can be thought of as a short-term memory, which in theory, enables the network to model infinite size of histories. RNNLMs are shown to improve perplexities and WERs better than any feed-forward NNLM [94]. In [92], a feature layer is added to RNNLMs, where topic features are used as additional context to the NNLM. In addition, Mikolov et al. [91] present the training of maximum entropy features jointly with the RNNLM. The maximum entropy features are over n-grams and they are implemented with a hash-based implementation, where n-grams are clustered based on hash functions. The RNNLMs that are trained jointly with maximum entropy models are referred to as RNNME.

Learning long-range dependencies with gradient descent is not possible

because the gradients of errors become smaller and smaller; this is known as the vanishing gradient problem [13]. Mikolov et al. [95] address the problem of vanishing gradients by comparing *Long Short Term Memory* (LSTM) networks [62] with *structurally constraint recurrent networks* (SCRNs) that have a hidden layer that is designed to capture long term dependencies. Authors report that the performance of the two architectures are similar and they outperform RNNLMs.

3.8 Combining LMs

The most widely used method for combining LMs is the linear interpolation method given by Equation 3.11.

$$p(w|h) = \sum_i \lambda_i p_i(w|h) \tag{3.11}$$

The mixture weights λ_i sum to 1 so that it gives a probability distribution. The weights are adjusted over a development data by minimizing the overall perplexity either by using the expectation maximization algorithm or empirically over WER.

Log-linear interpolation can also be applied for combining LMs. Log-linear interpolation has the following form given in Equation 3.12 [73].

$$p(w|h) = \frac{1}{Z_\lambda(h)} \prod_i p_i(w|h)^{\lambda_i} \tag{3.12}$$

Log-linear interpolation does not impose any explicit constraint over the mixture weights, however, the normalization factor $Z_\lambda(h)$ is needed to make the interpolation a probability distribution. This factor needs to be computed for every possible history. Although log-linear interpolation is reported to perform better than linear interpolation, since it requires normalization for every history, it is more complex.

In this thesis, we apply linear interpolation to combine multiple neural network language models.

3.9 Evaluation of LMs

LMs are evaluated by using two different metrics. The first one is the measure of how well it captures the unseen data, *perplexity*. Perplexity is closely related to cross entropy from information theory. The second one measures how well LMs perform in ASR tasks, word error rate (WER), which is the basic evaluation metric for ASR systems.

Perplexity (PPL) is calculated on an evaluation corpus of N words as in Equation 3.13 [52]. Perplexity is proportional to the cross entropy[2] of the evaluation corpus given the model. Therefore, the lower the perplexity is, the better the LM models the evaluation data.

$$PPL = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}} \tag{3.13}$$

As described in Chapter 2, WER is calculated by first aligning the ASR hypothesis with the reference transcription and then dividing the errors made (insertions, deletions, and substitutions) to the number of tokens in the reference transcription.

Both metrics have advantages and disadvantages. Computing perplexity is easy and it correlates with the system performance, however, may mislead the comparison between two models if one relies on future information. WER, on the other hand, requires a full ASR system and does not discriminate between type of errors. However, it gives an idea about the actual performance of the LM for the specific task at hand.

[2]Cross entropy is log_2 of perplexity.

Chapter 4

Spoken Language Understanding

"The noblest pleasure is the joy of understanding."
Leonardo da Vinci

Language is far more than strings of words. As Fillmore [46] points out, when speakers want to greet someone, they consider the utterance "Good morning, sir." as one of the options to greet the addressee. Speakers also know that this utterance is appropriate when the addressee is an adult male and only during a certain time in the day. Therefore, utterances have several complexities beyond being composed of lexical units; on one hand they have the function they serve, greeting in this case, and, on the other hand, they have their appropriate context.

Language understanding, in addition to the analysis of compositional meaning of lexical units, deals with finding out the function and the context of an utterance. *Semantic-frame* based understanding addresses the function and the context of an utterance based on the notion of *frames*.

Spoken language understanding (SLU) is most often performed by using a *semantic-frame* approach [130]. Its goal is to correctly identify the

semantic *frames* in the utterance and extract the values for the *slots* associated to the frames.

The notion of a "frame" [99] is proposed as a data structure to represent knowledge with a stereo-typed situation. In this framework, upon encountering a situation, *one* selects an appropriate frame from the memory and changes the details if necessary. The frame is represented as a network of nodes and relations. The frame has top levels which are always fixed and low levels which have "slots" that need to be filled by an instance of a situation. The theory of frame semantics is based on the notion of frames and consider semantics as the set of relations between linguistic forms and their meanings [47]. For instance, in a situation where there is a *commercial event*, i.e., when the *commercial event* frame is evoked, this frame contains roles or slots like the *buyer*, the *seller*, the *goods*, and the *money*. The linguistic units that evoke these frames are called *lexical units*, *target words* or *targets* which can be one of the following words, "buy", "sell", or "charge" [46]. FrameNet [48] is a project that extracts relations about the semantic and syntactic properties of English words from large text by means of manual semantic annotations and automatic processing. FrameNet project constructs a relational network of frames, and includes the list of lexical units and the frames they evoke with respect to their senses. The following example is taken from the FrameNet project that demonstrates the frame of "Commerce Scenario":

My local grocery store raised **prices** on *meat*

In this example, the lexical unit or the target word **prices** evokes the frame "Commerce Scenario". In this case, two slots of this frame "Seller" and "Goods" are filled with the phrases "*My local grocery store*" and "*meat*" respectively.

This chapter presents *semantic-frame* based SLU in detail. In addition,

an overview of frame-semantic parsing is given that is required by the semantic models which are described later in the thesis.

4.1 Spoken Language Understating

Spoken language understanding is the problem of extracting meaning representations from user utterances [129, 34]. The application domains of SLU are spoken dialog systems, speech information retrieval, and speech translation [34]. SLU and natural language understanding (NLU) is closely related. The focus of NLU is extracting meaning representations from a generic domain written text. On the other hand, SLU focuses on application specific domains and works on speech signals. The nature of speech brings additional difficulties that are not present in NLU. These difficulties are defined as: [130]

- The syntactic structure of spoken utterances are not well-formed as written-text.
- Disfluencies (hesitations, corrections, or repetitions) occur in speech.
- SLU relies on the ASR output, and ASR errors are propagated to SLU. Therefore, SLU systems must be trained to be robust to this noise.
- Out-of-domain utterance cannot be modeled well.

The research on SLU is started to gain interest with the Air Travel Information System (ATIS) evaluations [106]. ATIS evaluations resulted in systems that process spontaneous speech queries for air travel information and that bring back the relevant information from a database. A typical ATIS system is shown in Figure 4.1 [129].

This section focuses on the structure of the semantic representations and on the evaluation of SLU systems.

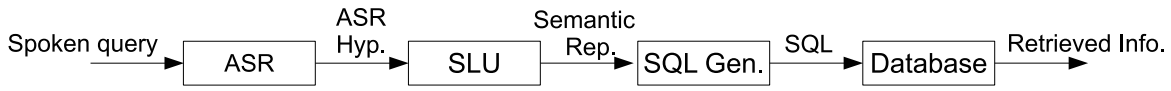

Figure 4.1: A typical ATIS system uses a cascaded approach. The SLU uses the ASR hypothesis and produces a semantic representation. This semantic representation is converted to a SQL query to retrieve the relevant information from the database.

4.1.1 Semantic Representation

In *semantic-frame* based SLU, semantic information is represented by *semantic frames*. Semantic frames are similar to the frames in the theory of frame semantics and they contain typed *slots* that need to be filled with respect to the semantic information. Figure 4.2 gives an example of three frames taken from the ATIS domain [129].

```
<frame name="ShowFlight" type="Void">
    <slot name="subject" type="Subject"/>
    <slot name="flight" type="Flight"/>
</frame>
<frame name="GroundTrans" type="Void">
    <slot name="city" type="City"/>
    <slot name="type" type="TransType"/>
</frame>
<frame name="Flight" type="Flight">
    <slot name="DCity" type="City"/>
    <slot name="ACity" type="City"/>
    <slot name="DDate" type="Date"/>
</frame>
```

Figure 4.2: Three semantic frames from the ATIS domain. The frames and the database are designed together so that they represent the same information. For example, the flight frame has slots "DCity" which represents the destination city, "ACity" which represents the arrival city, and "DDate" which represents the date of departure. The frames are simplified in this example.

Semantic frames are designed together with the database regarding the slots they contain. Therefore, a flight would contain destination and arrival

cities and also the departure time. An instantiation of the flight frame for the user query "Show me flights from Seattle to Boston." is given in Figure 4.3 [129].

```
<frame name="Flight" type="Flight">
    <DCity type="City"> Seattle </DCity>
    <ACity type="City"> Boston </ACity>
</frame>
```

Figure 4.3: An instantiation of the "flight" frame with for the user query "Show me flights from Seattle to Boston.". The departure and arrival city slots are filled respectively.

Semantic frames use a hierarchical approach to represent meaning representations. Another representation scheme is a simplified *attribute-value pair* representation or a *keyword pair* representation [104]. Attribute-value pairs use a flat representation scheme and do not possess a hierarchical structure. For the same user query, "Show me flights from Seattle to Boston.", we may have the following attribute-value pair representation in the ATIS domain: "(Command DISPLAY) (Subject FLIGHT) (DCity SEA) (ACity BOS). In addition, if an attribute spans multiple tokens of values, in/out/begin (IOB) representation can be used. For instance, in "Show me flights from Seattle to New York" the arrival city "New York" is represented with the following attribute-pairs "(ACity-B New) (ACity-I York)".

The representations presented for *semantic-frame* based SLU also have an impact on the definition of the SLU problem. In this respect, SLU may be defined as recognizing the correct frames and filling the slots in the semantic frame representation. On the other hand, it may also be defined as the alignment between the attributes and values. Considering these, SLU is also referred to as "slot filling".

4.1.2 Evaluation Metrics

The evaluation of SLU is performed on the semantic representation it outputs. The common evaluation metrics are [130]:

- **Concept Error Rate (CER):** Also known as slot error rate (SER). It measures the concept/slot performance of the SLU system. SLU hypothesis is aligned with the reference semantic annotation. Insertion (I), deletion (D), and substitution (S) errors are determined and CER is computed by Equation 4.1, where N represents the number of concepts/slots in the reference annotation.

$$CER = \frac{I + D + S}{N} \tag{4.1}$$

- **Slot Precision/Recall/F1 Score:** Precision/Recall/F1 scores are also used to assess the slot performance of the SLU system. They are calculated as follows:

$$Precision = \frac{\text{Number of correctly recognized slots}}{\text{Number of total slots recognized by the system}} \tag{4.2}$$

$$Recall = \frac{\text{Number of correctly recognized slots}}{\text{Number of total slots in the reference annotation}} \tag{4.3}$$

$$F1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{4.4}$$

4.2 Data Driven Approaches to SLU

SLU approaches are categorized into two; knowledge-based approaches and data driven approaches. Knowledge-based approaches use hand-crafted grammars, which are not easy to tune and scale. Data driven approaches, on the other hand, learn SLU models from semantically annotated training examples [130]. This thesis focuses on data driven approaches, therefore, we do not present knowledge-based approaches.

In a statistical framework, the semantic frame-based SLU problem is formalized as finding the most likely semantic representation, $\hat{C}$, given a sequence of words, W. It corresponds to finding the semantic representation that maximizes the probability $P(C|W)$.

4.2.1 Generative Models

Generative models, maximize the joint probability $P(W, C)$ by using the training data. They are described by Equation 4.5 [98, 130, 53].

$$\hat{C} = \arg\max_{C} P(C|W) = \arg\max_{C} P(W|C)P(C) \tag{4.5}$$

Therefore, generative models use two different models; the semantic prior model $P(C)$ which assigns a probability to semantic representations, and lexical realization model $P(W|C)$ which assigns a probability to word sequences given a semantic representation. A simple generative model can be implemented by hidden Markov models (HMMs), where states bear semantic meanings, and emissions from the states model the lexical realizations. Therefore, state transition probabilities model the semantic prior model and emission probabilities model the lexical realization model [130].

The SLU problem may also be modeled as a sequence labeling problem when attribute-value pair representation is used [107]. In this setting, each word is assigned a concept or a slot. The *null* concept/slot is assigned to

words which are not assigned to any slots. Therefore, the joint probability of a sequence of words, $W = \{w_1, \cdots, w_n\}$ and a sequence of concepts $C = \{c_1, \cdots, c_n\}$ are modeled by the joint probability $P(W, C)$, which is computed as in Equation 4.6.

$$P(W, C) = \prod_{i=1}^{k} P(w_i c_i | w_{i-1} c_{i-1}, \cdots, w_1 c_1) \tag{4.6}$$

Stochastic finite state transducers (SFSTs) are used as a generative SLU model [107]. In this framework, the transducer that models SLU takes words as input and outputs the concept tags. The SLU transducer, λ_{SLU} is composed of three other SFSTs. λ_W is used for representing the input, λ_{w2c} models all word to concept mappings that are hand-crafted or learned from a training corpus, λ_{SLM} is a language model over the concepts represented as a SFST. The SLU model, λ_{SLU}, which is the composition of the three SFSTs is given in Equation 4.7.

$$\lambda_{SLU} = \lambda_W \circ \lambda_{w2c} \circ \lambda_{SLM} \tag{4.7}$$

4.2.2 Discriminative Models

Discriminative approaches that are used for sequence labeling, model SLU by directly the conditional probability $P(C|W)$ [130, 107, 128, 53]. Conditional random fields (CRFs) [81] are reported to be more robust when the training data size is small compared to other discriminative models like perceptron [128]. CRFs use a log-linear model with feature functions that are defined on the observation sequence (words) and the label sequence (concepts). The formulation of SLU models with CRFs are given in Equation 4.8 [130].

$$P(C|W,\Lambda) = \frac{exp(\sum_k \lambda_k f_k(C,W))}{Z(W,\Lambda)} \tag{4.8}$$

$f_k(C,W)$ denotes the kth feature function that is defined over the sequences of words and concepts. $\Lambda = \{\lambda_1 ..., \lambda_k ..., \lambda_n\}$ is a set of parameters that are learned during training and $Z(W,\Lambda)$ is the partition function that normalizes the probability distribution. Training CRFs are very inefficient because feature functions are defined over the entire label sequence. Features functions can be constrained to be defined on the immediate states which makes training and inference more efficient [128].

Feature functions for CRF SLU models are typically n-gram features to capture the relationship between the label and the current and previously observed words, transition features which model the current label with the predicted previous label, and class member features which models the clustering for the words [130]. Trigger pairs are introduced as features to handle long-range dependencies in [69]. They are similar to the trigger pairs introduced in [82].

4.2.3 Neural Network Models

The continuous space neural network models are also applied to SLU. Deep convex networks (DCN) [37], which stack classifiers on top of each other to learn complex functions of classifiers, are applied to SLU in [36]. DCNs perform slightly better than linear CRFs and the combination of two models perform better than both models. Recurrent neural networks (RNNs) are applied to SLU in [90, 133]. The Elman architecture [43], in which previous hidden layer activations are used for recurrent connections, and the Jordan architecture [70] in which output posterior probabilities are used for recurrent connections are investigated in [90]. Mesnil et al. [90] report inconclusive results for the comparison of CRFs and RNNs; for ATIS,

RNNs outperform CRFs, however, for another dataset CRFs outperform RNNs. Yao et al. [133] build models using word embeddings and named entity features; the authors report that RNNs outperform CRFs on the ATIS dataset. In [132], long short-term memory (LSTM) [62] models are used for SLU modeling. LSTM models use a memory cell to store information and they are reported to model long-range dependencies better than RNNs [132]. Deep belief networks are applied to SLU in [39], and convolutional neural networks are jointly trained with CRFs for slot filling [131].

As can be seen, with the increased interest on neural networks, SLU has started to be approached by different architectures of neural networks. The use of distributed representations of words and concepts provide leverage also for the SLU problem.

4.2.4 Using Multiple Hypotheses

SLU systems may also make use of multiple hypotheses generated by the ASR. Hakkani-Tür et al. [54] present an algorithm to construct word confusion networks (WCNs) from ASR lattices and use WCNs for named-entity recognition. Deoras et al. [38] present a joint decoding algorithm, for jointly performing speech recognition and slot filling. The authors report that this approach outperforms CRFs. WCNs are used to train CRFs for slot-filling in [123]. In this approach, rather than training with manual annotations, WCNs that are obtained from ASR lattices are used to train CRFs; in this way SLU models are built to be more robust to ASR noise. A cache neural network architecture that is based on word-concept units is proposed for handling long-range dependencies [136]. The cache neural network is used for re-scoring n-best list that ASR outputs. A re-ranking approach that first generates multiple hypotheses by using a generative model and then uses support vector machines (SVMs) with tree kernels is proposed in [40].

In this thesis, we focus on re-scoring multiple hypotheses by using RNNs.

In that respect, we use n-best ASR hypotheses and feed it into the baseline SLU model and obtain the multiple SLU hypotheses. Then, we re-score these SLU hypotheses by using RNN models.

4.3 Frame-Semantic Parsing

Frame-semantic parsing is the problem of automatically assigning the frame-semantic structure to natural language sentences. Frame-semantic parsing has been most often applied to written text and in generic domains (e.g. news text). Statistical domain independent frame-semantic parsing is started with the task of "semantic role labeling" in which the semantic roles are defined with frame elements of semantic frames [50]. The CoNLL shared tasks [23] on semantic role labeling also contributed to this research.

Although, semantic role labeling deals only with filling frame elements a full pipeline for frame-semantic parsing needs to handle all of the following subtasks [32]:

- **Target Recognition:** This subtask decides which words evoke semantic frames in a sentence.
- **Frame Identification:** This task identifies the correct frame that a target evokes. Since, a word can evoke more than one frame with respect to the context, this is a multi-class classification task given the frame evoking word.
- **Frame Element (Argument) Detection:** This task detects which frame elements are filled in the sentence and finds the correct span of the frame elements over the phrases.

The reader may refer to [32] for a more detailed overview of frame-semantic parsing. In this thesis, we have used frame-semantic parsing as a black box to extract features for semantic language models. We use

the state-of-art frame-semantic parser, SEMAFOR [32], for English. SEMAFOR relies on the output of a statistical dependency parser. The target recognition is performed with a rule-based system. And for frame identification and frame element detection, exponential models are used. Italian LUNA frame-semantic parser [29], which we have used for frame-semantic parsing in Italian, has a rule-based target recognition model and a simple statistical frame identification model. The Italian LUNA semantic parser mostly focuses on frame element detection by using SVMs. The parser relies on the output of a statistical constituency parser.

Chapter 5

Neural Network Models

This chapter gives a detailed explanation of the neural network architectures used in this thesis. The first section introduces the related terminology of neural networks on single layer networks. In Section 5.2, feed forward neural network LMs are presented. Section 5.3 continues with recurrent neural network LMs. Finally, deep autoencoders that are used for semantic feature extraction are described in Section 5.4.

5.1 Single Layer Networks

This section presents the building blocks of multi-layer neural networks and introduces the related terminology. The reader may refer to [19] for a detailed explanation of the concepts, from which this section is compiled.

A classification problem on two categories may be performed by using a discriminant function, $y(\mathbf{x})$ such that the value of the function determines which class the m dimensional vector $\mathbf{x}$ belongs to. A decision rule assigns $y(\mathbf{x})$ to class C_1 if $y(\mathbf{x}) > 0$ and to class C_2 if $y(\mathbf{x}) < 0$. A simple discriminative function is given in Equation 5.1.

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \tag{5.1}$$

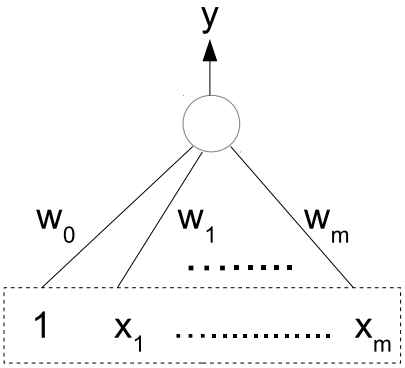

Figure 5.1: Neural network diagram of the linear discriminant function. The bias w_0 is represented as an additional weight with an input value of 1.

The m dimensional vector $\mathbf{w}$ is referred to as the *weight* vector and the scalar w_0 is referred to as the *bias*. The decision boundary is given by $y(\mathbf{x}) = 0$ and corresponds to a $(m-1)$ dimensional hyperplane in m dimensional space. Since this decision boundary is linear, it is appropriate for the classification problems that are linearly separable. The classification that is performed with such a discriminative function is referred to as linear classification.

The linear discriminant function in Equation 5.1 can be represented by a network diagram as given in Figure 5.1. This figure represents the weight vector as the concatenation of the bias w_0 with $\mathbf{w}$ and the input vector as the concatenation of 1 with $\mathbf{x}$.

5.1.1 Multi-Class Classification

The classification can be extended to c classes by using a linear discriminant function $y_k(\mathbf{x})$ for each class C_k. Then the decision is made by assigning the input to C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. The linear discriminant function is given in Equation 5.2 and the corresponding network, which uses c *nodes* for c classes, is shown in Figure 5.2.

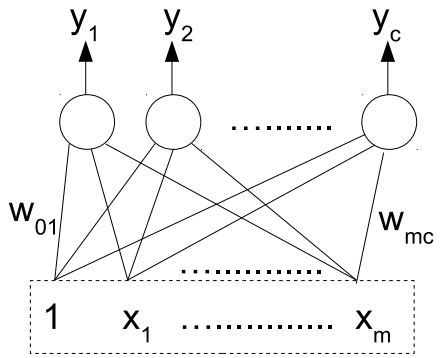

Figure 5.2: Neural network diagram for multi-class classification

$$y_k(\mathbf{x}) = \sum_{i=0}^{m} w_{ki} x_i \tag{5.2}$$

The weights of the network can be learned by minimizing an error function over training data of size N, $\mathbf{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, where each element, $\mathbf{x}^i$ is a vector of size m. The target value at node k for the input $\mathbf{x}^i$ is given by $\mathbf{t}_k^i$. Also we represent the output of node k with the weights $\mathbf{w}$ for the input $\mathbf{x}^i$ as $y_k(\mathbf{x}^i; \mathbf{w})$. Then the sum-of-squares error function that is defined over the parameters of the network, is written as in Equation 5.3.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{c} \{y_k(\mathbf{x}^n; \mathbf{w}) - \mathbf{t}_k^n\}^2 \tag{5.3}$$

Then the optimal values of weights can be determined by *gradient descent*, i.e., by iteratively going over the training data and updating the weights as given in Equation 5.4 for the time step $\tau + 1$. w_{kj}^0 denotes the initial weights.

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \bigg|_{w_{kj}^{(\tau)}} \tag{5.4}$$

The parameter η is called the *learning rate* and usually determined empir-

ically.

5.1.2 Activation Function

The discriminant function given in 5.2 is a linear function of the input, $\mathbf{x}$. A monotonic non-linear function $g(.)$ may also be used. In this case, the function $g(.)$ is called the *activation function*. A discriminant function that uses an activation function can be given as in Equation 5.5. The decision boundary generated by such a discriminant function is still linear.

$$y_k(\mathbf{x}) = g(a_k), \text{ where } a_k = \sum_{i=0}^{m} w_{ki} x_i \tag{5.5}$$

The *sigmoid* activation in Equation 5.6 is one of the most widely used non-linear functions because of its nice derivative, and it maps the interval $(\infty, -\infty)$ onto interval $(0, 1)$.

If we want to model the conditional probability distribution $P(C_k|\mathbf{x})$ that estimates the probability of input $\mathbf{x}$ belonging to class C_k, the *softmax* function that is given in Equation 5.7 can be used.

$$g(a_k) = \frac{1}{1 + e^{-a_k}} \tag{5.6}$$

$$g(a_k) = \frac{e^{a_k}}{\sum_{k'=1}^{c} e^{a_{k'}}} \tag{5.7}$$

5.2 Feed Forward Neural Network LMs

Feed forward neural network LMs (FFLMs) were first introduced in [14]. The main contribution of FFLMs is their ability to learn the distributed representation of words (they represent words as vectors in a continuous space). FFLMs estimate LM probabilities over this continuous space. The

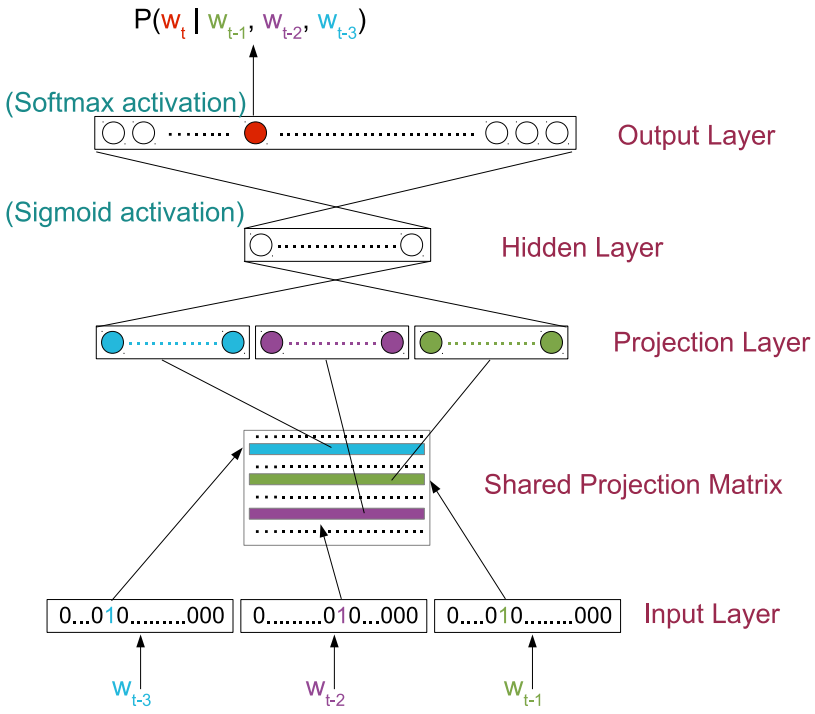

Figure 5.3: Feed forward neural network language model architecture. The input layer takes the history by 1-of-n encoding of words. The projection layer, project the word vectors onto a continuous space by using a shared projection matrix. The output layer estimates the probability of the next word. The projection layer is also fully connected to the output layer, however, it is not shown in the illustration.

structure of an n-gram FFLM is given in Figure 5.3. FFLMs usually are composed of four layers.

The input layer represents the words with 1-of-n encoding. In this representation each word is represented by a vector where only the index of the designated word is set to 1 and the others are set to 0. Therefore, the dimension of the input word vector is equal to the vocabulary size, $|\mathcal{V}|$. The size of the input layer is also proportional to the size of the history, $n-1$ for an n-gram model. The projection layer, is the layer where word representations are projected onto a continuous space in $\mathbb{R}^m$. The size of the projection layer is determined by the projection dimension, m, and the history size, $n-1$. The projection of the words onto the continuous space

is performed by using a shared matrix, $\mathcal{C}$. $\mathcal{C}$ is a $|\mathcal{V}| \times m$ matrix, where each row represents a feature vector for each word, hence $\mathcal{C}(i) \in \mathbb{R}^m$ represents the feature vector for the word with index i. In the actual FFLM implementation, the input layer is omitted and the ith row of the shared matrix is copied to the projection layer for the word that has index i. Therefore, the input layer is just presented in the conceptual model but not in the actual implementation.

The projection layer is fully connected to the hidden layer that has a dimension h by a $h \times (n-1)m$ weight matrix $\mathcal{H}$. The hidden layer has a non-linear activation and the possible choices are $tanh$ function or the sigmoid function. Bengio et al. [14] use $tanh$ as the non-linear activation function, however, in all the experiments that are conducted in this thesis the sigmoid function is used.

The output layer of FFLMs is a probability distribution, hence, it outputs a probability estimate for every word in the vocabulary $\mathcal{V}$. Therefore, its size is equal to the vocabulary size, $|\mathcal{V}|$. The output layer is fully connected to the hidden layer by a $|\mathcal{V}| \times h$ weight matrix $\mathcal{U}$ and also to the projection layer[1] by a $|\mathcal{V}| \times (n-1)m$ weight matrix $\mathcal{W}$. The softmax activation function is used to output a probability distribution at the output layer. The FFLM n-gram model can be written as in Equation 5.8, where a_i shows the total input to the output unit at index i.

$$P(w_t|w_{t-n+1}, \cdots, w_{t-2}, w_{t-1}) = \frac{e^{a_k}}{\sum_{i=0}^{|\mathcal{V}|} e^{a_i}} \quad , k = index\ of\ w_t \tag{5.8}$$

The total input for all the nodes at the output layer, $\mathbf{a}$, is computed as in Equation 5.9, where $\mathbf{b}$ refers to the biases of the output layer, $\mathcal{W}$ refers to the weight matrix from the projection layer to the output layer, $\mathcal{U}$ refers to

[1]For the sake of simplicity the connections from the projection layer to the output layer are not shown in Figure 5.3

the weight matrix from the hidden layer to the output layer, $\mathbf{d}$ refers to the biases of the hidden layer, $\mathcal{H}$ refers to the weight matrix from the projection layer to the hidden layer, and $\mathbf{x}$ refers to the activations at the projection layer. These activations are composed by concatenating the feature vectors of the words in the history, i.e., $< \mathcal{C}(w_{t-n+1}), \ldots, \mathcal{C}(w_{t-2}), \mathcal{C}(w_{t-1}) >$.

$$\mathbf{a} = \mathbf{b} + \mathcal{W}\mathbf{x} + \mathcal{U}sigmoid(\mathbf{d} + \mathcal{H}\mathbf{x}) \tag{5.9}$$

5.2.1 Training of FFLMs

The training of FFLMs is performed by minimizing the cross-entropy error function, E, over the training data by using the *backpropagation algorithm* [112]. For instance, when training a 4-gram FFLM, for each 4-gram $(w_{i-3}, w_{i-2}, w_{i-1}, w_i)$ in the training data, the history $(w_{i-3}, w_{i-2}, w_{i-1})$ is given as input and the probability estimate $P(w_i|w_{i-3}, w_{i-2}, w_{i-1})$ is obtained at the output layer by using the softmax function. Then the cross-entropy for this training instance is calculated by using Equation 5.10 [19], where the target, $\mathbf{t}$ (for word w_i) is represented by 1-of-n encoding[2], i.e., t_k is 1 if $k = i$ and 0 otherwise. The n given in the equation refers to the *nth* instance of the training data.

$$E^n = -\sum_{k=1}^{\mathcal{V}} t_k y_k \tag{5.10}$$

The error at the output node k (denoted as δ_k^o), which is equal to the gradient of this error function with respect to the total input (a_k) at the output unit k, is given in Equation 5.11

$$\delta_k^o = \frac{\partial E^n}{\partial a_k} = y_k - t_k \tag{5.11}$$

[2]The target is a binary vector, in which the component at index i is set to 1 and the rest is set to 0 for the i^{th} word w_i.

Backpropagation of Errors

The backpropagation algorithm propagates the error at a layer $l+1$ (δ_k^{l+1}) to a lower layer that is connected to this layer by the weights $\mathbf{w}$ using the general backpropagation formula [19] that is given in Equation 5.12, where K refers to the size of the layer $l+1$, j refers to the jth unit at layer l, and $g'(.)$ refers to the gradient of the activation function.

$$\delta_j^l = g'(a_j) \sum_{k=1}^{K} w_{kj} \delta_k^{l+1} \tag{5.12}$$

The weights between the node j at layer l and the node i at layer $l-1$ are updated by using the following formula, where η denotes the *learning rate*:

$$\Delta_{ji} = -\eta \delta_j g(a_i) \tag{5.13}$$

The weight update rule for biases is similar to Equation 5.13, where the activation function for the node at the lower layer, $g(.)$ is replaced with 1.

Using this algorithm all the errors are computed at each layer and the corresponding weights are updated. The shared projection matrix is updated by just updating the row that corresponds to the given word in the history.

Practical Issues

The training of neural network LMs are performed by using stochastic gradient descent, i.e., the parameters of the network is updated for each training instance one by one.

The neural network models may overfit the training data, therefore to avoid overfitting the training procedure must be performed by considering the performance of the network on a held-out data or development data.

Therefore after each epoch of training (one pass over the whole training data), the cross entropy of the model over the development data is calculated and the learning rate is decreased if the cross entropy increases or the training stops if the improvement of the cross entropy is less than a threshold (*early stopping*).

5.3 Recurrent Neural Network LMs

Recurrent neural network LMs (RNNLMs) were introduced in [96]. RNNLMs are based on the Elman architecture given in [43]. The basic idea of RNNLMs is to save the state of the network by recurrent connections, which aim at handling unlimited size of histories. RNNLMs predict the probability of a word by using the immediate history (the previous word) and the hidden layer activations at the previous time step. The structure of RNNLMs is given in Figure 5.4. Therefore, RNNLMs model the probability of a word w_t given the history as in Equation 5.14, where s_t refers to the hidden layer activation at time step t.

$$P(w_t|history) = P(w_t|w_{t-1}, s_{t-1}) \tag{5.14}$$

5.3.1 Training RNNLMs

The training of RNNLMs is also done by performing error backpropagation. However, because of the recurrent architecture, the network must be unrolled in time. Figure 5.5 shows the unrolled network for 3 time steps back. After unrolling the network, the standard backpropagation algorithm is applied. This is also known as the *backpropagation through time* (BPTT) algorithm [20].

Because of the vanishing gradients, the network is unrolled at most 4, 5

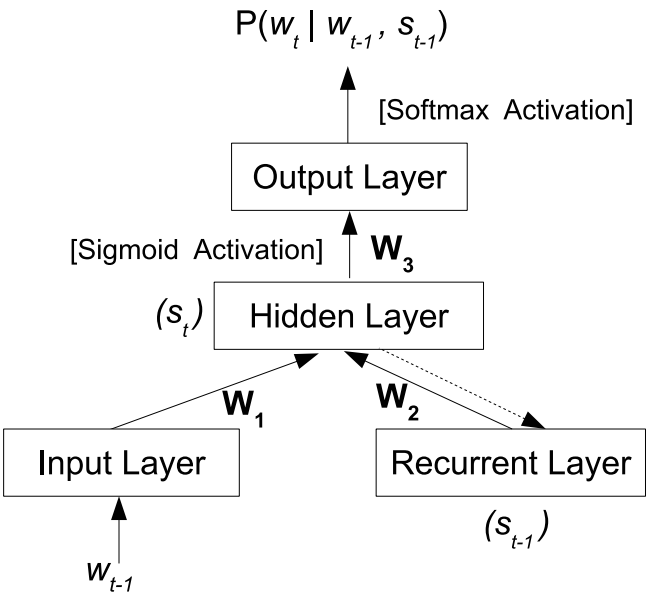

Figure 5.4: The basic recurrent neural network LM architecture. s_t represents the activation of the hidden layer at time step t.

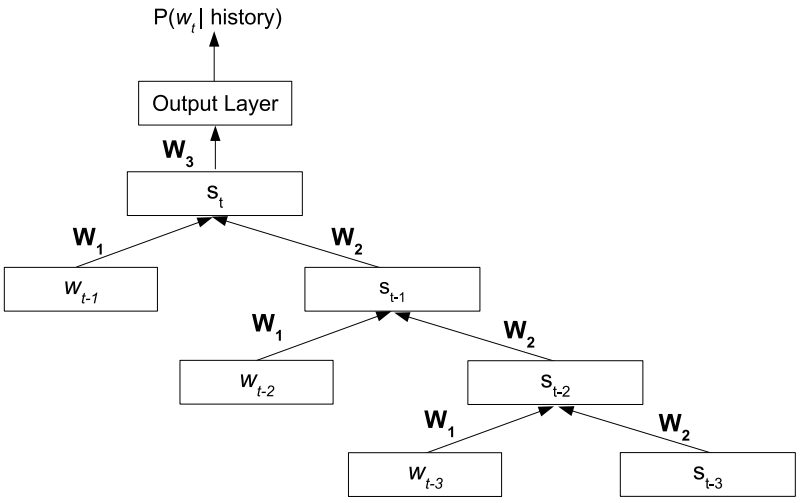

Figure 5.5: The backpropagation through time algorithm unrolls the network for N time steps back. In this figure, the network is unrolled for 3 time steps back.

time steps back practically.

5.3.2 Class-Based RNNLMs

The main complexity of neural network LMs comes from the size of the vocabulary. As the vocabulary size increases the size of the weight matrix between the hidden layer and the output layer becomes the dominant factor in the complexity of training. There are strategies like using a shortlist of words [115] or a hierarchical representation of words [101, 100, 83, 84] that reduce this complexity. In this thesis, we use the class-based RNNLM architecture that is introduced in [97]. The class-based approach divides the output layer into two, where one part computes the class probability, and the other part computes the class membership probability of the word. In this setting, the training is done by first updating the connections between the "class" part (that estimates the class probabilities) of the output layer and the hidden layer. Then, only the weights between words that belong to the given class and the hidden layer is updated. The factorization of the output is given in Equation 5.15, where cl_t denotes the class for word w_t, and s_t denotes activations of the hidden layer when predicting w_t.

$$P(w_t|w_{t-1}, s_{t-1}) = P(cl_t|w_{t-1}, s_{t-1})P(w_t|cl_t, w_{t-1}, s_{t-1}) \tag{5.15}$$

5.3.3 Maximum Entropy Features

The maximum entropy LMs are introduced in [110]. In [91], a joint training procedure for RNNLMs with n-gram maximum entropy features is introduced. The resulting architecture is referred to as RNNME. RNNMEs represent maximum entropy features as direct connections between the output layer and the nodes that represent n-gram histories. Therefore,

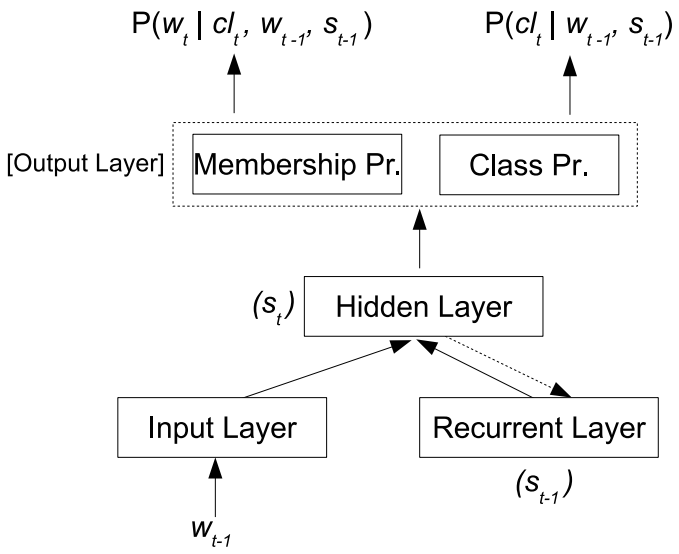

Figure 5.6: The class-based RNNLM. The output layer factorizes the conditional probability into class and class membership probabilities.

each possible n-gram is represented by a cluster of nodes and these nodes are fully connected to the output layer. If an n-gram in the history exists that node is activated with the value 1 and the weights between that node and the output layer are updated with the weight update formula.

The size of all possible n-gram histories could be very large for tri-grams and higher order n-grams when the vocabulary size is large, which makes building RNNME models impossible. To overcome this problem [91] uses a hash-based implementation, where each n-gram history is passed through a hash function and is mapped to the node in the maximum entropy feature layer. In this setting, a node may represent more than one n-gram history if there are collisions in the hash function. In addition, for class-based RNNLMs half of this layer is connected to the word nodes at the output layer and the other half is connected to the "class" part. The implementation details are given in [93].

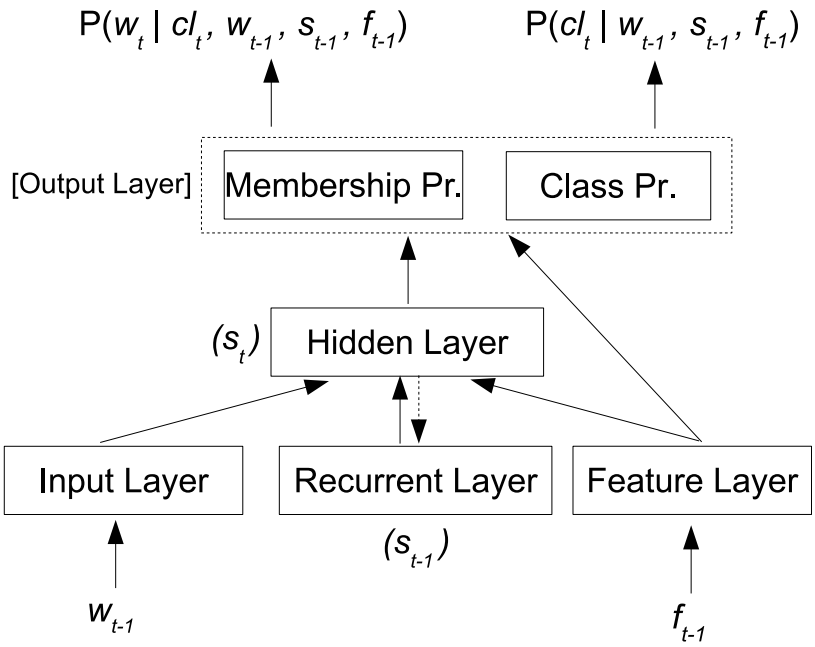

Figure 5.7: The context dependent RNNLM with a class-based implementation. f_{t-1} represents the features for the context of w_t.

5.3.4 Context-Dependent RNNLMs

Context dependent RNNLMs are introduced in [92], where an additional input layer (feature layer) is added to represent the additional context for the model. The proposed architecture is given in Figure 5.7. The feature layer is originally used to represent the topic for LM [92].

5.4 Deep Autoencoders

In general, deep learning aims at learning high-level representations of data that can be used in classification tasks [12]. One of the architectures that is used for representation learning is *autoencoders*. Autoencoders employ two functions; the first one is the *encoder* function that computes a feature vector from the input, and the second one is the *decoder* function that maps the feature back to the input space. The autoencoder is trained to minimize

the reconstruction error, i.e., the error between the input vector and the reconstructed image of the input [12].

Deep autoencoders that use multiple hidden layers are used to reduce the dimension of the data in [58] and they are reported to outperform other approaches like principal components analysis. They also outperform latent semantic analysis for document similarity tasks. In addition, an autoencoder that outputs binary codes are used for efficient document retrieval [113]. The training of deep autoencoders can be done by using gradient descent, however, with random initialization of weights they tend to converge to a poor local minima. To overcome this problem the weights must be initialized close to a good solution [58]. In this respect, the training of deep autoencoders are performed at two steps: the unsupervised pretraining phase, and the fine-tuning phase [58] as described in Figure 5.8.

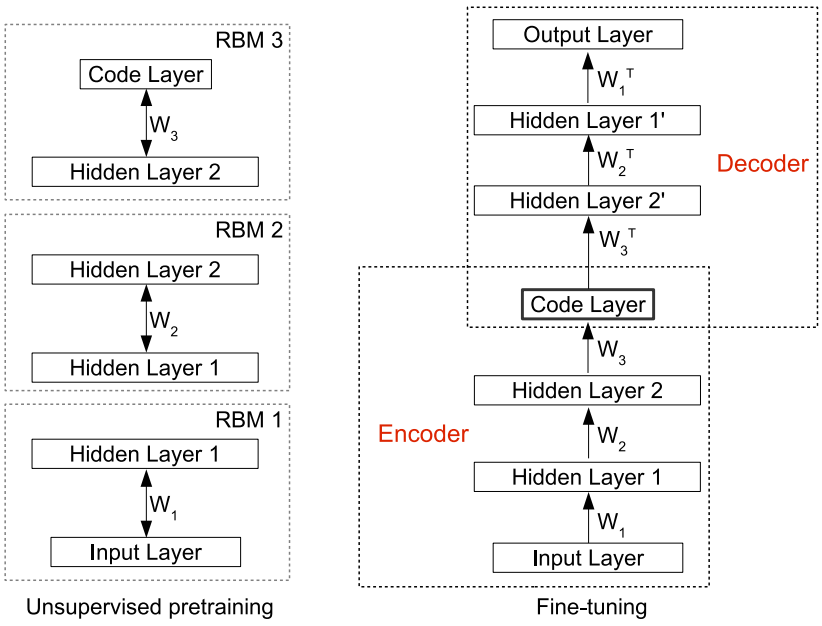

Figure 5.8: Deep autoencoder and training phases. The weights of the autoencoder are initialized with the unsupervised pretraining phase by using a greedy layer-by-layer approach. The weights are then fine-tuned on the unrolled network.

5.4.1 Unsupervised Pretraining

Unsupervised pretraining is done by using a greedy layer-by-layer approach that is given in [61]. In this approach, each pair of layers is represented by a restricted Boltzmann machine (RBM) and the weights of the RBMs are learned by contrastive divergence [60].

Restricted Boltzmann Machines

Restricted Boltzmann machines (RBMs) consist of two layers, a visible layer and a hidden layer, that are connected with each other using undirected connections [118, 60]. In addition, there are no connections between the nodes that are at the same layer. The structure of a RBM which has 3 nodes at the visible layer and 2 nodes at the hidden layer is given in Figure 5.9. The visible layer is the layer that is observed and the hidden layer is the layer that holds hidden representations of the observations.

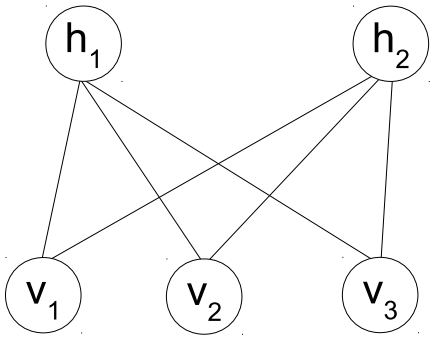

Figure 5.9: The structure of a RBM that has 3 nodes in the visible layer and 2 nodes at the hidden layer. The nodes are connected by using undirected weights.

The joint configuration $(\mathbf{v}, \mathbf{h})$ is represented by the energy of the network that is given in Equation 5.16, where v_i is the binary state of the ith visible node, h_j is the binary state of the jth hidden node, b_k is the bias of the kth unit and w_{ij} is the weight of the undirected connection between ith visible node and the jth visible node [59].

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in visible} b_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{5.16}$$

Every state of the network, i.e., every pair of visible and hidden vector is assigned a probability by Equation 5.17

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}, \text{ where } Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{5.17}$$

Then the probability of the network to assign the visible vector $\mathbf{v}$ is calculated by marginalizing Equation 5.17 over $\mathbf{h}$ as in Equation 5.18.

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{5.18}$$

The probability of the visible vector (the input) can be increased by lowering the energy for that input. The derivative of the probability function with respect to the weights is given in Equation 5.19, where $\langle v_i h_j \rangle_{data}$ denotes the expectation of $\{v_i, h_j\}$ pair with respect to the data, and $\langle v_i h_j \rangle_{model}$ denotes the expectation of $\{v_i, h_j\}$ pair with respect to the model.

$$\frac{\partial log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{5.19}$$

The expectation with respect to the data can be obtained easily by just feeding the input into the network through the visible layer and computing the probabilities of the hidden states[3] to turn on (having state 1) by using the sigmoid activation function, as given in Equation 5.20. In addition, the probability of a visible state to be 1 is computed similarly by Equation 5.21.

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + e^{-b_i - \sum_i v_i w_{ij}}} \tag{5.20}$$

[3]Hidden states are modeled by stochastic binary units.

$$P(v_i = 1|\mathbf{v}) = \frac{1}{1 + e^{-b_j - \sum_j h_j w_{ij}}} \tag{5.21}$$

The expectation with respect to the model can be approximated by performing alternating Gibbs sampling. Hinton [60] introduces *contrastive divergence* for approximating the gradient of the log probability. The approximate learning procedure works by: (1) obtaining the hidden states from the input, (2) reconstructing the image of the hidden states at the visible layer, (3) repeating this sampling procedure with this reconstructed image n times, as given in Figure 5.10. This is denoted as CD_n. RBMs learn better as n increases, however, even CD_1 works well practically, for a good initialization of weights.

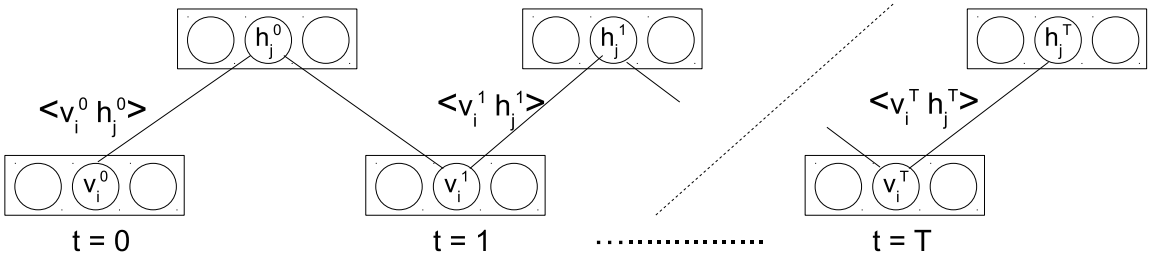

Figure 5.10: Contrastive divergence$_T$ (CD_T) procedure for learning the weights.

Contrastive Divergence

The CD_1 procedure is performed as follows when stochastic binary units are used:

- The probability of the hidden layers ($\mathbf{h^0}$) is computed by Equation 5.20 by using the training data ($\mathbf{v^0}$), which gives $\langle v_i^0 h_j^0 \rangle$.
- The states of the hidden units are determined by comparing the probabilities with random numbers that are distributed uniformly between

0 and 1. The state is 1 if the probability is greater than the random number, 0 otherwise.

- The probability of the visible units is computed by using the hidden states found in the previous step by Equation 5.21.
- The probabilities of the visible units ($\mathbf{v^1}$) are used to compute the probability of the hidden layers ($\mathbf{h^1}$) once more by Equation 5.20, which is used for $\langle v_i^1 h_j^1 \rangle$.
- The weights are updated by using the rule: $\Delta w_{ij} = \eta(\langle v_i^0 h_j^0 \rangle - \langle v_i^1 h_j^1 \rangle)$

5.4.2 Fine-tuning

The fine-tuning of the weights can be done by first unrolling the network as shown in Figure 5.8 and then by minimizing the reconstruction error at the output layer by using the backpropagation algorithm.

5.4.3 Semantic Hashing

Semantic hashing [113] encodes documents as binary vectors, where the similarity between two documents is measured by using the Hamming distance between these binary vectors. Semantic hashing is a deep autoencoder that is trained over the normalized bag-of-words (BoW) vectors of the documents. In addition, during fine-tuning Gaussian noise is added to the inputs of the code layer. In this way, the codes are enforced to be binary.

During the pretraining phase the bottom RBM is modeled by the constrained Poisson model [113] that is defined as follows. The activations at the visible and hidden layers are defined by the Equation 5.22 and 5.23 respectively, where n represents the BoW count for a specific words and N represents the total number of words in the document.

$$P(v_i = n|\mathbf{v}) = Ps(n, \frac{e^{b_i+\sum_j h_j w_{ij}}}{\sum_k e^{b_k+\sum_j h_j w_{kj}}} N) \tag{5.22}$$

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + e^{-b_j-\sum_i w_{ij} v_i}} \tag{5.23}$$

where $Ps(n, \lambda)$ defines the Poisson distribution:

$$Ps(n, \lambda) = e^{-\lambda} \frac{\lambda^n}{n!}$$

This model is equivalent to using normalized BoW vectors at the visible layer and multiplying the weights by N when the hidden layer probabilities are computed. Visible layer probabilities are computed by using the softmax activation function, which reconstructs the normalized BoW representations as given in Figure 5.11 [113].

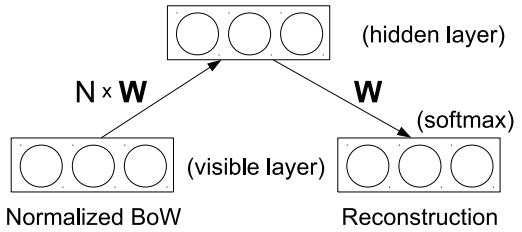

Figure 5.11: Constrained Poisson model. The input is represented as normalized BoW vectors. When activating the hidden layer the weights are multiplied by the words in the document (N). When the hidden state is used to activate the visible layer softmax activation function is used.

5.4.4 Deep Autoencoder for Semantic Context

In Chapter 9, we have used the autoencoder model that is described in semantic hashing [113] for encoding semantic context. Semantic hashing is designed to work on documents, i.e., written text, therefore it is appropriate to encode semantic information for utterances. We have trained "semantic hashing" autoencoders with semantic BoW features. The pretraining step

for encoding semantic context uses the same constrained Poisson model for the input layer. During fine-tuning the codes are made binary by using stochastic binary units at the the code layer. The state of a node at this layer is computed by comparing the activation of that node (the sigmoid activation function is used) with a random value between 0 and 1 that is generated at run time. The state is set to 0 if the activation is smaller than the random value and set to 1 otherwise. This state is used during the forward-pass, on the other hand, during backpropagation, actual activation values are used. In this manner, we enforce the autoencoder to output binary vectors.

Chapter 6

Data Description and Baselines

This chapter describes the data that is used in the experiments presented in the thesis. The data is analyzed and the related baselines in ASR and SLU are given. We have three different types of recorded speech:

- LUNA Human-Machine: Domain specific conversational speech between a human and a machine.
- LUNA Human-Human: Domain specific conversational speech between two humans.
- Wall Street Journal: Read speech of newspaper articles on a broad domain.

The Italian LUNA corpus is a collection of Italian conversational speech which consists of Human-Machine (HM) and Human-Human (HH) dialogs [41]. The conversations are recorded by a customer care and technical support center, and the domain of conversations is software/hardware troubleshooting. The dialogs in the LUNA corpus are annotated at a multi-level scheme. The levels of annotation are:

- *Word level annotation*: It consists of lemmas, part-of-speech tags and morpho-syntactic information following EAGLES corpora annotation. [85]

- *Attribute-value annotation*: It specifies concepts and their relations based on a predefined ontology.
- *Predicate argument annotation*: It is based on the FrameNet model [3].
- *Dialog act annotation*: It is inspired by DAMSL [30], TRAINS [121], and DIT++ [22] and marks the intentions in an utterance.

The Wall Street Journal (WSJ) speech recognition corpus contains the read WSJ articles, which is designed to be the first general purpose, large vocabulary speech recognition corpus [103]. The corpus is designed for speech recognition tasks, hence it is annotated only with the transcription of the utterances.

6.1 LUNA Human-Machine Corpus

The Italian LUNA Human-Machine (HM) corpus is a conversational speech corpus [41]. The corpus contains 723 human-machine dialogs in the hardware/software help desk domain. The dialogs are conversations of the users involved in a problem solving scenario collected by using the Wizard of Oz (WOZ) technique: the human agent (wizard) reacts to user requests and follows one of the ten scenarios identified as the most common scenario by the help desk service provider. Text-to-speech synthesis is used to provide responses to the users. The statistics about the level of annotation for the LUNA HM corpus is given in Table 6.1.

Table 6.1: The annotation level statistics for the LUNA HM corpus.

Annotation Level	**Number of Dialogs**
Word level	723
Attribute-value	723
Predicate argument structure (FrameNet)	129
Dialog act	224

In this thesis we work on the word level annotations for ASR and attribute-value annotations for SLU. The whole corpus is annotated at these levels. The statistics about the training, development, and evaluation splits are given in Table 6.2.

Table 6.2: The splits and the statistics of the LUNA HM corpus.

	Training		**Development**		**Evaluation**	
No. of Utterances	3171		387		634	
	Words	Concepts	Words	Concepts	Words	Concepts
No. of Tokens	30470	18408	3764	2258	6436	3783
Vocabulary Size	2386	44	777	38	1059	38
OOV rate	-	-	4.2%	0	3.7%	0

6.1.1 ASR Baseline

The Loquendo ASR system is used to obtain the ASR baseline for LUNA HM corpus. This system uses hybrid ANN/HMM acoustic models that are adapted to the LUNA HM corpus. It uses a tri-gram LM that is trained on the training split of the corpus with modified Kneser-Ney smoothing. The ASR performs finite state transducer decoding and outputs lattices, 100-best lists for re-scoring are compiled by using those lattices. The WER performance of the baseline system with the oracle performance[1] of the 100-best list is given in Table 6.3.

Table 6.3: The WER performance of the baseline ASR for the development (Dev) and the evaluation (Eval) splits of the LUNA HM corpus.

	Dev	**Eval**
ASR 1st-best	21.9%	22.3%
Oracle on 100-best	14.1%	15.6%

[1]The best possible performance that can be obtained from the n-best list.

6.1.2 SLU Baseline

We present two different SLU baselines. The first one is a generative model that is built by using stochastic finite state transducers (SFSTs) and the second one is a discriminative model that is constructed by conditional random fields (CRFs).

The SFST model creates a SFST for a statistical LM over word-concept pairs, a SFST for mapping words to word-concept pairs. Concept tagging is then done by feeding the input into the composition of these SFSTs.

The CRF model uses the following features. The first type of features is the orthographic feature. These features consider the first or last i letters of the word, where i changes between 1 and 5. Next, bi-gram features are used that are on the "previous word and current word", "current word and next word", and "previous word and next word". In addition to these features we have used binary features which label numerical expressions. We have also considered the value of the previous concept when predicting the current one. All these features are independent of each other in the window of [-1, +1].

The baseline SLU performances are given both for the reference transcription of the test set and for the ASR 1st-best hypothesis in Table 6.4.

Table 6.4: The CER performance of the SLU baselines on the reference transcription and the ASR hypothesis of the Test set.

Model	**Ref. Trans.**	**ASR 1st-best**
SFST	29.6%	46.3%
CRF	21.5%	26.7%

6.2 LUNA Human-Human Corpus

LUNA HH corpus consists of human-human dialogs that are recorded between a customer and an operator that supplies technical service. The

dialogs are recorded on a single channel. Because of the nature of human-human conversations these dialogs contain overlapped segments[2] which are discarded. The statistics on LUNA HH corpus is given in Table 6.5.

Table 6.5: The splits and the statistics of the LUNA HH corpus.

	Training	**Development**	**Evaluation**
No. of Utterances	14465	1656	4006
No. of Tokens	116178	11668	28476
Vocabulary Size	6840	1957	3269
OOV rate	-	3.8%	4.0%

6.2.1 ASR Baseline

The ASR baseline for LUNA HH is constructed by using the Kaldi [105] speech recognition toolkit. The ASR uses mel-frequency cepstral coefficients (MFCC) that are transformed by linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT). These features are then spliced in the window of $[-3, +3]$. The acoustic models are trained by advance training approaches such as "speaker adaptive training". The speaker adaptation during decoding is performed by feature-space maximum likelihood linear regression (fMLLR) [87]. The LM for the ASR is a modified Kneser-Ney tri-gram model that is built over the training data. The ASR and the oracle WER performances on the 100-best list is given in Table 8.8.

Table 6.6: The WER performance of the baseline ASR on LUNA HH corpus. The oracle performance is given for 100-best lists.

	Dev	**Test**
ASR	37.7%	36.7%
Oracle on 100-best	25.3%	24.4%

[2]Two speakers are speaking at the same time.

Table 6.7: The frame accuracy of the ASR with respect to the reference transcriptions. Density shows the ratio between the number of frames evoked and the number of tokens occur in the data.

	Development	**Evaluation**
Density	17.7%	14.9%
Precision	73.2%	76.1%
Recall	70.6%	73.2%
F1	71.9%	74.6%

Table 6.8: The target accuracy of the ASR with respect to the reference transcriptions.

	Development	**Evaluation**
Precision	73.1%	74.5%
Recall	74.3%	75.1%
F1	73.7%	74.8%

6.2.2 FrameNet Semantic Parsing

In this section, we present the performance of the ASR baseline on frame accuracy and target accuracy by using the Italian LUNA semantic-frame parser [29]. The evaluations are done by taking the output of the parser on the reference transcriptions as the gold standard. The Italian LUNA frame-semantic parser is trained with domain specific frames, and the frame vocabulary of the training data is 143 and the target vocabulary is 679. The frame accuracy is given in Table 6.7 with the frame densities, i.e., the ratio between the number of frames evoked and the number of tokens occur in the data. The target accuracy is given in Table 6.8.

6.3 Wall Street Journal Corpus

Wall Street Journal (WSJ) speech recognition corpus we have used is the publicly available WSJ0/WSJ1 (DARPA November'92 and November'93 Benchmark) sets. We have used the following split as development and evaluation sets. All the development data under WSJ1 for speaker inde-

pendent 20k vocabulary is used as the development set ("Dev 93" - 503 utterances). The evaluation is done on the November 92 CSR Speaker independent 20k NVP test set ("Test 92" - 333 utterances) and on the November 93 CSR HUB 1 test set ("Test 93" - 213 utterances). The vocabulary is the 20K open vocabulary word list for non-verbalized punctuation that is available in WSJ0/WSJ1 corpus. The data that is used for LM training is the whole WSJ 87, 88, and 89 sets. The statistics on the WSJ setting we have used is given in Table 6.9.

Table 6.9: The splits and the statistics of the Wall Street Journal corpus.

	LM Training	**Dev93**	**Test92**	**Test93**
No. of Sentences	1.6M	503	333	213
No. of Tokens	37M	8206	5643	3446
Vocabulary Size	165K	2461	1836	1314
OOV rate	2.7%	0.2%	0.0%	0.1%

6.3.1 ASR Baseline

The baseline ASR system is built by using the Kaldi [105] speech recognition toolkit. The language model that the baseline system uses is the baseline tri-gram back-off model for 20K open vocabulary for non-verbalized punctuation that is also available in the corpus.

The acoustic models are trained over the SI-284 data by using the publicly available Kaldi recipe with the following settings. MFCC features are extracted and spliced in time with a context window of $[-3, +3]$. Linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) are applied. Tri-phone Gaussian mixture models are trained over these features.

The ASR baseline performs weighted finite state decoding. We have extracted 100-best lists for each development and evaluation set. The (WER) performance of ASR and oracle hypotheses are given in Table 6.10.

Table 6.10: The WER performance of the ASR baseline system on Dev 93, Test 92, and Test 93 sets of the Wall Street Journal corpus.

	Dev 93	**Test 92**	**Test 93**
ASR 1-best	15.3%	10.2%	14.0%
Oracle on 100-best	8.3%	5.1%	7.3%

6.3.2 FrameNet Semantic Parsing

This section presents the frame and target accuracy of the ASR baseline. The semantic frames and targets are extracted by using the semantic-frame parser SEMAFOR [32]. The ASR performance is evaluated against the output of the semantic parser on the reference transcriptions. WSJ speech recognition corpus has 841 distinct frames and 29043 distinct targets on the LM training data. The density shows the ratio between the number of frames evoked and the number of tokens occur in the data. The accuracy of the frames is given in Table 6.11 and the accuracy of the targets is given in Table 6.12.

Table 6.11: Frame accuracy of the ASR baseline on the development and evaluation sets. Density shows the ratio between the number of frames evoked and the number of words occur in the data.

	Dev 93	**Test 92**	**Test 93**
Density	41.9%	42.4%	43.0%
Precision	88.3%	91.1%	90.1%
Recall	90.6%	93.3%	89.3%
F1	89.5%	92.2%	89.7%

Table 6.12: Target accuracy of the ASR baseline on the development and evaluation sets.

	Dev 93	**Test 92**	**Test 93**
Precision	88.5%	91.6%	90.7%
Recall	90.8%	93.8%	89.8%
F1	89.6%	92.7%	90.2%

Chapter 7

Joint Models for Spoken Language Understanding[1]

Spoken language understanding is the problem of extracting semantic structures from utterances. Therefore, spoken language systems (SLS) that aim at understanding what the user means consist of two main modules. The first module is the ASR and the second module is the SLU. The SLU module relies on the output of the ASR module, therefore, the errors introduced in the ASR module are propagated to the SLU module. One solution to this problem is to use multiple ASR hypotheses. In this chapter, first we show how LMs can be optimized either for their recognition performance or for their understanding performance. Then, we present how semantic models can be adapted to the current dialog by means of instance-based adaptation. Finally, we present an application of joint LMs to cross-language SLU porting.

The SLU module we present in this chapter takes the n-best list of ASR hypotheses from the ASR module. It has two components; the first component is the *alignment model*, this model outputs word-concept alignments for each hypothesis in the n-best list. The second component is the *scoring model*, which scores these word-concept alignments by joint LMs that is

[1]The work presented in this chapter is the revised version of the publications [5, 6, 119].

introduced in this chapter. The structure of the SLU module is depicted in Figure 7.1.

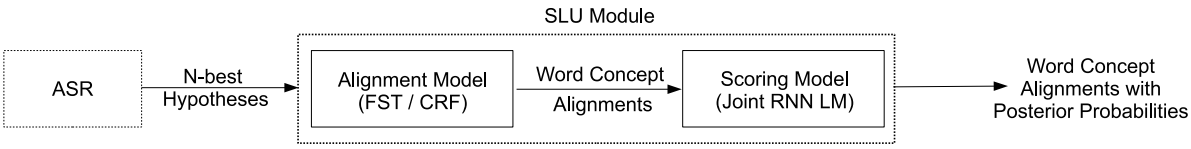

Figure 7.1: The structure of the SLU module. The SLU module consists of an alignment model and a scoring model. The alignment model extracts word-concept alignments for each hypothesis in the n-best list and the scoring model assigns posterior probabilities for these alignments.

7.1 Optimization of Joint LMs for SLU

This section addresses the training and optimization of joint LMs for ASR and SLU tasks and provides a procedure for training joint LMs and selecting its best parametrization. We show how to optimize joint LMs by re-scoring experiments on the LUNA HM corpus.

Joint LMs use word-concept pairs as the modeling unit for incorporating semantic information into the LM. This type of a model is first used in [136] for cache neural network LMs that also use the previous turns of the dialog in the cache component. They are reported to improve concept error rate.

We show an example of word-concept pairs for the following utterance from the LUNA HM corpus: *"Buongiorno io ho un problema con la stampante da questa mattina non riesco piu a stampare"*.

We have the following semantic annotation where concepts are shown in bold: "**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}".

The word-concept pairs that joint LMs use are constructed by using a

one-to-one mapping of words with their annotated concepts. For example the first five pairs are: *"buongiorno - null, io - null, ho - null, un - HardwareProblem.type, problema - HardwareProblem.type"*.

7.1.1 Joint RNNLMs

We use class-based RNNLMs as the architecture of joint LMs. The RNNLM structure we have used is a modified version of the class-based RNNLM structure given in Mikolov et al. [97], which is available as a toolkit[2]. The toolkit automatically assigns words to classes with respect to the frequencies of the words. We have modified the toolkit to handle manual clustering of the LM units (words or word-concept pairs). Joint LMs are constructed over word-concept pairs. We have clustered word-concept pairs with respect to concepts they evoke. Therefore, word-concept pairs which are semantically related, i.e. that have the same concept label, are mapped to the same class. The input layer has a node for each word-concept pair (w_i, c_i). Each word-concept pair is fed into the network using 1-of-n encoding. The LM probabilities at the output layer is factorized into class probabilities given the history and the class membership probabilities as in Equation 7.1, where (w_i, c_i) denotes the ith word-concept pair, h_i denotes the history for the ith pair, cl_i denotes the ith class, which is the class that (w_i, c_i), the ith word-concept pair, is assigned to.

$$P((w_i, c_i)|h_i) = P(cl_i|h_i)P((w_i, c_i)|cl_i, h_i) \tag{7.1}$$

The training of the RNNLM is done by using back-propagation through time (BPTT), in which the error is propagated through recurrent connections up to a certain previous time step. As given in [97], in this way it is guaranteed that the RNNLM learns the history. When calculating the

[2]Available at http://www.fit.vutbr.cz/~imikolov/rnnlm/

activations of the layers, the input layer and the recurrent layer is directly fed into the hidden layer. The activation of the hidden layer is computed by using the sigmoid function, and the output probabilities are computed by using the softmax function to guarantee a valid probability distribution. The structure of the joint RNNLM is given in Figure 7.2.

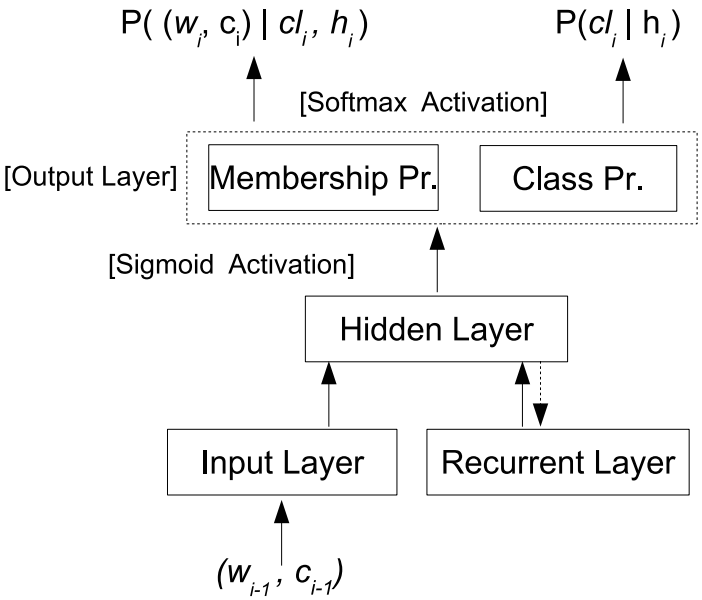

Figure 7.2: Joint RNNLM structure. The input layer has as many nodes as the number of distinct word-concept (w_i, c_i) pairs. The output layer estimates probabilities for all the classes and word-concept pairs. The classes are determined manually by mapping each word-concept pair that has the same concept label to the same class. The previous word-concept pair is fed to the input layer using 1-of-n encoding. (w_i, c_i) denotes the ith word-concept pair, cl_i denotes its class h_i denotes the history for that pair.

7.1.2 ASR Baseline

The baseline system is the system we have described for the LUNA HM corpus in Section 6.1. The ASR system is trained by using the Loquendo speech recognition system. It uses a modified tri-gram Kneser-Ney LM. The SLU model is based on stochastic finite state transducers that are presented in [107]. The SLU has 29.6% concept error rate (CER) on the

reference transcription of the test set. The WER and CER performance of the ASR output and the 100-best list is given in Table 7.1.

Table 7.1: The baseline recognition (WER) and understanding (CER) performance of the baseline system. The oracle performance is on the 100-best list generated by the ASR.

	WER	CER
ASR 1st-best	22.3%	46.3%
Oracle on 100-best	15.9%	35.2%

7.1.3 Baseline for Recognition

The baseline system that maximizes the *recognition performance* uses a word-based LM. In this setting, we have re-scored the 100-best list that the ASR outputs by using a class-based RNNLM that was constructed only over the words. The number of classes were given as a parameter, and the words were assigned to the classes with respect to their frequencies as given in [97]. We have found out that 150 classes with 100 hidden units were performing the best for the development set. The re-scoring was also done with linear interpolation of the RNNLM and the same tri-gram LM that was used by the ASR. The results are given in Table 7.2.

Table 7.2: Performance of the baseline system that maximizes the recognition performance. The class-based RNNLM is constructed over words. The RNNLM+ngram refers to the linear interpolation of the RNNLM with the tri-gram that is used in the ASR.

	WER	CER
RNNLM	21.5%	47.0%
RNNLM+ngram	21.5%	47.2%

As can be seen from the results; although we are able to reduce WER by performing re-scoring, there is a reduction in the understanding performance and in general SLU performance will not be predictable as WER is perturbed.

7.1.4 Re-scoring by Using Joint RNNLMs

We have optimized our system for SLU by using semantic components in the LM, i.e., we have built a joint LM that uses word-concept pairs as the LM unit. The training of the LMs is performed by using the reference ontology annotations in the training data. By using these annotations word-concept pairs are extracted for each utterance. We have trained an n-gram joint LM, and several RNNLMs with different sizes of hidden layers. We only report the results for the one which has the hidden layer size of 150, which gives the lowest perplexity on the reference word-concept pair annotation of the development set.

The joint RNNLM has 3639 nodes in the input layer, which is equal to the number of distinct word-concept pairs in the training set and a special token that denotes the end of utterance. The output layer consists of 45 classes; 44 for concepts and 1 for the *null* concept. In addition, it has 3639 nodes for each word-concept pair and the end of utterance token. The size of the hidden layer and the recurrent layer is 150. The network was trained for 14 iterations, by using the development set for setting the learning rate and for early stopping to avoid overfitting. Also a conventional n-gram model was trained by using the word-concept pairs. It is a tri-gram model with Kneser-Ney smoothing. The performance of 100-best re-scoring experiments with RNN, n-gram, and their linear interpolation is given in Table 7.3. As can be seen from the results, we have obtained an improvement in CER by using the joint RNNLM. The WER, on the other hand, has increased with respect to the baseline. Alternatively, the transcription performance can be improved by reducing the concept space. Therefore, the joint LM that is based on word-concept pairs is appropriate to optimize systems for understanding tasks. N-gram LM has suffered from data sparseness and performed worse than the baseline both for CER and

for WER. Better generalization ability of NNs makes them more robust to data sparseness, and makes them applicable to joint LMs.

Table 7.3: Performance of the joint RNNLMs after re-scoring the 100-best list. RNNLM and n-gram were trained on word-concept pairs from the reference transcription. RNNLM+n-gram refers to the linear interpolation of the two models.

	WER	CER
RNNLM	23.0%	44.1%
n-gram	26.7%	47.3%
RNNLM+n-gram	25.8%	46.8%

7.1.5 Parameter Optimization of the Joint Model

To see the effect of different amount of semantic information on the performance of ASR and SLU, we have trained RNNLMs with various samplings of the concepts to be included in the joint model parameters. We have selected 1, 2, 4, 8, 16, and 32 concepts from the set of concepts and map the other concepts to *null*. The concepts were grouped into 5 sets with respect to their frequencies. It was guaranteed that the concepts from all of these sets were selected randomly, while favoring the most frequent ones, i.e. when 8 concepts were selected, 2 concepts were selected from each of the most frequent 3 sets; and 1 each, from the rest. This randomization was performed for 5 times for each sampling. The mean and standard deviation of WER and CER for each of the samplings are given in Figure 7.3. As can be seen from the figure as the number of concepts incorporated into the LM increases there is a significant drop in the CER. On the other hand, WER increases initially as small number of concepts are included in the model and then as more concepts are added to the model WER is slightly affected.

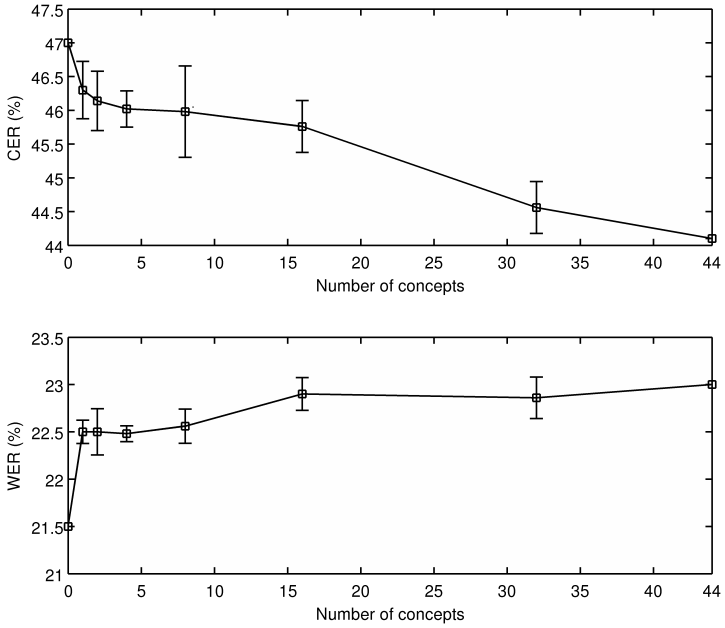

Figure 7.3: WER and CER for different number of SLU concepts. For each concept number we have sampled over the entire concept space and plot the mean and the standard deviation for 5 random concept draws.

7.1.6 Statistical Significance of the Results

In this section we show the statistical significance of the results by using two different methods. We have used the bootstrap method that is given in [18] to calculate the confidence intervals. We have calculated *bootstrap-t* confidence intervals using 10^4 bootstrap replications. The p-value is calculated using the randomization method given in [134]. The results are given in Table 7.4. It can be seen that the improvements are statistically significant.

7.1.7 Conclusion

In conclusion, we have presented LMs that are built over word-concepts pairs and aimed at increasing the understanding performance while compromising for higher WER. By performing re-scoring experiments over 100-

Table 7.4: WER and CER of the two systems that are optimized for ASR and SLU one by one. 90% confidence intervals using 10^4 bootstrap replications are given in brackets. Also p-values for WER and CER of the systems are presented. The results show that the improvements are significant.

	WER	CER
Best ASR	21.5% [20.2 - 22.7]	47.0% [44.2 - 49.7]
Best SLU	23.0% [21.7 - 24.4]	44.1% [41.3 - 46.7]
p-value	1.99e-4	9.99e-5

best lists, we have obtained 6% relative improvement in CER over the RNNLM that gives the best WER. The improvement is statistically significant. We have also shown that a better transcription performance does not always yield a better understanding performance. Spoken language systems may be tuned either for transcription or understanding task. The lexical-semantic relations used in the LM is very important when optimizing the system for a specific task. By searching over the lexical-semantic relation space, we may control the system with respect to its performance metric.

7.2 On-line Adaptation of Semantic Models

In this section, we present an instance-based on-line adaptation scheme for SLU scoring models. In this approach relevant instances are retrieved from the training data with respect to their similarity to the SLU hypothesis for that utterance. The background RNNLM scoring model is adapted on-line by using these instances. The n-best list for that utterance is re-scored by using the adapted scoring model.

LM adaptation has long been applied to ASR systems to improve their performance on a targeted domain. The process involves adapting a background LM by using domain specific data. In general, LM adaptation is applied to conventional n-gram LMs. However, recently there have been

studies that apply LM adaptation to neural networks (NNs). One of the approaches that has been applied to RNNLMs is to train the NN for one more iteration with the adaptation data [79]. Information retrieval approaches that use *tf* and *idf* statistics for selecting the relevant documents [1, 42] have been applied to LM adaptation.

Instance-based approaches solve a new problem by remembering similar problems that were encountered before [1]. Therefore, extracting the relevant instances from previous experience plays a crucial role. In addition to retrieval, how to use the retrieved instances for improvement is also important. The first step in instance-based learning cycle is the retrieval of the most similar instance or instances [1]. The process involves selecting instances from a collection of *previous instances* that are similar to the *new instance* at hand.

7.2.1 On-line Adaptation for SLU

SLU systems may benefit from adaptation as much as ASR systems. LM adaptation has been successfully applied to ASR systems for improving the performance of domain independent LMs on specific domains. Figure 7.4 shows the distributions of concept frequencies within dialogs that are randomly selected from the training set of the LUNA HM corpus. It can be seen that except for a few concepts that occur very frequently, the general concept distribution is very sparse. Due to this sparsity a general model may fail to capture the distributions well, and adaptation of the general model to the target dialog may yield improvements in the performance.

The adaptation is applied to the scoring model component of the SLU module which is depicted in Figure 7.1. The scoring model is a joint RNNLM that is presented in Section 7.1.1. with structure given in Figure 7.2. The adaptation procedure is applied to a background joint RNNLM model that is trained over the whole training data. Adaptation

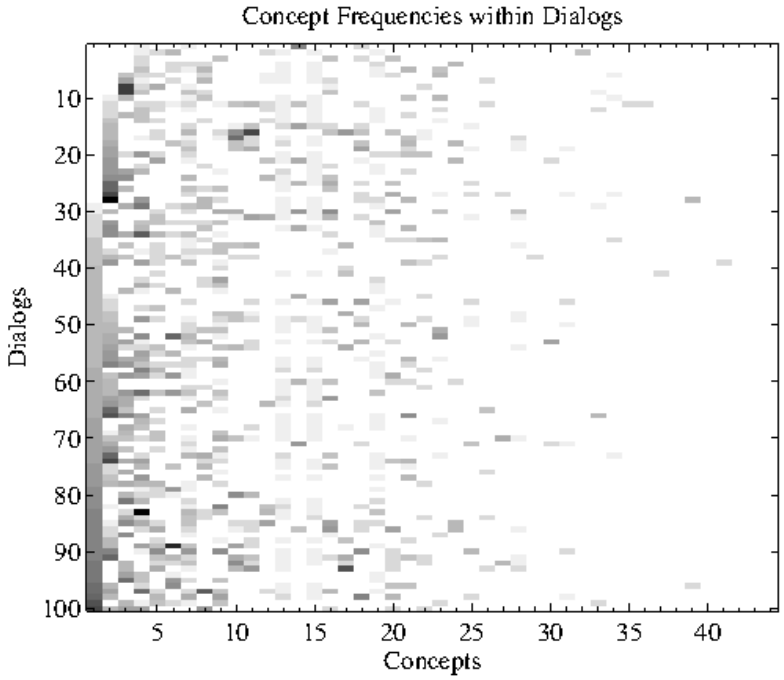

Figure 7.4: The concept frequencies within dialogs for the LUNA HM corpus. The dialogs are randomly selected from the training set. The darker areas show more frequent concepts, whereas the lighter areas show less frequent concepts. The concepts are rank ordered on the x-axis with respect to their frequencies in the training data.

procedure we adopt involves constructing the adaptation data for an utterance, and further training of the background model for 5 more iterations only with the adaptation data.

7.2.2 Instance-Based On-line Adaptation

The main component of instance-based adaptation is to retrieve the most similar instances from the training data for each test utterance. The retrieved instances are then used as the adaptation data for the target test utterances. This section first presents the instance retrieval process in detail especially for SLU systems. Then, two different similarity metrics are proposed. Finally, we provide the on-line adaptation architecture, and show how it can be applied to spoken language systems.

Instance retrieval

Instance retrieval searches for the most similar instances in the training data for each hypothesis that the system produces. Therefore, it computes a similarity score between the system hypothesis and each training set instance. The errors that the system introduces in the hypotheses decrease the precision of the similarity scores when these scores are computed on the reference transcription. Thus, to increase the precision, the training data is passed through the SLU system and similarity scores are computed on the system hypotheses for the training data. However, the instances are retrieved from the corresponding reference transcription. In addition, since in general ASR is more precise on meaning bearing words, the words that map to *null* concepts are pruned before the similarity scores are computed. For the adaptation procedure the comparison is performed over three different tokenizations; word-concept pairs, words, and concepts. This process is depicted in Figure 7.5.

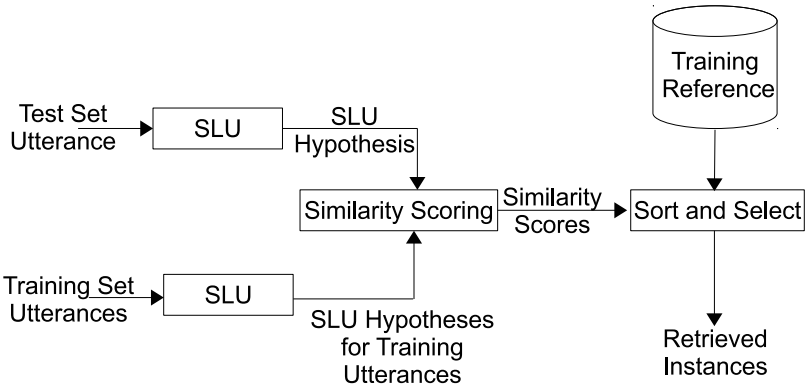

Figure 7.5: Instance retrieval process for SLU systems. To compensate errors that SLU produces similarity scores are computed between the SLU hypothesis of the test utterance and the SLU hypotheses of the training set. However, the instances are retrieved from the reference transcription of the training data.

Similarity metrics

We have used the following metrics for computing the similarity for instance retrieval. The first metric is the *edit distance* in which the hypothesis is aligned with every utterance in the training data and the total number of errors (deletions, insertions, and substitutions) are computed for each alignment. Then, the instances are sorted in ascending order with respect to the number of errors and they are retrieved from the reference transcription of the training data.

The second metric is the *n-gram match score*, which computes the similarity by considering n-grams. To compute the similarity, each system hypothesis is aligned with the hypotheses of the training data. The score is computed by using Equation 7.2, where n refers to the number of words in the system hypothesis, ug, bg, and ng refer to matching uni-gram count, matching bi-gram count and matching n-gram count respectively, and ins refers to the number of insertions. The instances are sorted in descending order and instances are retrieved from the reference transcription of the training data.

$$score = (\frac{ug}{n} + \frac{bg}{n-1} + ... + \frac{ng}{1} - \frac{ins}{n})/n \tag{7.2}$$

Instance-based on-line adaptation scheme

The instance-based on-line adaptation procedure can be applied at the SLU output. The general flow of instance-based on-line adaptation is as follows. The first step is to retrieve the relevant instances from the training data. Then, the background model is adapted by using these instances. The n-best hypotheses of the system are re-scored by combining the posterior probabilities of the adapted model with acoustic scores. The general flow is depicted in Figure 7.6.

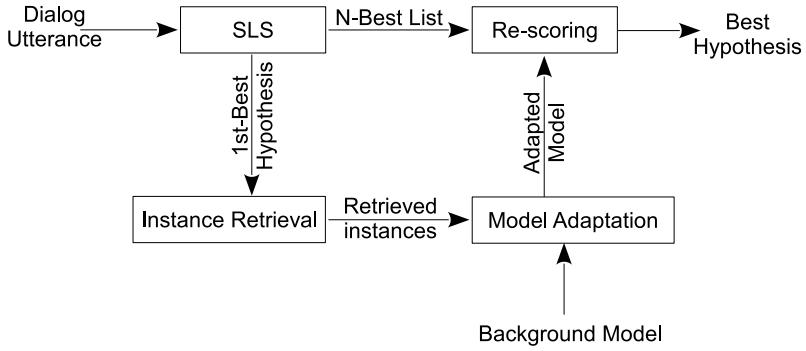

Figure 7.6: The diagram of instance-based on-line adaptation scheme. The relevant instances are selected by using the spoken language system (SLS) hypothesis. The background model is adapted by using the retrieved instances. N-best SLS hypotheses are re-scored by using the adapted model.

7.2.3 LUNA HM Experiments

The baseline system has two modules; an ASR system and a SLU model that are described in Section 6.1.1 and Section 6.1.2 respectively. The ASR system is built by the Loquendo speech recognition system and the SLU alignment model is built with conditional random fields (CRFs). The CER performance of this model with the performance of the baseline system is given in Table 7.5.

Table 7.5: Baseline SLU performance (CER). Oracle CER is given for the 100-best list. The baseline performance of the background RNNLM scoring model is given for 100-best list re-scoring.

	CER
1-best	26.7%
Oracle on the 100-best list	18.3%
100-best re-scored with the RNNLM model	26.1%

Lower bounds for instance-based on-line adaptation

This section presents the lower bounds by using both the reference transcription of the test set and the oracle hypothesis of SLU system. The instance retrieval process differs at the similarity computation for these

experiments. The utterances do not include any errors when the reference transcription is used or they are at minimum when the oracle hypothesis is used. Therefore, similarity scores are computed with the reference transcription of the training data rather than the SLU hypotheses. Table 7.6 gives the performance of the system with the reference transcription and Table 7.7 presents the performance for the oracle hypothesis.

As can be seen from the results a significant improvement can be achieved when instance-based on-line adaptation is applied to the SLU model. In general we can see that the performance is best when similarity is computed at *concept* tokens. Additionally, using the oracle hypothesis yields better performance than using the reference transcription. We can obtain 10.8% relative (2.9% absolute) improvement on CER with respect to the baseline when oracle hypothesis is used and similarity is computed at the *concept* level with *n-gram match* score.

Table 7.6: CER lower bounds when using the reference transcriptions as input to instance retrieval. "Ins." refers to the number of instances that are retrieved; 3, 9, 16, 31, and 158 corresponds to 0.1%, 0.3%, 0.5%, 1.0%, and 5.0% of the number of training utterances. "wc pr." refers to word-concept pairs. "conc." refers to concept tokenization.

Ins.	**Edit distance**			**n-gram match score**		
	wc pr.	**words**	**conc.**	**wc pr.**	**words**	**conc.**
1	25.2%	25.7%	25.0%	24.9%	25.6%	25.4%
3	24.8%	**24.8%**	25.1%	25.3%	25.3%	24.8%
9	24.8%	25.1%	**24.4%**	**25.2%**	25.8%	**24.4%**
16	24.9%	25.6%	**24.4%**	**25.2%**	25.4%	24.7%
31	24.9%	25.1%	24.7%	25.3%	**25.2%**	24.7%
158	**24.5%**	25.6%	25.2%	25.7%	26.1%	25.3%

Actual performance of instance-based on-line adaptation

In this section we present actual performance of the instance-based on-line adaptation on the SLU model. Therefore, instance retrieval is performed by using the SLU hypothesis of the system. As we have mentioned, to

Table 7.7: CER lower bounds when using the oracle hypotheses as input to instance retrieval. "Ins." refers to the number of instances that are retrieved; 3, 9, 16, 31, and 158 corresponds to 0.1%, 0.3%, 0.5%, 1.0%, and 5.0% of the number of training utterances. "wc pr." refers to word-concept pairs. "conc." refers to concept tokenization.

Ins.	**Edit distance**			**n-gram match score**		
	wc pr.	**words**	**conc.**	**wc pr.**	**words**	**conc.**
1	25.3%	25.3%	25.1%	24.8%	25.1%	25.6%
3	**24.6%**	25.1%	25.2%	24.7%	**24.7%**	25.1%
9	**24.6%**	25.3%	**24.1%**	**24.6%**	24.9%	24.7%
16	24.7%	25.1%	24.2%	**24.6%**	25.0%	24.2%
31	24.8%	**24.7%**	24.3%	24.8%	24.9%	**23.8%**
158	24.8%	25.1%	24.8%	25.4%	25.7%	25.3%

compensate for the errors that SLU hypothesis possesses we have used the SLU hypotheses of the training data when computing the similarity scores. The performance of this approach is given in Table 7.8.

Table 7.8: CER on-line adaptation performances. The instances are retrieved by using the SLU hypothesis of the system for each utterance. "Ins." refers to the number of instances that are retrieved; 3, 9, 16, 31, and 158 corresponds to 0.1%, 0.3%, 0.5%, 1.0%, and 5.0% of the number of training utterances. "wc pr." refers to word-concept pairs. "conc." refers to concept tokenization.

Ins.	**Edit distance**			**n-gram match score**		
	wc pr.	**words**	**conc.**	**wc pr.**	**words**	**conc.**
1	26.0%	26.1%	26.0%	25.7%	25.9%	26.1%
3	26.3%	25.8%	26.6%	25.4%	**25.2%**	26.2%
9	**25.3%**	25.2%	**25.7%**	**25.3%**	25.3%	25.9%
16	**25.3%**	**25.1%**	25.9%	25.6%	25.8%	26.2%
31	25.5%	25.2%	26.1%	25.7%	25.6%	25.7%
158	**25.3%**	26.2%	26.0%	25.8%	26.1%	**25.6%**

The results show that when the instance-based on-line adaptation is applied to the SLU model, it gives significant improvements on CER. When these results are compared to the lower bounds we can see that there is still a huge gap of possible improvement. In addition to that, when similarity computation scores over *word* tokens give the worst performance with the reference transcription and the oracle hypothesis, they perform the best

when actual SLU hypothesis is used for instance retrieval. Also *concept* tokens perform the worst which is not the case with the lower bounds. This is most likely due to the fact that the SLU hypothesis has more errors on *concept* tokens when compared to *word* tokens. We have obtained 6.0% relative (1.6% absolute) performance improvement on CER with respect to the baseline system.

7.2.4 Statistical Significance of the Results

This sections shows that achieved improvements on CER by using instance-based on-line adaptation for the SLU model are statistically significant with respect to the baseline system. We compare the performance of the baseline system with the best performing on-line adaptation system (Table 7.8) with the two similarity metrics on *word* tokens. The *bootstrap-t* confidence intervals are calculated by using bootstrap method that is given in [18]. In addition, p-values are calculated by using the randomization method given in [134] which is implemented in the toolkit[3]. As can be seen from Table 7.9 the improvements on CER are statistically significant since p-values are smaller than 0.05.

Table 7.9: The comparison of the baseline system with the instance-based on-line SLU model adaptation. 90% confidence intervals using 10^4 bootstrap replications are given in brackets. Also p-values for the comparison between the baseline and the two approaches are given. The results show that the improvements are significant.

	CER	p-value
Baseline	26.7% [24.2 - 29.2]	NA
Edit dist. best	25.1% [22.7 - 27.5]	0.01
n-gram match best	25.2% [22.8 - 27.7]	0.03

[3]The toolkit is available at http://www.nlpado.de/~sebastian/software/sigf.shtml.

7.2.5 Conclusion

In this section, we have presented an instance-based on-line adaptation scheme that aims at improving the performance of SLU systems. The main idea behind instance-based on-line adaptation is to select relevant instances from the training data by using the hypothesis that the system outputs for each utterance. These instances are then used to adapt the model that will be used for re-scoring. We have achieved significant improvements on CER for SLU by using *word* tokens with the *edit distance* and the *n-gram match score* metrics. However, there is still a huge possibility of improvement as the lower bounds show.

7.3 Application of Joint LMs to Cross-Language SLU Porting

In this section, we show how joint models can be applied to cross-language SLU porting by using statistical machine translation (SMT). The work presented here is a collaborative work that also focuses on the details of improving the performance SMT systems. However, in this section SMT systems are not analyzed in detail and just an overview is given. The main focus is on the use of joint models on cross-language SLU porting.

Cross language porting is the problem of transferring the semantic knowledge obtained in one language (*source language*) to a new language (*target language*) where there is no semantic annotation [65, 117]. Automatic cross-language porting uses statistical machine translation (SMT) for translating and aligning the resources. The methodology can be divided in two categories with respect to the direction of translation: *Test-on-Source* and *Test-on-Target* (also known as *Train-on-Target*) [64]. In Test-on-source the direction of translation is from the target language to

the source language i.e. user utterances are first translated to the source language and the existing SLU model in the source language is applied. In this manner, the SLU system is extended for the new (target) language. In Test-on-Target, on the other hand, the direction of translation is from the source language to the target language, i.e. the semantically annotated data in the source language is translated and the annotation are transferred based on SMT alignments. A new SLU model is trained on the target language. In both of the approaches, the quality of SLU porting depends directly on the quality of the automatic translation. Test-on-Source approach has been credited to have a better performance [65, 63, 64, 86]. In addition, it is simpler to implement compared to the Test-on-Target approach since no semantic annotation is transferred through SMT. In this section, we present how joint LMs can be applied to the Test-on-Source SLU porting pipeline.

The Test-on-Source approach presented in this section uses off-the-shelf SMT systems with style adaptation and a manually trained SMT system on out-of-domain data with domain adaptation. The corpora used for domain adaptation is the in-domain corpus which is used to train the SLU model. The style adaptation and domain adaptation take place in the SMT pipeline. At the final step the semantic hypotheses of the translation output are re-scored by using joint LMs. The complete Test-on-Source SLU porting pipeline is given in Figure 7.7. We evaluate our approach on two different language pairs on the LUNA HM corpus: Spanish-Italian, which is a close language pair, and Turkish-Italian, which is a distant language pair.

7.3.1 Corpora

The in-domain corpus used throughout the experiments is the LUNA HM corpus. Multilingual LUNA corpus [120] is the translation of Italian LUNA

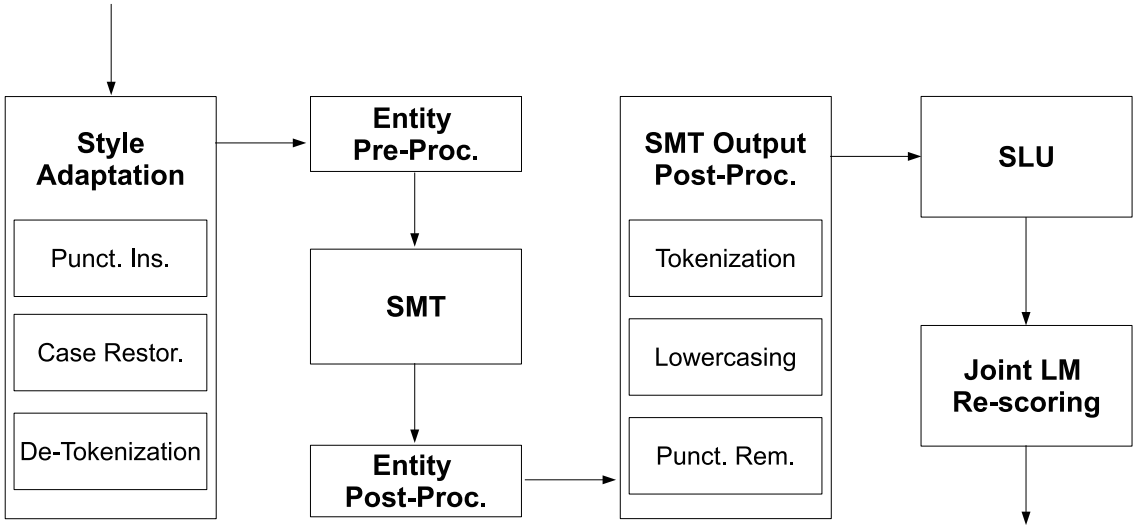

Figure 7.7: Test-on-Source SMT based cross-language SLU porting pipeline. The target language utterances are translated to the source language by using an SMT system that output multiple hypotheses. Translation hypotheses are passed through the SLU module in the source language. The SLU hypotheses are re-scored by using the joint LM model.

HM corpus to the target languages Spanish, Turkish, and Greek by professional translation services. The corpus is used to train the in-domain SMT systems: Spanish-Italian and Turkish-Italian.

The out-of-domain Europarl Parallel Corpus [76] of the proceedings of the European Parliament is the most popular corpus in machine translation community. It encompasses 21 European languages, including the languages of interest: Spanish and Italian. Version 7 (from May 2012) of the corpus is used to create Spanish-Italian parallel corpus of approximately 1.8M sentence pairs. This parallel corpus is used to train the out-of-domain Spanish-Italian SMT system.

Europarl is used for Spanish, and for Turkish Wikipedia dump is used to train LMs for language style adaptation experiments. The Wikipedia text was extracted and sentence split to result in approximately 3M sentences

7.3.2 SMT Systems

Google Translate is a general-domain SMT system designed to provide reliable translations of text in multiple genres. It is trained on a vast variety of parallel written texts (as opposed to speech transcriptions). Since it is targeted for a wide range of languages, the translations go through English as an intermediate language, i.e. a sentence in Turkish or Spanish is first translated into English and then to Italian.

Europarl Moses is an out-of-domain SMT system that is trained by using Moses [77] SMT toolkit. The toolkit supports various translation models: phrase-based and tree-based, as well as factored models; and input of different levels of complexity from text to ASR lattices. Here we use a phrase-based translation model on plain text. Prior to training, Europarl corpus was pre-processed to be suitable for speech transcriptions: it was tokenized, lowercased and all punctuation was removed.

LUNA Moses is an in-domain SMT system that is trained with the Moses toolkit. Multilingual LUNA Corpus was used to train both Spanish-Italian and Turkish-Italian systems. These systems represent a lower-bound performance.

7.3.3 Style Adaptation

The use of off-the-shelf SMT enables users to obtain satisfactory translations without the need for the expertise in SMT, however, these systems are general domain and trained on written text. Therefore, the performance of this general domain systems drop when the style changes from written text to spoken conversation. The performance of these systems can be improved by pre-processing the input and then post-processing the output. Therefore the following steps are applied to improve the performance of SMT systems:

SMT Output Post-Processing: The SMT output is tokenized, converted to lowercase, and all punctuation except single quotes used in contractions are removed.

Language Style Adaptation: The conversation style is converted to the written text style. The steps include:

1. *Automatic Punctuation Insertion:* A language model is used for inserting punctuation to the transcriptions of the conversations.
2. *Automatic Case Restoration: Moses recaser* is used that is available in the Moses toolkit to train translation models from lowercased to cased text.
3. *De-tokenization:* Punctuation marks and contractions are attached to respective tokens with language based rules.

Entity Processing for SMT: Numerical expressions, which are most frequent entities in the LUNA HM corpus, are converted to digits in the target language before passing them to the SMT system. Then, after the translations these entities are converted back into their corresponding word forms.

7.3.4 Domain Adaptation

Training SMT systems require parallel corpora in the source and the target language, which is a limited resource. In addition, SMT systems are sensitive to the differences in the domain the system is trained and tested. There are variety of methods proposed for domain adaptation to improve the performance of SMT systems that are trained on an out-of-domain corpus [17, 78]. Simple approaches for domain adaptation are [78]:

1. Pooling large out-of-domain and small in-domain parallel corpora together for training the models.

2. Using the out-of-domain corpus for the translation model and the in-domain-corpus for the LM.

Domain adaptation is only applied to the Spanish-Italian data since Europarl is not available for Turkish. Domain adaptation is applied by using the in-domain corpus, i.e. LUNA HM corpus.

7.3.5 SLU Performance

In this section we present the Test-on-Source performance of several SMT systems. First, results with style adapted SMTs are presented for Spanish-Italian and Turkish-Italian language pairs. Next, the results with domain adapted SMTs are given. Domain adaptation, is only presented for the language pair Spanish-Italian since domain adaptation is applied to the "Europarl Moses" SMT system which is not available for Turkish-Italian.

We use the same joint LM that is used for the instance-based on-line adaptation experiments presented in Section 7.2. The SLU model we use is also the same CRF model that is used for the instance-based on-line adaptation experiments, that is described in Section 6.1.2. The re-scoring scheme is similar to the ones we have presented in this chapter. The n-best hypotheses that the SMT outputs are passed through the CRF SLU model and word-concept alignments are obtained. The joint RNNLM is used to re-score these word concept alignments. When off-the-shelf SLU systems are used ("Google Translate") only the posterior probability of the joint RNNLM is used during re-scoring. On the other hand, for the trained SMT systems, since we can obtain the translation scores, we combine translation scores with the posterior probability of the joint RNNLM for re-ranking.

Table 7.10 shows the Test-on-Source performance for style adaptation. The style adaptation improves the performance of the SMT systems, therefore, the SLU performance also increases (CER decreases) when style

Table 7.10: The CER performance of Test-on-Source SLU porting with style adapted SMT systems. Since LUNA Moses is trained on speech, no style adaptation is applied. The Oracle performance of the n-best list is given in parentheses.

SMT System	**Baseline**	**Style Adapted**	**Re-scoring JLMs**
Spanish - Italian			
Google Translate	43.0%	36.1%	34.6% (31.1%)
Europarl Moses	39.2%	35.4%	31.3% (22.8%)
LUNA Moses	25.8%	N/A%	25.3% (20.7%)
Turkish - Italian			
Google Translate	56.9%	50.4%	49.2% (44.7%)
LUNA Moses	39.2%	N/A%	37.9% (27.7%)

adapted systems are compared with their baselines. When performing the re-scoring experiments, for the "Moses" SMT systems the joint LM scores is combined with the translation scores, however, since "Google Translate" is a closed system only joint LM scores are used for re-scoring. This affects the contribution of re-scoring such that "Moses" SMT systems benefit from re-scoring more than "Google Translate". In addition, the "Moses" SMT systems output 100 best hypotheses where as "Google Translate" outputs 4.5 hypotheses per sentence on average.

Table 7.11: The CER performance of Test-on-Source SLU porting with domain adaptation for the language pair Spanish-Italian. The Oracle performance of the n-best list is given in parentheses.

Translation Model	**Language Model**	**SLU 1st-best**	**Re-scoring JLMs**
LUNA HM	LUNA HM	25.8%	25.3% (20.7%)
Europarl	Europarl	35.4%	**31.3%** (22.8%)
	LUNA HM	**31.2%**	29.8% (23.6%)
Europarl + LUNA HM	Europarl + LUNA HM	28.4%	27.2% (23.1%)

We present the CER performance of Test-on-Source SLU porting with domain adaptation in Table 7.11 for Spanish-Italian language pair. We observe that re-scoring with joint LMs improve the understanding performance in general. For the in-domain SMT system (LUNA HM with LUNA HM Table 7.11) the improvement is not significant. However, for the out-

of-domain SMT systems re-scoring improves the SLU as much as domain adaptation (comparison of bold CERs in the second group of Table 7.11). In general, the improvement with re-scoring is proportional to the amount of out-of-domain data that is present in the LM of SMT systems. Thus, re-scoring achieves 11.6% relative (4.1% absolute) improvement on the out-of-domain "Europarl Moses" system over the SLU 1st-best hypotheses.

7.4 Discussion

In this chapter we propose SLU module to be composed of two different models for processing multiple hypotheses aiming at improving the SLU performance. The first model (alignment model) assigns word-concept alignments to n-best hypotheses of ASR and the second model scores each alignment hypotheses. We mainly focus on the scoring model, where the alignment model can be a standard generative model that is based on SFSTs or a discriminative model that uses CRF for word-concept alignments. We train joint RNNLM models, which use word-concept pairs as the unit of modeling. In addition, they use the RNNLM architecture to exploit distributed representations these architectures work on.

As previously mentioned, the best hypothesis that gives a better WER does not necessarily lead to a better understanding performance. We have shown that joint RNNLMs are appropriate architectures that can be tuned either for recognition tasks or for understanding tasks. The optimization can be done by incorporating different amount of semantic information in these joint RNNLMs. In addition, we have shown that a joint RNNLM has a significantly better understanding performance than a RNNLM that is built over words.

We have also presented how this joint RNNLM scoring models can be adapted to the context of the dialog by using instance-based on-line adap-

tation. We have observed a huge gap of possible improvement when accurate instances are retrieved for an utterance. On the other hand, we have achieved significant improvements by performing instance-based adaptation on joint RNNLMs.

Finally, we have applied the proposed SLU module to cross-language SLU porting. In this task the SLU module rather than processing multiple ASR hypotheses, processes multiple SMT hypotheses. In general we have observed improvements over the SLU 1st-best hypothesis when multiple hypotheses are re-scored by using joint RNNLMs. The most important contribution of joint RNNLMs is that they can replace the process of domain-adaptation with almost same performance improvement. We have observed that the amount of contribution the joint RNNLMs can supply is inversely proportional to the amount of out-of-domain data that is present in the LM of the SMT system.

In conclusion, joint RNNLMs are promising for improving SLU in spoken language systems. A general model can also be adapted to the dialog context. In addition, they perform well on other SLU tasks like cross-language SLU porting.

Chapter 8

Semantic Language Models[1]

The task of language modeling is simply to predict words. The artificial language processors can learn from humans, who are the most effective language processors. Cognitive studies show that humans may be anticipating words as they are incrementally processing a sentence. For instance, in [125] human subjects were presented sentences like the following in Dutch:

> *"The burglar had no trouble locating the secret family safe. Of course, it was situated behind a big but unobtrusive..."*

which is expected to be completed by the word *painting*. In Dutch the gender of the adjective *big* must agree with the gender of noun that is not presented. When subjects were presented with prediction-inconsistent adjectives, which did not agree with *painting*, a positive deflection was observed in event-related brain potential (ERP) waveforms. This deflection was observed because of the failed prediction that subjects came up with. This effect was not observed in ERP when the sentences are presented without the constraining discourse. This suggests that human language processing involves a prediction process which benefits from discourse.

In this thesis, we are not trying to model human cognition, or emulate human language processing. On the other hand, we believe that language

[1]The work presented in this chapter is the extended and the revised version of the publication [7].

models can benefit from contextual information in a similar way. The concept of contextual information can be very broad which can range from simple semantic information like the *topic* of an utterance to any information about the environment where the speech takes place, which would be almost infeasible to capture. In this thesis, we restrict the contextual information with the linguistic scene rather than any perceptual cues that are in the environment. The linguistic scene is very much related to the context of an utterance, as Fillmore [46] points out this is a pragmatic knowledge and can be represented by *frames*. For example, the linguistic scene related to the words "buy", "sell", or "pay" can be represented by a commercial event scenario frame and the frame elements or slots of this frame can be filled with the further relevant information in the sentence.

On the other hand, as Bellegarda [10] points out one way to overcome the locality[2] problem in language modeling is by performing span extension. Span extension can be performed either syntactically or semantically. In this chapter, we introduce semantic language models (SELMs) which perform semantic span extension in the direction Fillmore [46] points out, by means of the theory of frame semantics.

8.1 The Linguistic Scene

The linguistic scene or the linguistic context can be considered as the pragmatic knowledge about the utterance. In this respect, we use the theory of frame semantics for modeling the linguistic scene. In the commercial event scenario example by Fillmore [46] the *target* words like "buy", "sell", etc. trigger the frame commercial event scenario. Figure 8.1 presents a similar example from Penn-Treebank [88]. In this example, the frames "Commerce Scenario" and "Commerce Sell" create a linguistic scene where the non-

[2]Locality problem refers to the deficiency of fixed histories in handling long-range dependencies.

Figure 8.1: An example sentence from Penn-Treebank. The target or target words are shown in red, the evoked frames are shown in blue. The linguistic scene constructed by the frames (in bold blue) "Commerce Scenario" and "Commerce Sell" helps in predicting the relevant non-target word "market", which is shown in green.

target word "market" is an *expected* relevant word. Therefore, we expect that a model that uses this pragmatic information that frames construct, must have a high expectation for this relevant word, "market". In addition, the target words "traders" and "selling" are also related with the word "market" and must increase the expectation of the model for "market".

In this thesis, we model and consider the linguistic scene or the semantic context of a sentence as the set of frames evoked, or the set of targets that occur in that sentence. Hence, SELMs exploit the semantic relationship between the linguistic scene and relevant words that occur in the sentence.

Before going into the details of training SELMs, we present a working word prediction example over the sentence given in Figure 8.1 by using SELMs that will be described in detail. We compare probability estimates of SELMs with n-gram models and RNNLMs. The prediction example is on estimating a probability for the non-target relevant word "market" in the context of the given sentence. Also, we replace the word "market" with an irrelevant word, "computer", in the same context and estimate its probability by using the same models. For probability estimation, we have trained the following LMs: a modified Kneser-Ney 5-gram model, a RNNME model with a 4-gram maximum entropy model (RNNME), and a

SELM that uses frames as the semantic context (SELM on Frames). The probability estimates of these LMs for the words "market" and "computer" in the same context are given in table 8.1.

Table 8.1: Probability estimates for the relevant non-target word *market* and the substituted irrelevant word *computer* in the same context. For the 5-gram LM, the history, h is the fixed history of preceding words; for the RNNME the preceding word and the hidden state. For the SELM, h is the preceding word, the hidden state and the set of the semantic frames evoked in the utterance.

Model	$\mathbf{P}(market \mid \mathrm{h})$	$\mathbf{P}(computer \mid \mathrm{h})$
Modified Kneser-Ney 5-gram	4.2×10^{-3}	8.2×10^{-4}
RNNME	6.9×10^{-3}	1.8×10^{-3}
SELM on Frames	1.2×10^{-2}	1.9×10^{-5}

As can be seen in Table 8.1 the highest estimate for the relevant word "market" and the lowest estimate for the irrelevant word "computer" is given by the SELM. The 5-gram model estimates the probability by considering only the recent history of 4 words, therefore, it would not be able to model any long-range dependencies in this sentence[3]. The RNNME models long-range dependencies by using the recurrent connections, however, the RNNME considers sentences as sequences of words and does not exploit the semantic relationships between these words. The SELM, on the other hand, uses the set of frames evoked in the sentence as the semantic context and explicitly models the relationship between the frames that are evoked and the words that occur in the sentence. Hence, it has a high expectation for the semantically relevant word "market" and a low expectation for the irrelevant word "computer".

[3]In this example, a 5-gram would not be able to capture the dependency between the word "market" and the relevant target word "traders" that evokes the "Commerce Scenario" frame.

8.2 Frames and Targets as the Linguistic Scene: Feature Extraction

We choose to represent the linguistic scene or the semantic context of sentences by the set of frames that are evoked and the set of targets. Therefore, to recognize targets and identify which frames are evoked we use the frame-semantic parser SEMAFOR [32] for English and the LUNA FrameNet parser [29] for Italian.

The feature extraction process for an English utterance proceeds as follows. First,the utterance is fed into the SEMAFOR frame-semantic parser. Then the set of frames or targets in that utterance is extracted as features of the semantic context. We investigate two different way of constructing the semantic context: one is based only on the set of frames, and the other only on the set of targets. The feature extraction step for frames is depicted in Figure 8.2.

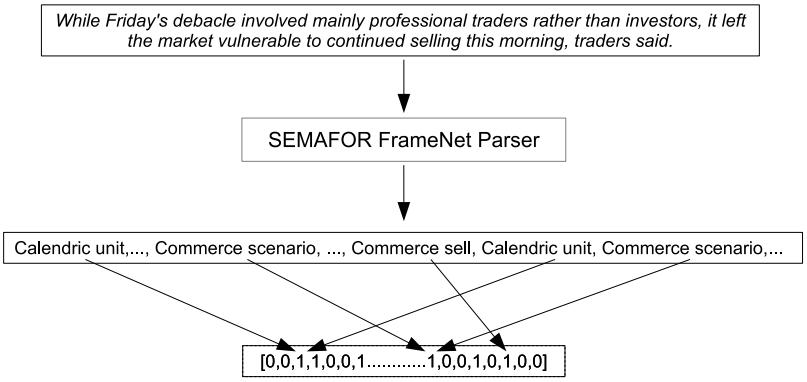

Figure 8.2: Semantic feature extraction for frames. The utterance is fed into the semantic parser, and the frames are extracted. The semantic feature is a binary vector over frames, where the component at index i is set to 1, if the frame i occurs.

8.3 SELM Structure

The SELM architecture we present is based on the context-dependent RNNLM structure that is given in [92]. We aim at using distributed word representations in SELMs by using neural networks. In addition, using the RNNLM architecture will enable us to model long-range dependencies in a better way. The semantic features that are presented in the previous section[4] are used as the context in the context-dependent RNNLM. The structure of the SELM is given in Figure 8.3. Further, we also use n-gram maximum entropy features that are trained jointly and implemented as direct connections in RNNMEs [91]. N-gram maximum entropy features are not shown in the figure.

8.4 Penn-Treebank Experiments

In this section we present the perplexity result on the Penn-Treebank [88] part of the Wall Street Journal corpus. The experiments presented here are performed on the same data with the same settings and by using the same pre-processing steps as given in [44, 94, 92].

The pre-processing steps involve replacing numerical values with a special token "N" and limiting the vocabulary to the most frequent 10K tokens. Out-of-vocabulary words are replaced with the "<UNK>" token. The following split is used as training, development, and evaluation splits. Sections 0-20 are used as the training set, sections 21-22 are used as the development set, and sections 23-24 are used as the evaluation set. These sets have 930K, 74K, and 82K tokens respectively.

The semantic features are obtained from the raw data which are not pre-processed. The semantic features, i.e., the binary vector over the frames

[4]The binary vector over frames or targets.

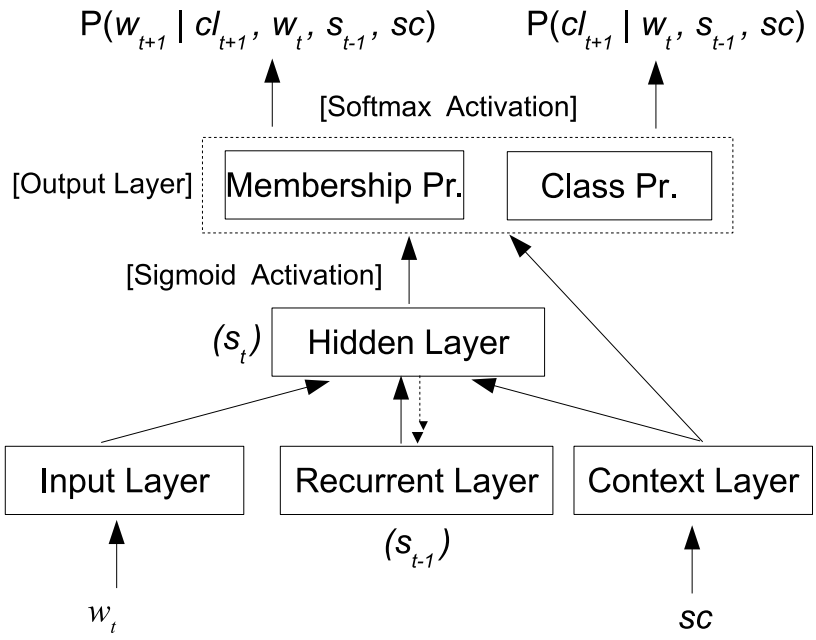

Figure 8.3: The SELM structure that is based on the class-based RNNLM structure. The network takes the current word w_t and the semantic context sc for the current utterance as input. In addition the previous hidden state is copied into the recurrent layer s_{t-1}. The output layer estimates the probability for the next word w_{t+1} factorized into class probabilities and class-membership probabilities (cl_{t+1} denotes the recognized class for the next word). Therefore these probabilities are conditioned on the current word w_t, the previous state s_{t-1} and the semantic context sc. The direct connections from n-gram histories to output layers are not shown.

and the targets, are extracted by using the semantic feature extraction step described before. We have obtained 819 distinct frames and 11271 distinct targets for the training set.

For comparison we have trained a 5-gram modified Kneser-Ney LM with singleton cut-offs (KN5), a 4-gram feed forward NNLM that has 160 nodes in the hidden layer, and a RNNME model that has 150 nodes in the hidden layer and that uses maximum entropy model up to 4-grams with a size of 10^9 connections. The NNLMs use the same 200 word classes that are obtained with respect to the frequencies of the words.

The SELMs are trained with the set of frames ("SELM on Frames") and

with the set of targets ("SELM on Targets") separately. All SELMs use the same word clusters as the other NNLMs to reduce the computational complexity. They also use maximum entropy features up to 4-grams with 10^9 connections. They have a hidden layer size of 200. In addition to these SELMs, we have also trained SELMs on the most frequent frames that cover the 80% of the training set ("SELM on 80% Frames") and on the most frequent targets that cover the 80% of the training set ("SELM on 80% Targets"). In this setting, the number of frames is 181 and the number of targets is 1386. The perplexities of the development and the test set are given in Table 8.2.

Table 8.2: Perplexity (PPL) results on Penn-Treebank part of the Wall Street Journal corpus on the development split (DEV PPL) and the test split (Test PPL). SELMs achieve 50% and 64% relative reduction in perplexity with respect to Kneser-Ney 5-gram model when frames and targets are used as semantic context respectively.

Model	**Dev PPL**	**Test PPL**
KN5	148.0	141.2
FF4	165.9	156.3
RNNME	133.6	127.9
SELM on Frames	**73.7**	**70.3**
SELM on 80% Frames	84.6	81.4
SELM on Targets	**53.8**	**51.1**
SELM on 80% Targets	63.3	60.5

As can be seen from the results we have achieved 50% and 64% reduction in perplexity with respect to the modified Kneser-Ney 5-gram model. However, these result may be misleading and should not be compared directly to the other LMs because SELMs use also the future information in estimating probabilities. In addition, self information about a word is also used if the predicted word is a target. Limiting the semantic context to the most frequent 80% frames or targets also gives satisfactory results. Therefore, to reduce the complexity we continue our re-scoring experiments with frames and targets of 80% coverage.

8.5 Wall Street Journal Experiments

As it is mentioned in the previous section perplexity results may be biased because of the way SELMs employ semantic information. To assess the performance of these models better, we present the WER performance on N-best re-scoring experiments on the WSJ speech recognition task. All of the experiments presented in this section are performed by using the publicly available WSJ0/WSJ1 (DARPA November'92 and November'93 Benchmark) sets. The acoustic models are trained on the WSJ0/WSJ1 training utterances also known as SI-284. All the development data under WSJ1 for speaker independent 20k vocabulary is used as the development set ("Dev 93" - 503 utterances). The evaluation is done on the November 92 CSR Speaker independent 20k NVP test set ("Test 92" - 333 utterances) and on the November 93 CSR HUB 1 test set ("Test 93" - 213 utterances).

8.5.1 ASR baseline

The baseline ASR system is built by using the Kaldi speech recognition toolkit [105] that is given in Section 6.3.1. This system generates the N-best lists that are used for re-scoring. The WER performance of the ASR baseline is given once more in Table 8.3.

Table 8.3: The WER performance of the ASR baseline system on Dev 93, Test 92, and Test 93 sets.

	Dev 93	**Test 92**	**Test 93**
ASR 1-best	15.3%	10.2%	14.0%
Oracle on 100-best	8.3%	5.1%	7.3%

8.5.2 Re-scoring Experiments – A First Attempt

Re-scoring experiments are performed on the 100-best lists that are generated by the baseline ASR system. These 100-best lists are re-scored by

using the SELMs. In addition, we have trained an n-gram LM. All LMs are trained over the whole WSJ 87, 88, and 89 data with the vocabulary that is limited to the 20K open vocabulary for non-verbalized punctuation. The n-gram LM is a modified Kneser-Ney 5-gram model with singleton cut-offs (KN5). The KN5 model is built on words without any classes.

We have trained SELMs that use frames and targets as semantic context separately. The SELMs are also trained on the same data with the same vocabulary setting. The semantic features for each utterance in the training data are extracted by feeding them into the semantic parser SEMAFOR. The training data has 841 distinct frames and 17736 distinct targets. We have limited the number of frames and targets to the most frequent ones that cover the 80% of the training data, which results in 184 distinct frames and 1182 distinct targets. The SELMs use 200 word classes which are determined with respect to word frequencies and use up to 4-gram maximum entropy features with 10^9 connections. We have trained the SELMs by using the back-propagation through time (BPTT) algorithm [20] on the training data. We have used the reference transcription and the semantic context of the reference transcription of the Dev 93 set as the validation set for early stopping to avoid over-fitting.

The re-scoring experiments that use the SELMs are conducted by using the following setting. The semantic context for the utterance that will be re-scored can be extracted either from the reference transcription, oracle hypothesis, or the ASR 1st-best hypothesis. The experiments that use the semantic context of the ASR 1st-best hypothesis would reflect the actual performance of SELMs. The others can be used to observe a lower bound for WER. Therefore, we refer to the output of the semantic parser as follows. The output of the semantic parser (frames and targets) on the reference transcription are referred to as *reference frames and reference targets*. The output of the parser on the ASR output are referred to as

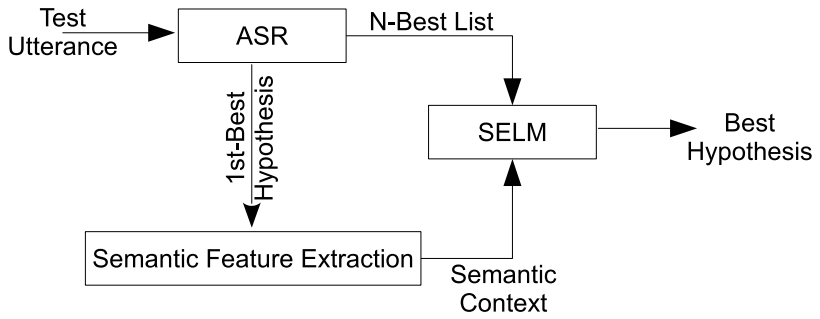

Figure 8.4: The diagram of re-scoring experiments for ASR frames and targets. The test utterance is passed through the baseline ASR. The ASR 1st-best hypothesis is given to the semantic feature extraction module, which extracts the semantic context for that utterance. The N-best list is re-scored by using the SELM with the semantic context for that utterance.

ASR frames and ASR targets. Finally, the output on the ASR oracle hypotheses are referred to as *oracle frames and oracle targets*. We present the results on reference frames/targets and on oracle frames/targets to present a lower bound on WER performance of the SELMs, the actual performance is given by the ASR frames/targets. The re-scoring procedure for ASR frames/targets is depicted in Figure 8.4.

Table 8.4: The WER performance of the modified Kneser-Ney 5-gram model and the SELMs. The bold WERs (ASR Frames and ASR Targets) present the actual performance. Results on the reference and oracle frames/targets are given to show a possible lower bound on WER.

Model	**Dev 93**	**Test 92**	**Test 93**
KN5	14.6%	9.7%	13.3%
SELM on Frames			
ASR Frames	**14.5%**	**9.5%**	**13.9%**
Reference Frames	13.4%	8.7%	12.3%
Oracle Frames	13.2%	8.7%	12.0%
SELM on Targets			
ASR Targets	**15.0%**	**10.0%**	**14.4%**
Reference Targets	12.9%	8.4%	11.7%
Oracle Targets	12.9%	8.4%	11.6%

The result of the re-scoring experiments are given in Table 8.4. As can be seen from the results, the SELMs performs significantly better than

the modified Kneser-Ney 5-gram model when accurate semantic context is used (as seen in *Reference Frames* and *Oracle Frames*). However, when the ASR 1st-best hypothesis is used due to the ASR noise the performance drops and SELM on Targets perform even worse than the n-gram model.

8.5.3 Error Pruning for a Better Semantic Context

The results in Table 8.4 show the potential performance of SELMs. When SELMs are supplied with accurate semantic context, their performance significantly improves. However, the noise on the ASR frames and targets drops their performance to an unacceptable range. Therefore to improve the actual performance, and to lower the noise on the semantic context, we have pruned the frames and targets that have high error rate on the ASR hypothesis. This error is computed on the ASR frames and targets with respect to the frames and the targets of the reference transcription. We have eliminated the frames and targets that have a more than 10% error rate of on the development set (Dev93). After the elimination, we have ended up with 60 distinct frames and 541 distinct targets. The SELMs are trained from scratch by using this subset of frames and targets and re-scoring experiments are repeated with these new SELMs. The performance of the these models is given in Table 8.5.

As can be seen in Table 8.5, pruning erroneous frames and targets lower the ASR noise and enable SELMs to perform better. Although, it improves the performance of the SELMs with the frames and the targets, the SELMs with targets do not perform well and they have a high performance gap between the evaluation with the ASR targets and with the reference targets. This gap is larger especially on the evaluation sets, which are not considered when pruning the errors. On the other hand, SELMs with frames benefit from pruning better. We do not observe a huge difference between the SELMs with ASR frames and reference frames after pruning. In ad-

Table 8.5: Improved WER performance of the SELMs by using low error frames and targets. The results show that by eliminating erroneous frames and targets, we can get significant improvements on WER with ASR frames and targets (given in bold). The SELM on Frames achieve 12% relative improvement on Test 92 evaluation set and 7% relative improvement on Test 93 evaluation set with respect to the ASR baseline.

Model	**Dev 93**	**Test 92**	**Test 93**
KN5	14.6%	9.7%	13.3%
SELM on Frames			
ASR Frames	**13.8%**	**9.0%**	**13.0%**
Reference Frames	13.6%	8.9%	13.0%
Oracle Frames	13.5%	8.9%	12.8%
SELM on Targets			
ASR Targets	**13.9%**	**9.5%**	**13.9%**
Reference Targets	13.7%	8.9%	13.1%
Oracle Targets	13.7%	8.9%	13.0%

dition, the SELMs on target are computationally more complex models to train since their context size is almost 9 times larger than the SELMs on frames with error pruning. Therefore, we continue to make further analysis only on the SELMs that use frames as the semantic context.

8.5.4 Further Analysis

We make further analysis by pruning erroneous frames at various error rates. Also we randomize the training data when training the SELMs for the stochastic gradient descent to converge to a better local minima. We compare the SELMs on frames with a RNNME model. The RNNME model is also trained with the same parameters, i.e., with 200 hidden layers and up to 4-gram n-gram features that uses 10^9 connections. The RNNME is also a class-based model that uses the same word classes and it is also trained with the same randomization of the training data, and initialized with the same random weights. We have pruned the frames which have an error rate more than a threshold. For this purpose, the following thresholds are used: 0%, 5%, 8%, 10%, 12%, 15%, 20%. Table 8.6 gives the number

of distinct frames after the pruning is performed.

Table 8.6: The pruning thresholds and the corresponding number of distinct frames in the development set for the WSJ corpus.

Pruning Error Rate	**No. of Frames**
0%	37
5%	41
8%	52
10%	60
12%	69
15%	78
20%	92

We have trained SELMs on frames with the given pruning threshold and we have re-scored the n-best lists. The WER performance of the SELM on Dev93 with various thresholds of error pruning is given in Figure 8.5. Also the WER performance of RNNME is given. As can be seen, when the reference frames are used SELMs perform better than the RNNME model. However, the ASR noise reduces the performance of SELMs when the ASR frames are used. The error rate for the SELMs with ASR frames has a general tendency to increase as the pruning error rate increases. In addition, we can see that the difference between the performance of ASR frames and reference frames increases as the pruning error rate increases. We observe that for the given pruning thresholds, the SELM performs better than RNNME for 0% and 8%.

We present the performance of SELMs with the thresholds 0% and 8% also on the test set in Table 8.7. We do not present the results with the "oracle frames" since they give lower bounds similar to the "reference frames" on the WER performance of the SELMs. The linear interpolation of the two SELMs are also presented. The linear interpolation is applied with equal weights without performing any optimization.

We can see from the results in Table 8.7 that the SELMs with error pruned frames can achieve slight improvement in WER over the RNNME

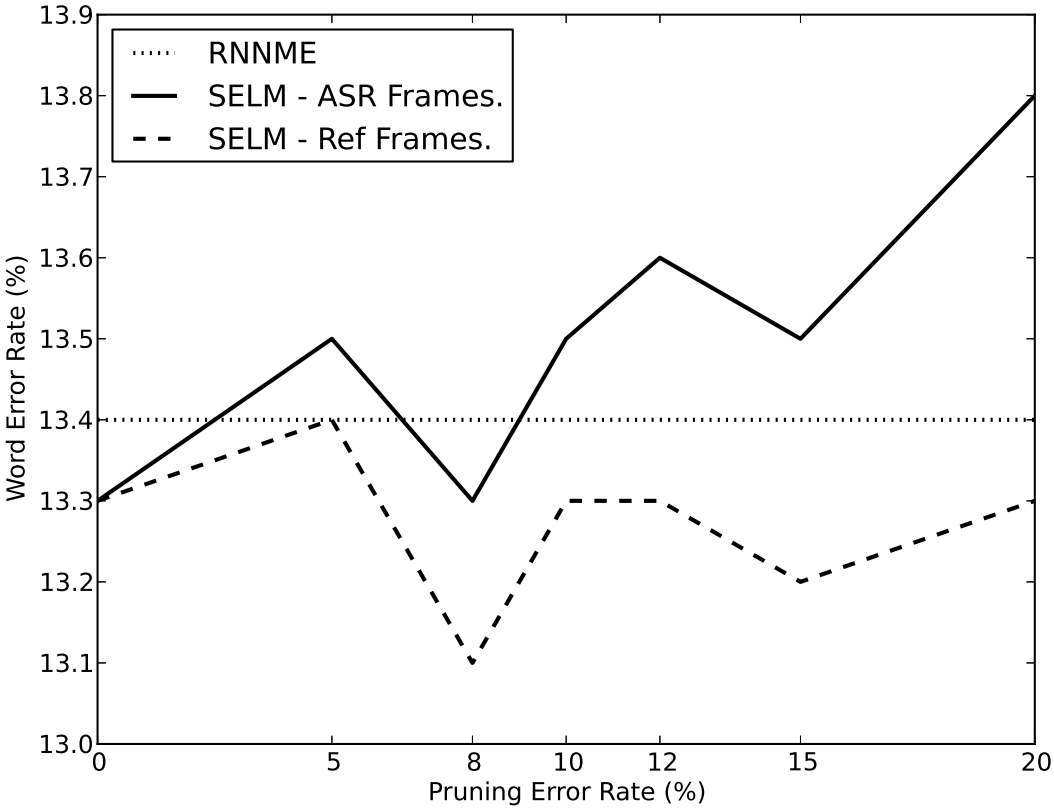

Figure 8.5: The WER performance of SELM on frames on the Dev93 set with different pruning error rates.

for Dev93, when the error pruning is applied. For the unseen test set (Test92, Test93) we do not observe any consistent improvement. Therefore, error pruning does not perform well for the unseen data. One way to improve the performance of SELMs is to combine these models at different pruning thresholds. The linear interpolation[5] of the two SELMs perform consistently better than the RNNME model on all the data sets. Therefore, we see that linear interpolation can be a choice to combine the performance of different semantic contexts. In the next chapter, we will also address the understanding performance of these models.

[5]The linear interpolation uses equal weights in this case.

Table 8.7: The WER performance of the SELMs with pruning thresholds of 0% and 8%. The linear interpolation of the two SELM model with equal weights is also presented. The results in bold show the actual performance of SELMs.

Model	**Dev 93**	**Test 92**	**Test 93**
KN5	14.6%	9.7%	13.3%
RNNME	13.4%	8.8%	12.7%
(1) SELM Err = 0			
ASR Frames	**13.3%**	**8.7%**	**12.7%**
Reference Frames	13.3%	8.7%	12.7%
(2) SELM Err < 8%			
ASR Frames	**13.3%**	**8.9%**	**12.7%**
Reference Frames	13.1%	8.8%	12.6%
(1)+(2) Linear Int.			
ASR Frames	**13.2%**	**8.7%**	**12.1%**
Reference Frames	13.2%	8.7%	12.0%

8.6 LUNA Human-Human Experiments

In this section, we present the performance of SELMs on the Italian LUNA HH corpus [41]. As described in Chapter 6, LUNA HH corpus is composed of human-human dialogs that are recorded by a hardware/software technical support company. The nature of human-human speech has a negative impact on WER. Hence, the ASR performance is low, and ASR 1st-best hypothesis contains more errors compared to the ASR hypotheses we have used in WSJ experiments. The LUNA HH corpus is recorded with a single channel, therefore, there are overlapping speech segments and these segments are discarded.

The LUNA frame-semantic parser [29] is built on the semantic annotations on the LUNA HH corpus. The frame-semantic annotations on LUNA are in the domain of hardware/software technical support, therefore, the frames the parser identifies are domain dependent. This creates more sparse features compared to WSJ.

We investigate the performance of SELMs by re-scoring experiments under these two problems related to LUNA HH corpus, i.e., high error

rates on the ASR hypotheses, and sparse semantic features.

8.6.1 ASR Baseline

The ASR baseline for LUNA HH is constructed by using Kaldi [105] speech recognition toolkit. The baseline system is described in Section 6.2.1 and its performance is given once more in Table 8.8.

Table 8.8: The WER performance of the baseline ASR on LUNA HH corpus. The oracle performance is given for 100-best lists.

	Dev	**Test**
ASR	37.7%	36.7%
Oracle on 100-best	25.3%	24.4%

8.6.2 Re-scoring Experiments

The re-scoring experiments are performed by using the same setting as the WSJ experiments. Therefore, 100-best lists are re-scored by using SELMs trained over the frames and the targets that are extracted by using the LUNA frame-semantic parser. We have removed the frames and targets that just occur once in the training data and we have used 128 distinct frames and 508 distinct targets. The SELMs use a hidden layer of size 100 and use up to 3-gram maximum entropy features that are implemented by 10^8 connections. Since the vocabulary size is small, no word classes are used. In addition, an RNNME model is trained with the same settings. All neural network models are initiated with the same random weights. The results of the re-scoring experiments are given in Table 8.9. In this table, "ASR Frames/Targets" means that we have given the ASR 1st-best hypothesis to the semantic parser, and "Reference Frames/Targets" means that we have given the reference transcription to the semantic parser. As in the WSJ experiments, the actual performance is reported with "ASR

Frames/Targets". "Reference Frames/Targets" shows a lower bound to WER. We omit the results with oracle hypotheses, since they behave similar to the reference transcriptions.

Table 8.9: The WER performance of the re-scoring experiments on LUNA HH corpus by using the SELMs that are trained over frames and targets. KN3 is the LM that ASR uses therefore it gives the ASR baseline. The bold results ("ASR Frames/Targets") show the actual performance of the SELMs.

Model	**Dev**	**Test**
KN3	37.7%	36.7%
RNNME	36.8%	35.5%
SELM on Frames		
ASR Frames	**37.3%**	**36.2%**
Reference Frames	35.0%	33.9%
SELM on Targets		
ASR Targets	**37.3%**	**36.0%**
Reference Targets	35.4%	34.0%

We observe a similar situation for LUNA. When the accurate semantic information ("Reference Frames/Targets") is used, the SELMs have a significant improvement on WER. However, although the SELMs with "ASR Frames/Targets" perform better than the ASR baseline, they fail to outperform the RNNME model.

8.6.3 Error Pruning

We perform error pruning similar to the WSJ experiments. For LUNA HH corpus, we also prune the erroneous targets together with the erroneous frames. The thresholds for pruning are 10%, 20%, and 30%. The results are presented in Table 8.10.

The results show that error pruning also improve the results compared to training with all the semantic features. Although, the best performing SELMs have a performance close to RNNME, they fail to outperform RNNME. The interpolation of the best model on frames with the best model

Table 8.10: The WER performance with error pruning at 10%, 20%, and 30% on LUNA HH corpus. The number of distinct frames and targets with the corresponding thresholds are given in parenthesis.

Model	**Dev**	**Test**
RNNME	36.8%	35.5%
SELM on Frames - Err. < 10% (12 Frames)		
ASR Frames	**36.9%**	**35.5%**
Reference Frames	36.9%	35.4%
SELM on Frames - Err. < 20% (20 Frames)		
ASR Frames	**37.3%**	**35.8%**
Reference Frames	37.1%	35.7%
SELM on Frames - Err. < 30% (28 Frames)		
ASR Frames	**37.1%**	**35.7%**
Reference Frames	37.1%	35.5%
SELM on Targets - Err. < 10% (88 Targets)		
ASR Targets	**37.0%**	**35.5%**
Reference Targets	36.9%	35.5%
SELM on Targets - Err. < 20% (104 Targets)		
ASR Targets	**37.1%**	**35.8%**
Reference Targets	37.0%	35.5%
SELM on Targets - Err. < 30% (124 Targets)		
ASR Targets	**37.0%**	**35.6%**
Reference Targets	36.9%	35.3%

on targets (not reported in the table) do not bring any improvements over the individual models. The high error rate on the semantic information and the sparse semantic features prevents SELMs to be totally optimized.

8.7 Discussion

In this chapter we have introduced SELMs that are based on the theory of frame semantics. SELMs solve the locality problem by considering long-range semantic dependencies. SELMs use frames that are evoked and the targets that occur in a sentence as features. For this purpose, they rely on the output of frame-semantic parsers.

The perplexity results on Penn-Treebank are encouraging, although as

we have argued they are biased. A better assessment of SELMs are made by performing n-best re-scoring experiments over WSJ speech corpus and LUNA HH corpus. WSJ corpus consists of read news articles and therefore, ASR baseline performance is acceptable. In WSJ experiments, we have shown that a slight improvement compared to RNNMEs[6] can be achieved. On LUNA HH corpus, the ASR baseline has a high WER. This affects the performance of SELMs and prevents them to be optimized as well as the SELMs that are built for WSJ corpus.

The semantic context is based on the ASR hypothesis and suffers from the ASR noise. To reduce the ASR noise on the semantic context we propose to do error pruning. We have achieved a slight improvement over RNNME with error pruning for the WSJ corpus. However, although error pruning improves the performance of the SELMs, they cannot outperform RNNMEs when there is too much noise in the ASR hypothesis as observed in LUNA HH experiments.

[6]RNNMEs are the basic building blocks of SELMs.

Chapter 9

Deep Encodings for Semantic Language Models[1]

The most important problem related to SELMs presented in the previous chapter is the fact that they rely on the semantic information that is extracted from the output of ASR. Therefore, the ASR noise plays a crucial role on the performance of SELMs. In addition to the ASR noise, the automatic frame semantic parser also introduces noise which prevents SELMs to fully capture the linguistic scene. One solution to this problem, as introduced in the previous chapter, is to prune the erroneous frames on a held-out dataset. However, this results in the loss of full semantic context and does not perform well on unseen data. Because of these restrictions, the full power of SELMs cannot be utilized.

SELMs are important because as it has been argued multiple times [108, 127] improving WER does not always yield an improvement on the understanding performance. This is especially important for spoken language systems which extracts semantic structures out of the utterances. Therefore, it is important to optimize LMs both in terms of recognition and understanding performance.

The performance of ASR can be improved by re-scoring an n-best list

[1]The work presented in this chapter is the extended and the revised version of the publication [8].

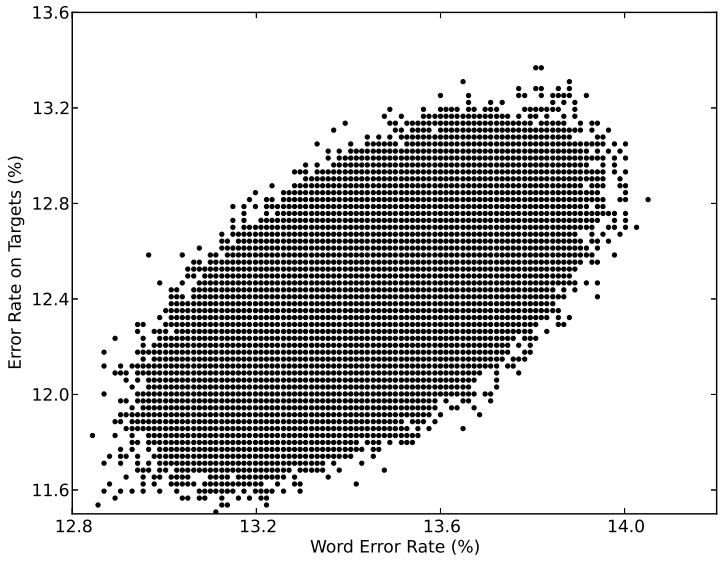

Figure 9.1: Scatter plot of transcription performance (WER) versus understanding performance (TER) for random selections of hypotheses from the 100-best list of the development set of Wall Street Journal corpus.

of hypotheses by using a more advanced LM than the one that is used for decoding. There may be various ways to select the hypotheses during re-scoring. Figure 9.1 shows the transcription versus the understanding performance for possible different selections of hypotheses. We measure the understanding performance by target error rate (TER), which is calculated from the errors made on target words[2], which are the main meaning bearing elements of semantic frames. If the sole purpose of improving the performance is to optimize with respect to the transcription performance (WER), one may not improve the understanding performance (TER). Hence, LMs that would be used for re-scoring must be built to jointly optimize the transcription and the understanding performance.

SELMs play an important role in this joint optimization, because they are trained by considering the semantic constraints. SELMs do not consider sentences only as sequences of symbols, but they consider the linguis-

[2]TER is simply equal to the WER on the target words.

tic scene, i.e., the semantic context which these words occur. In this way, they incorporate high level semantic information in the model. This high level information plays a crucial role in optimizing LMs both for recognition and understanding jointly. For this task, a *fully* semantics-aware LM would be a better choice than an LM which semantic information is pruned to get rid of erroneous frames. Hence, the whole semantic context of the utterance can be taken into consideration and unseen data can be handled more robustly. This chapter introduces the use of deep autoencoders that encode the semantic context with a noisy representation, which would not be significantly affected from the noise introduced by the ASR or the semantic parser.

9.1 Deep Autoencoders for Encoding Semantic Context

Hinton and Salakhutdinov [58] have shown that deep autoencoders can reduce the dimensionality of data with higher precision than principle component analysis. In addition, the authors have observed that for document similarity tasks deep autoencoders outperform latent semantic analysis. Therefore, high-dimensional low-level semantic context, which contains more than 1K target vocabulary, can be represented in a low-dimensional space with a high-level representation by means of autoencoders. Deep denoising autoencoders are also shown to learn high-level representations of the input. These high-level representations rather than hand-built features improve the performance of digit recognition systems [126].

In addition to dimensionality reduction, SELMs suffer from the noise introduced by the ASR and the frame semantic parser. We have shown that the accurate semantic information[3] improves the performance of SELMs

[3]The semantic parse of the reference transcription.

significantly. We propose to handle the noise in the semantic representation by building noisy representations of the semantic context. We have used the method of *semantic hashing* [113] for this purpose. Semantic hashing is a method in document retrieval that maps documents to binary vectors such that the Hamming distance between the binary vectors represents the similarity of documents. In this way, documents can be stored in the memory address represented by the binary vector and can be retrieved in a fast manner. A binary vector that is used in semantic hashing [113], compared to a continuous vector, introduces noise to the high-level representation of the document. For that reason, it is suitable to be used as a noisy representation of semantic context for utterances. This section describes how the training of deep autoencoders is performed for obtaining deep semantic encodings for utterances.

We employ 4-layer autoencoders for our experiments. The first layer is the input layer, which is activated by the bag-of-words representation of the frames or targets for an utterance. The final layer, encodes these representations to a binary vector of desired size by means of stochastic binary units. Therefore, the autoencoders have 2 hidden layers of the same size. The structure of the deep autoencoders is depicted in Figure 9.2.

9.1.1 Training Deep Autoencoders

Training deep neural networks that have more than one hidden layer by randomly initializing the weights, and then using gradient descent to learn the weights, converges to a poor local minima [58]. However, if a good initialization is done on the weights that is close to a good solution, gradient descent can converge to a good solution. For that purpose, the training of deep autoencoders are performed in two phases. The first phase of training deep neural networks is the unsupervised pretraining step that finds a good initialization of weights by greedy layer-by-layer training [61].

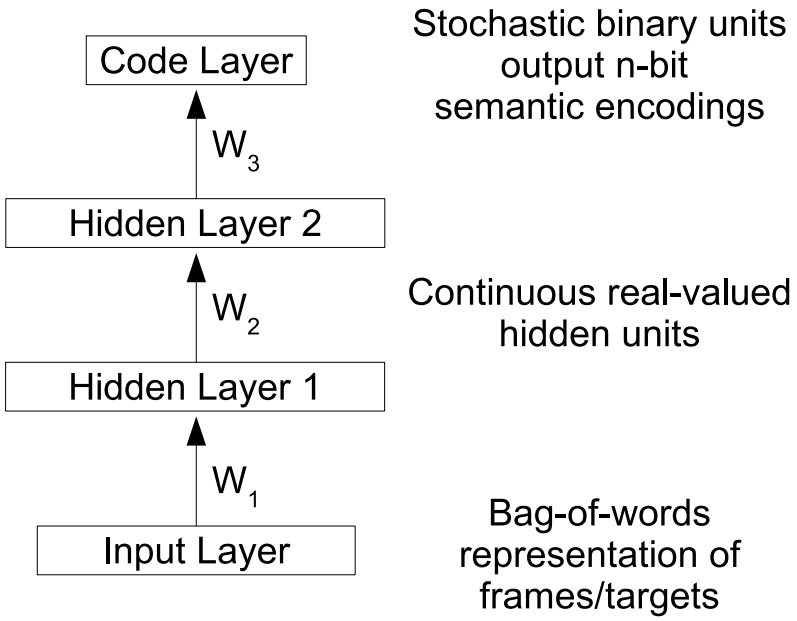

Figure 9.2: The structure of the 4-layer deep autoencoder used for semantic encoding. The input layer represents the input with a bag-of-words model. The output layer outputs n-bit binary encodings by using stochastic binary units. The hidden layers are continuous valued units. The weights between layers are represented with W_i.

At the second phase, a standard backpropagation algorithm can be used to train deep neural networks in a supervised way [58].

We train deep autoencoders for semantic encodings of frames and targets separately. The input given to the deep autoencoders is the normalized "bag-of-words" (BoW) vectors of frames for frame encodings, and BoW vectors of targets for target encodings. The deep autoencoder constructs n-bit encodings from these BoW vectors.

Unsupervised pretraining

The first phase is the unsupervised pretraining phase as shown in Figure 9.3. For this purpose, the greedy layer-by-layer training [61] is performed. In this approach, each pair of layers is modeled by restricted Boltzmann machines (RBMs) and each RBM is trained from bottom to

top. During the pretraining phase, the visible layer of the bottom RBM (RBM 1) is modeled by a constrained Poisson model as given in [113]. Therefore, unnormalized BoW vectors are used as the input only when the activations of the hidden layer are computed. The softmax activation function is used for the reconstruction of the input as the normalized BoW vector. The other RBMs use the sigmoid function as the activation function. The network is pretrained by using the single-step contrastive divergence CD_1 [60].

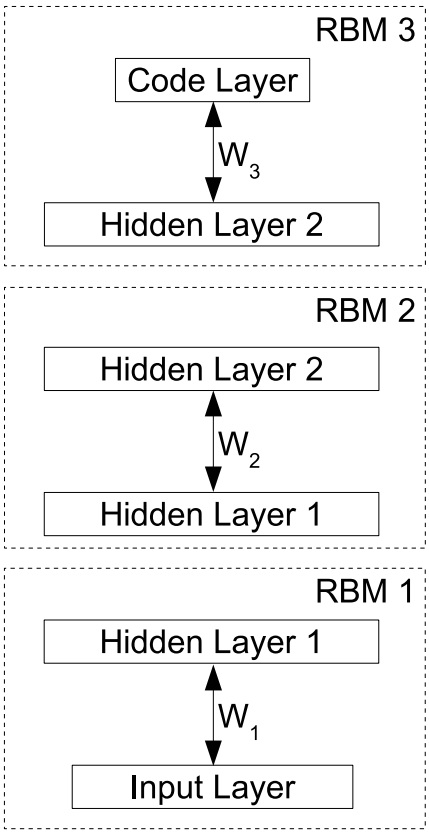

Figure 9.3: The unsupervised pretraining procedure. Each pair of layer is considered as a restricted Boltzmann machine and the whole network is trained in a greedy layer-by-layer approach.

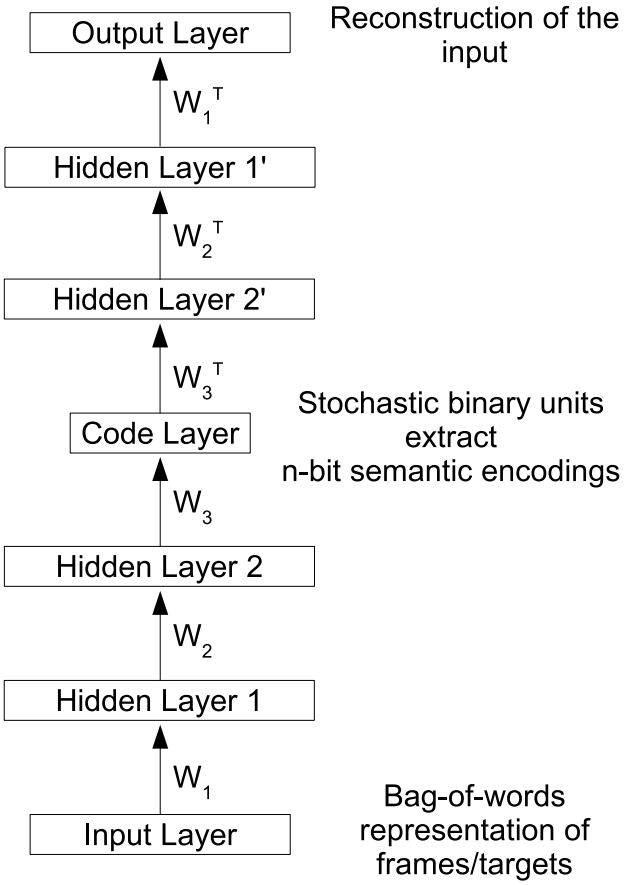

Figure 9.4: The fine-tuning phase of the deep autoencoder. The autoencoder unrolled such that W_i^T corresponds to the transpose of the W_i weight matrix. The output layer reconstructs the normalized BoW representation of the input. Stochastic binary units are used to enforce binary encodings. The network is fine-tuned by using the backpropagation algorithm over the reconstruction error.

Fine-tuning

In the second phase, the network is unrolled as shown in Figure 9.4. Here, W_i^T corresponds to the transpose of the W_i weight matrix, so that the network reconstructs the input at the output layer. The output layer uses the softmax function and reconstructs the normalized BoW input vector, the other layers use the sigmoid activation function. The backpropagation algorithm is used to fine-tune the weights by using the reconstruction error

at the output layer. The codes at the "code layer" are made binary by using stochastic binary units at that layer, i.e., the state of each node is set to 1 if its activation value is greater than a random value that is generated at run time; or set to 0 otherwise. This state value is used for the forward-pass. However, when backpropagating the errors, the actual activation values are used. After training the autoencoder, deep semantic encodings are obtained by using only the bottom part of the network which corresponds to the deep autoencoder given in Figure 9.2.

9.2 SELM Structure

The SELM structure that uses semantic encodings has the structure that is depicted in Figure 9.5. This structure is different than the one presented in Chapter 8, because the semantic context used in this case is a high-level hidden representation rather than a low-level representation of the frames and targets. For this reason, we have removed the connections between the semantic context and the hidden layer. Therefore, semantic encodings are directly fed into the output layer. The SELMs we have used in this chapter also have a class-based implementation that estimates the probability of the next word by factorizing them into class and class membership probabilities. The current word is fed into the input layer by 1-of-n encoding. The semantic layer uses the semantic encoding for the current utterance. SELMs are trained by using the backpropagation through time algorithm, which unfolds the network for N time steps back for the recurrent layer and updates the weights with the standard backpropagation [96]. SELMs also use n-gram maximum entropy features which are implemented as direct connections between n-gram histories and the output layer. The implementation applies hashing on the n-gram histories as given in [91].

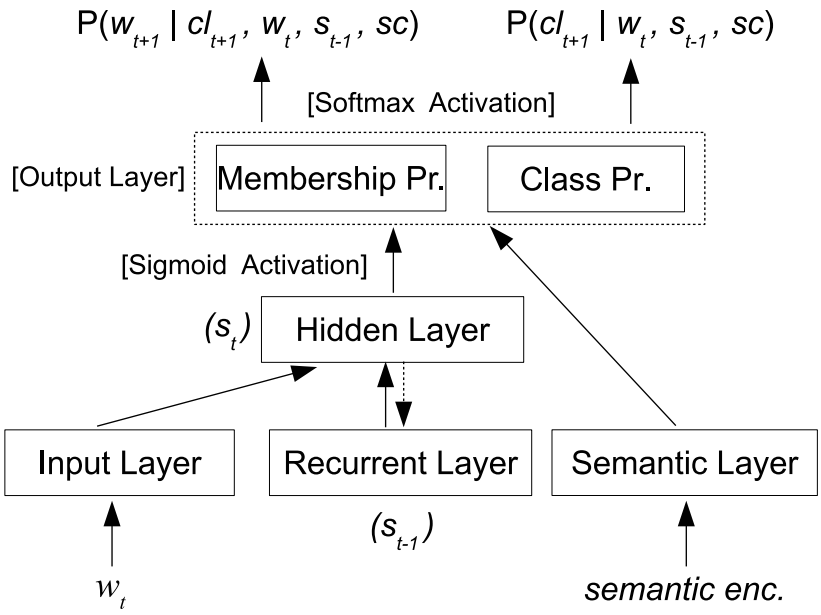

Figure 9.5: The class-based SELM structure for deep semantic encodings. The network takes the current word w_t and the semantic encoding for the current utterance as input. The output layer estimates the probability for the next word w_{t+1} factorized into class probabilities and class-membership probabilities (cl_{t+1} denotes the recognized class for the next word). The direct connections for the n-gram maximum entropy features are not shown.

9.3 Wall Street Journal Experiments

We have performed n-best re-scoring experiments on the Wall Street Journal (WSJ) corpus to assess the performance of these models on the recognition performance and the understanding performance. Originally, the WSJ corpus is designed for the speech recognition task. Therefore, the corpus does not contain any semantic annotation. To assess the understanding performance we use target error rate (TER), which measures the errors made on target words. The WSJ corpus is not annotated also for target words, therefore as a gold standard we have used the output of the SEMAFOR semantic parser on the reference transcription of the utterances. Although this would not give us an exact evaluation, we would have a good approximation to assess the understanding performance.

We have used the same training, development, and the evaluation set that is presented in Section 6.3 the re-scoring experiments are performed on ("Dev93" - 503 utterances), ("Test92" - 333 utterances), and ("Test93" - 213 utterances) which are defined in Section 6.3.

9.3.1 Experimental Setting

We use the same ASR baseline that is described in Section 6.3.1. The performance of the ASR baseline is presented here once more in Table 9.1.

Table 9.1: The WER performance of the ASR baseline system on Dev 93, Test 92, and Test 93 sets.

	Dev 93	**Test 92**	**Test 93**
ASR 1-best	15.3%	10.2%	14.0%
Oracle on 100-best	8.3%	5.1%	7.3%

The re-scoring experiments are performed on the 100-best lists that are obtained from the ASR baseline system. We have re-scored these 100-best lists by using the SELMs that are trained on frames and targets separately. In addition we have trained a RNNLM model and a 5-gram model with modified Kneser-Ney smoothing with singleton cut-offs. All models are trained on the whole WSJ 87, 88, and 89 data with the vocabulary that is limited to the 20K open vocabulary for non-verbalized punctuation. Therefore, the LMs used for re-scoring also include a 5-gram modified Kneser-Ney model with singleton cut-offs (KN5), a RNNLM model that has 200 nodes in the hidden layer and uses a maximum-entropy model that has 4-gram features with 10^9 connections (RNNME). RNNME uses 200 word classes that are constructed based on the frequencies of words, however the KN5 does not contain any classes.

The SELMs are trained by using either frame encodings or target encodings that are obtained with the relevant autoencoders for various code lengths. The SELMs have the same configuration with the RNNME model,

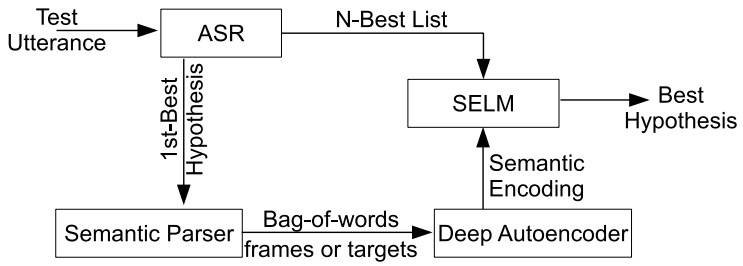

Figure 9.6: The SELM re-scoring diagram. The test utterance is fed into the ASR. The 1st-best ASR hypothesis is passed through the semantic parser and BoW features are given to the autoencoder for extracting semantic encodings for the test utterance. The n-best list is re-scored by using the SELM that uses the semantic encoding as the semantic context for the test utterance.

i.e., they have 200 nodes in the hidden layer and use a maximum-entropy model that has 4-gram features with 10^9 connections. They also use the same word classes. All NNLMs (RNNME and SELMs) are initialized with the same random weights to make the experiments more controlled. In addition to that, the training of all NNLMs are done by using the same randomization of the training data. Since the training data is randomized, we have built independent sentence models, i.e., the state of the network is reset after each sentence. Dev93 is used for adjust the learning rate and for early stopping.

The flow of the re-scoring experiments for SELMs is shown in Figure 9.6. The ASR 1st-best hypothesis is passed through SEMAFOR to extract frames and targets, then deep semantic encodings are obtained by feeding them into the relevant autoencoder.

9.3.2 Deep Semantic Encodings

The deep semantic encodings are constructed for the frames and targets that evoke these frames separately. Therefore, we have two different semantic encodings, one for the frames evoked and one for the targets that occur in the utterance. The autoencoders are trained over the whole training

data that is available for LM training, in addition, Dev93 development set is used to avoid overfitting. To obtain the frames and the targets for these data sets, the reference transcriptions are passed through the SEMAFOR frame-semantic parser, and frames and targets are extracted. Then BoW vectors of the frames and targets are created. Unsupervised pretraining is performed for 20 iterations with a mini-batch size of 100 over the BoW vectors of the frames and the targets on the training data. Then fine-tuning is performed by using stochastic gradient descent over the training data also by considering the reconstruction error on the development set (Dev93) to avoid overfitting by adjusting the learning rate and by "early stopping".

We have used the most frequent frames and targets that cover the 80% of the training data. Therefore, we use 184 distinct frames and 1184 distinct targets. The semantic encodings for frames are constructed by using a deep autoencoder of size $(184 - 200 - 200 - n)$ and for targets $(1184 - 400 - 400 - n)$, where n denotes the size of the binary semantic encoding.

9.3.3 The Accuracy of Semantic Encodings

To investigate the accuracy of encodings of various sizes, we compare ASR encodings with reference encodings. Therefore, we treat reference encodings as gold standards and assess the accuracy of ASR encodings with respect to them. Since semantic encodings are binary vectors, we can use the Hamming distance between the ASR and the reference encodings to determine how accurate they are. Hamming distance measures at how many bits the two representations differ. On the development set we plot the histogram of Hamming distance between the ASR and the reference encodings.

Figure 9.7 shows the histograms of frame encodings in the interval $[1, 8]$[4].

[4]We do not present the full interval, $[0, 24]$ for clear visualization. The interval $[1, 8]$ covers the critical part of the distributions.

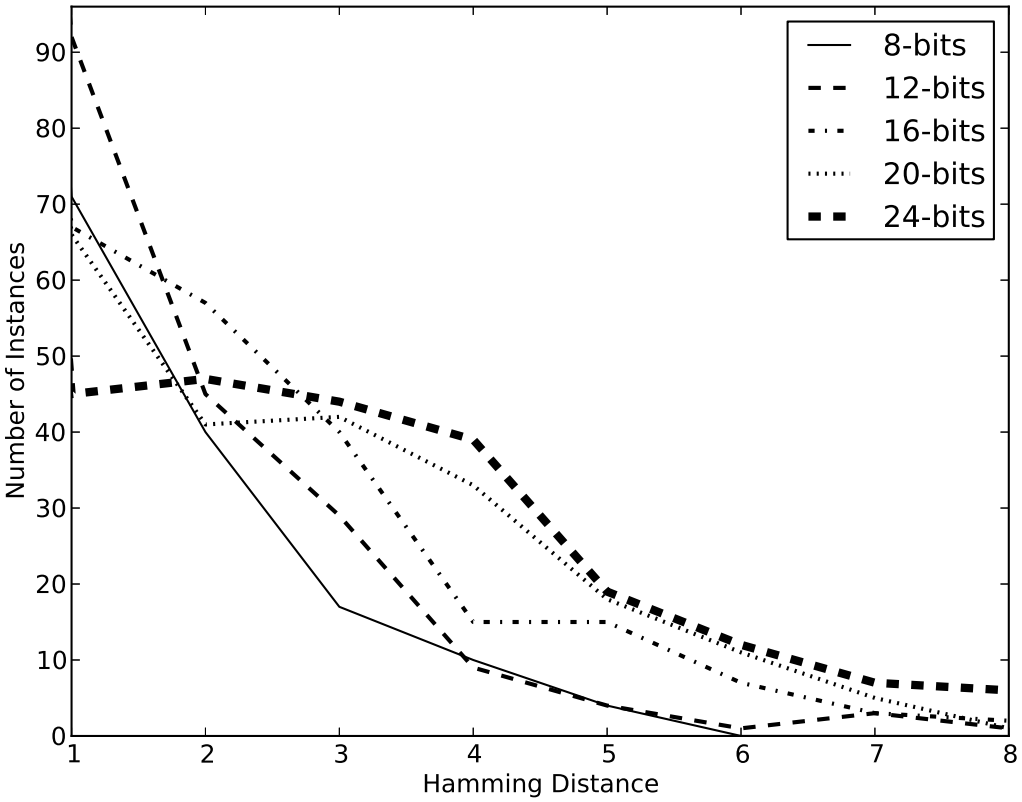

Figure 9.7: The histogram of Hamming distance between ASR frame encodings and reference frame encodings on the development data. The histogram is shown for the Hamming distance in the interval $[1, 8]$.

As can be seen, as the size of the increase the distribution moves to the right, i.e., the proportion of encodings with higher number of errors increases. We can see that, sizes of 8-bits and 12-bits have a similar distribution which has a better performance of suppressing the ASR noise. However, especially the sizes of 20-bits and 24-bits are very much affected by the ASR noise and show high discrepancy between ASR and reference encodings.

The histograms for target encodings are given in Figure 9.8. It is clearly seen that size of 8-bits has the lowest error distribution and size of 24-bits

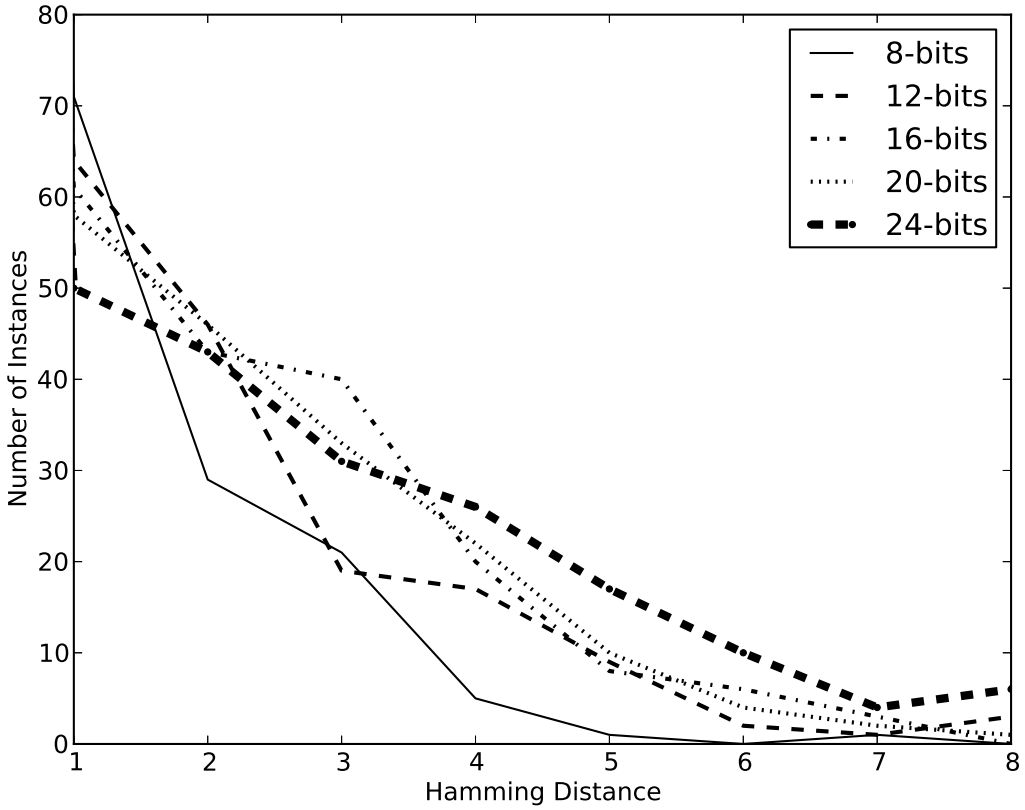

Figure 9.8: The histogram of Hamming distance between ASR target encodings and reference target encodings on the development data. The histogram is shown for the Hamming distance in the interval $[1, 8]$.

has the highest error distribution. The others, sizes of 12-bits, 16-bits, and 20-bits show almost a similar distribution. If both frame and target histograms are compared, target encodings are expected to be more robust to ASR noise, since they have a better error distribution (when comparing the same sizes of frame and target encodings with each other).

We also present the performance of these encodings on the target error rate (TER) and WER space by performing re-scoring experiments on the development set. Figure 9.9 shows that both for frame encodings and target encodings, semantic encodings of size 12-bits perform the best. The

semantic encodings of size 20-bits and 24-bits are not reliable since they perform worse than RNNME for target encodings. 8-bit and 16-bit encodings have a better joint performance than RNNME. As seen in the histograms, 8-bit encodings are more robust to noise than other encodings. However, the SELMs using these encodings perform worse than the SELMs that use 12-bit encodings. Therefore, it is possible that 8-bit encodings are not sufficient to capture the semantic context on the WSJ corpus.

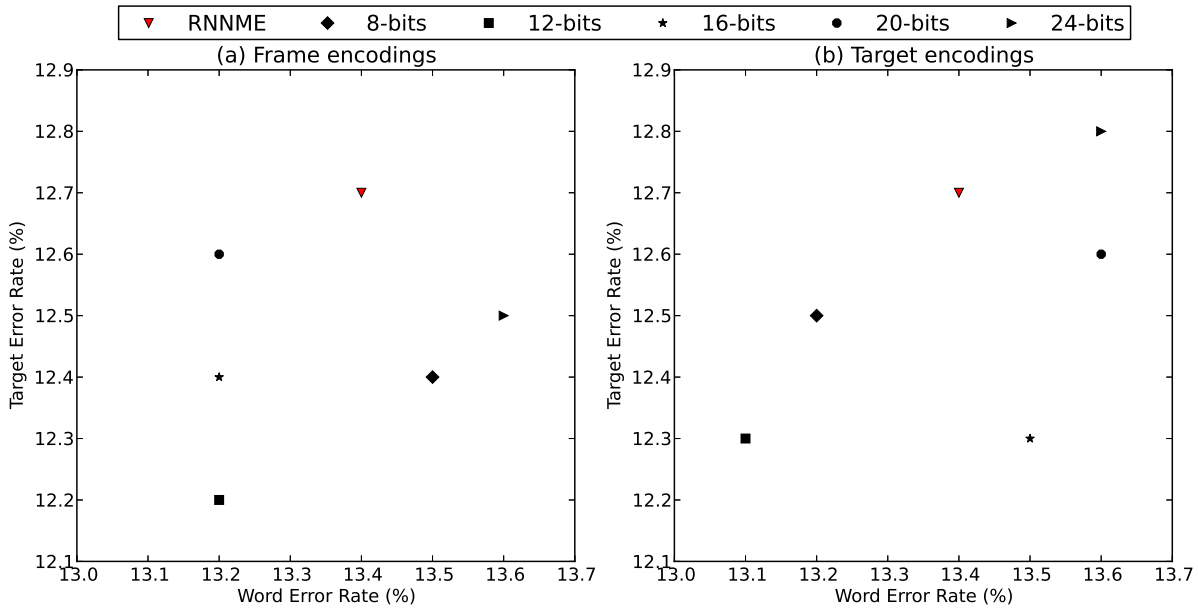

Figure 9.9: The performance of the SELMs on the TER-WER space with various size of frame and target encodings.

9.3.4 Re-scoring Experiments

In this section, we present the WER and TER performance of all of the models. These models include the KN5 and RNNME model for comparison, SELMs with all sizes of frame and semantic encodings. In addition, we present the results on the SELMs that use error pruned frames that are given in the previous chapter. All of the results are presented in Table 9.2.

Table 9.2: WER and TER performance of all of the models. The performance of SELMs with all of the sizes of frame and target encodings. The best performing SELMs on "Dev93" is given in bold. "DevP" refers to perplexity on "Dev93" In addition the SELMs that use error pruning are given ("SELM - Frame Err Pruning") are presented

Model	**DevP**	**Dev93**		**Test92**		**Test93**	
		WER(%)	**TER(%)**	**WER(%)**	**TER(%)**	**WER(%)**	**TER(%)**
KN5	146.0	14.6	13.4	9.7	10.4	13.3	13.2
RNNME	117.2	13.4	12.7	8.8	9.6	12.7	12.6
SELM - Frame Enc.							
8-bits	113.0	13.5	12.4	8.5	9.1	12.4	12.6
12-bits	**111.7**	**13.2**	**12.2**	**8.6**	**9.3**	**13.0**	**13.2**
16-bits	109.7	13.2	12.4	9.1	9.7	12.6	13.1
20-bits	110.8	13.2	12.6	8.5	8.9	12.5	12.6
24-bits	111.7	13.6	12.5	9.2	10.1	12.8	13.3
SELM - Target Enc.							
8-bits	112.5	13.2	12.5	8.5	8.9	12.4	12.6
12-bits	**110.6**	**13.1**	**12.3**	**8.5**	**9.3**	**12.4**	**12.4**
16-bits	113.6	13.5	12.3	8.7	9.2	12.5	12.6
20-bits	111.4	13.6	12.6	9.0	9.7	12.7	12.8
24-bits	118.8	13.6	12.8	8.8	9.4	13.1	13.2
SELM - Frame Err Pruning							
Err = 0	108.6	13.3	12.4	8.7	9.3	12.7	12.6
Err < 8%	101.1	13.3	12.4	8.9	9.5	12.7	12.6

We can see in Table 9.2 that the SELM with 12-bit target encodings always performs better than RNNME. The SELM with 12-bit frame encodings, on the other hand, performs worse than RNNME only for Test93. We also see that frame encodings are more noisy, since we observe that for each data set a different encoding performs the best; but for target encodings this is more stable, the 8-bit and 12-bit target encodings perform consistently better than other sizes. This result is consistent with the error analysis on the encodings, i.e., target encodings are more robust to noise.

We also see that the SELMs that use error pruned frames perform well in TER and better than RNNME except for the Test93 set. Therefore, we can see that error pruning also helps for the understanding performance.

9.3.5 Understanding Performance

In this section, we make deeper analysis on the understanding performance of the SELMs. For this purpose, we compute TER for the most frequent targets that cover the 20%, 40%, 60%, 80%, and 100% of the training data. To present statistically significant results, we combine all the data set (Dev93, Test92, and Test93) and make the error analysis on the combination[5]. We take the SELMs that have the best joint performance[6] on the development set. Therefore, we present results for the SELM with 12-bit frame encodings, the SELM with 12-bit target encodings, and the SELM with error pruning with a threshold of 0%. In addition, we present the results on RNNME to make a comparison.

We also make analysis on how much the ASR noise degrades the performance of SELMs. For this purpose, we present the results with both semantic encodings of the ASR hypothesis ("ASR encodings") and with the semantic encodings of the reference encodings ("Reference encodings").

Figure 9.10 shows the detailed understanding performance of reference frame and target encodings. For the SELM that uses error pruned frames, ASR and reference frames do not differ, since all erroneous frames are pruned. Hence, ASR and reference frames are the same. We can see that reference frame and target encodings have a similar performance and they perform significantly better than RNNME. We also see that, the SELM with error pruning performs consistently better than RNNME.

Figure 9.11 shows the detailed understanding performance for ASR encodings. We see that the performance of both frame and target encodings degrade because of the ASR noise. We observe that the frame encodings are affected more than the target encodings. Target encodings perform

[5]When TER analyzed on a single set the coverage below 60% does not make much sense.

[6]The joint performance may be simply given as the sum of WER and TER, considering both understanding and recognition have the same importance.

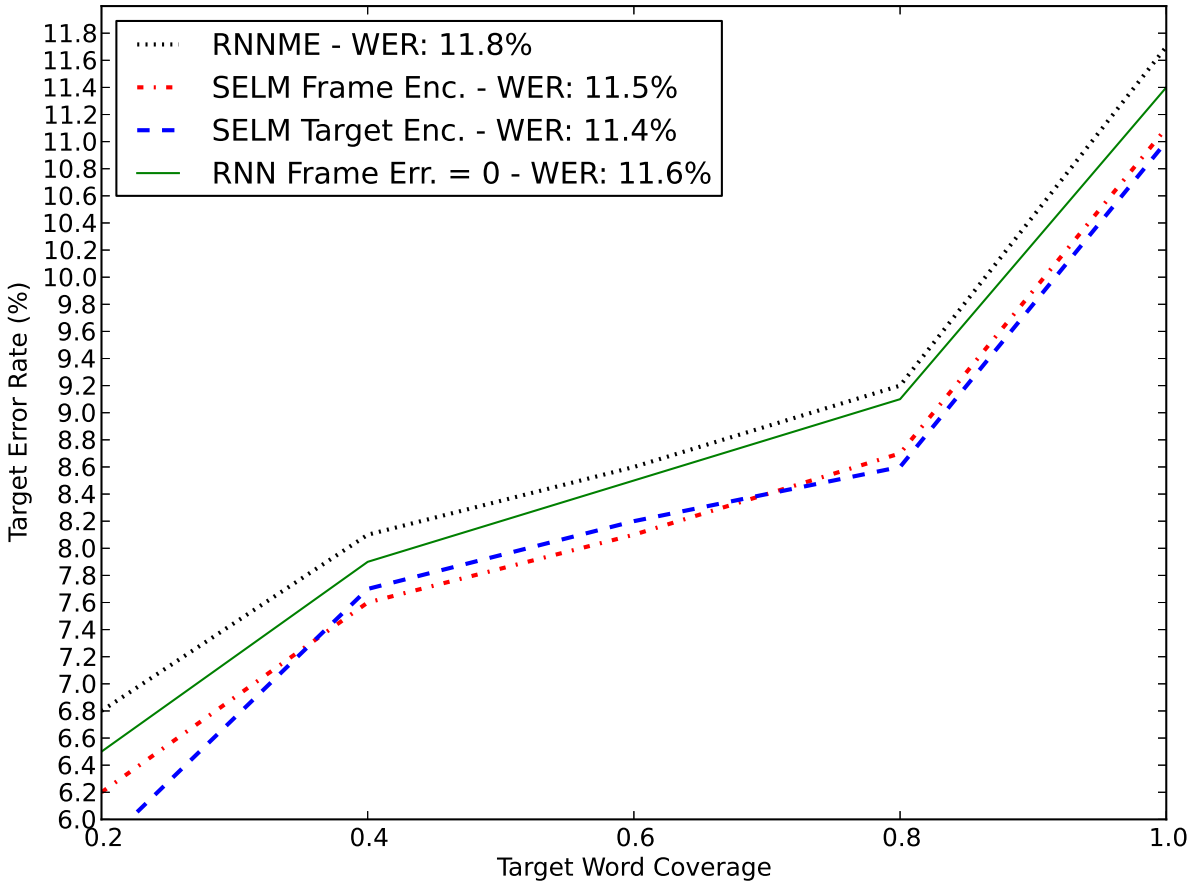

Figure 9.10: Target error rate at different coverages of target words for reference encodings.

consistently better than RNNME even with ASR encodings. Therefore, they are more robust to noise.

9.3.6 Combination of Models

We combine the best performing SELMs on "Dev93" by using linear interpolation with equal weights. We refer to the SELM with 12-bit frame encodings as "SELM - Frame Enc.", the SELM with 12-bit target encodings as "SELM - Target Enc.", and the SELM that prunes every erroneous frame (threshold at 0%) as "SELM - Err. Prune". We linearly interpolate the two SELMs with semantic encodings. And also combine all of the

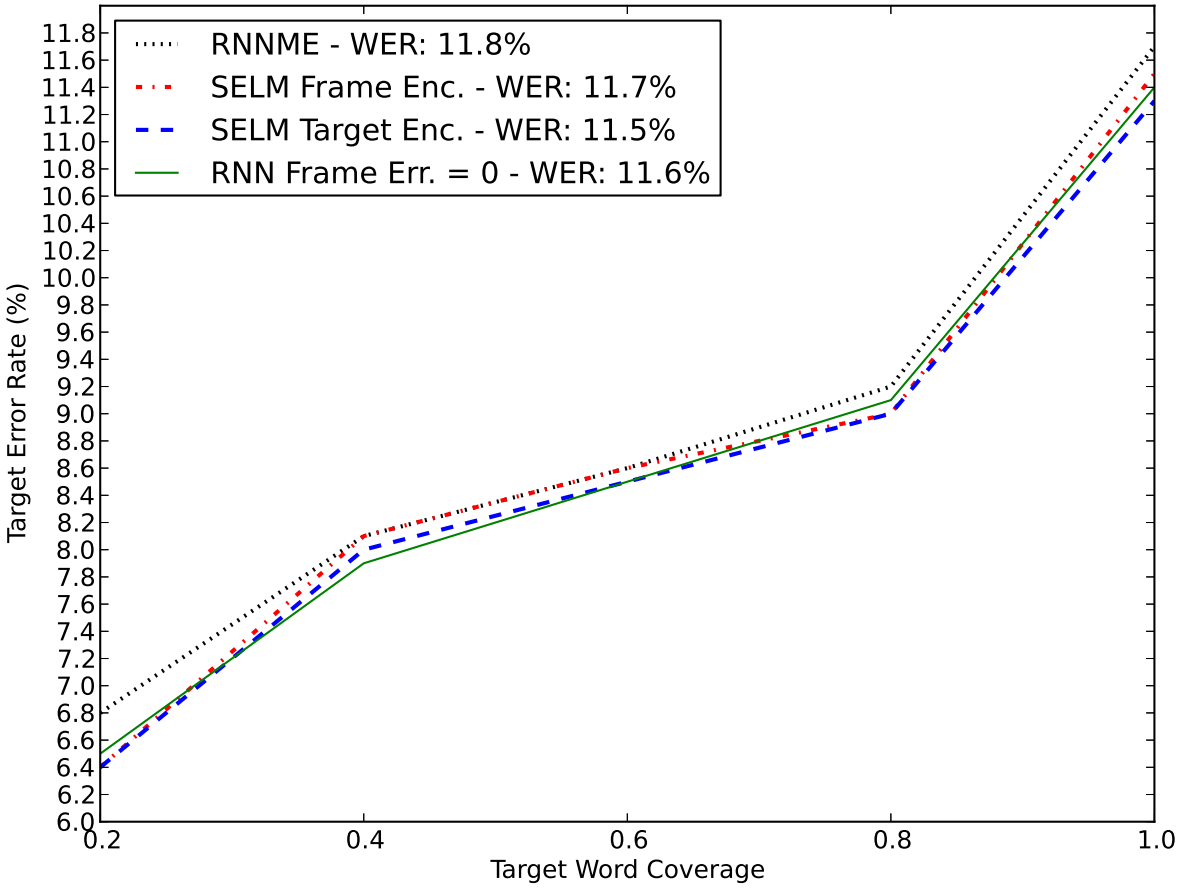

Figure 9.11: Target error rate at different coverages of target words for ASR encodings

SELMs with equal weights.

We also present the performance of two additional RNNMEs which are initialized with different random weights, which are referred to as "RNNME (2)" and "RNNME (3)". We linearly interpolate our original RNNME model ("RNNME (1)") and the new RNNMEs with equal weights. The purpose of presenting these results is to show that our improvements are not result of a *lucky* random initialization.

The WER and TER performance of these models are given in Table 9.3. "(1) + (2) + (3)" refers to the linear interpolation of all RNNME models. "(5) + (6)" refers to the linear interpolation of SELMs that use semantic encodings and "(5) + (6) + (7)" refers to the linear interpolating of all

SELMs.

Table 9.3: WER and TER performance of all of linearly interpolated SELMs. The interpolation is performed with equal weights. RNNME (1) is initialized with the same random weights as all the SELMs.

Model	**Dev93**		**Test92**		**Test93**	
	WER(%)	**TER(%)**	**WER(%)**	**TER(%)**	**WER(%)**	**TER(%)**
KN5	14.6	13.4	9.7	10.4	13.3	13.2
RNNME (1)	13.4	12.7	8.8	9.6	12.7	12.6
RNNME (2)	13.3	12.5	8.6	9.0	12.7	12.6
RNNME (3)	13.3	12.5	8.7	9.3	12.7	12.8
(1) + (2) + (3) (Lin. Int.)	**13.2**	**12.1**	**8.5**	**9.1**	**12.7**	**12.7**
SELM - Frame Enc. (5)	13.2	12.2	8.6	9.3	13.0	13.2
SELM - Target Enc. (6)	13.1	12.3	8.5	9.3	12.4	12.4
SELM - Err. Prune (7)	13.3	12.4	8.7	9.3	12.7	12.6
(5) + (6) (Lin. Int.)	**13.1**	**12.1**	**8.4**	**8.9**	**12.2**	**12.3**
(5) + (6) + (7) (Lin. Int.)	**13.1**	**12.2**	**8.5**	**9.4**	**12.1**	**11.8**

We observe that the interpolation of the two SELMs with semantic encodings gives the best performance on both WER and TER. They achieve a relative improvement of 4.5% in WER and 7.3% in TER for the "Test92" set and 3.9% in WER and 2.4% in TER for the "Test93" set over the corresponding RNNME (the RNNME that is initialized with the same random weights), "RNNME (1)". They still have a slightly better performance if all of the RNNMEs are combined together. Combination of the SELMs that uses semantic encodings with the SELM that uses error pruning brings instability, it improves performance on "Test93", however, reduces the performance on "Test92". The comparison of SELMs with different RNNMEs shows that SELMs outperform RNNMEs in general. The SELMs with semantic encodings are more stable when combined together with linear interpolation.

9.4 Discussion

In this chapter, we have introduced the use of deep semantic encodings as the semantic context for SELMs. Deep semantic encodings overcome the problem of the ASR noise in the semantic context of utterances. The deep semantic encodings are built either from the BoW vectors of frames or targets. These encodings are high-level binary representations of semantic information. The binary representation is more robust to the ASR noise than a continuous representation. We observe that target encodings are more robust to ASR noise than frame encodings. In addition, as the size of the encodings increases, the noise becomes more effective on encodings.

The SELMs that use deep semantic encodings as the semantic context perform better than RNNMEs that do not consider any semantic information. The re-scoring experiments on WSJ corpus show that SELMs with semantic encodings outperform RNNMEs in general. The detailed analysis on the understanding performance shows that even the ASR noise affects the accuracy of semantic encodings, the SELMs have a slightly better understanding performance.

The SELMs that use different kind of semantic information can be combined by linear interpolation. We observe that linearly interpolated SELMs that uses frame encodings and target encodings perform the best even compared to various RNNMEs interpolated together. Also, SELMs that use error pruning on frames are combined with the other SELMs. We observe an instability in this combination. Therefore, we prefer to combine only the SELMs that use deep semantic encodings.

In conclusion, deep semantic encodings are noisy semantic representations that can be used to suppress the ASR noise on the semantic context. The SELMs that use these deep semantic encodings outperform RNNME in general. The SELMs show better understanding and recognition perfor-

mance compared to their RNNME counterpart. Therefore, SELMs with deep semantic encodings are appropriate for SLS that needs joint optimization of recognition and understanding. SELMs can be made more robust by linearly interpolating SELMs that use different semantic encodings.

Chapter 10

Conclusion and Future Work

Language models (LMs) are one of the most important components of spoken language systems (SLS). They constrain the search space that the SLS use for finding the best hypothesis. SLS, most of the time, use a cascaded approach, in which the hypothesis of the ASR component is fed into the SLU component and the ASR hypothesis is optimized to have a better recognition performance. However, as it has been discussed, a better recognition performance does not always yield a better understanding performance. Therefore, for the ultimate goal of better understanding, an optimization must be performed by considering the lexical-semantic relation space. In this thesis, we introduce two different LMs that consider lexical and semantic constraints jointly when modeling the language: joint LMs and semantic LMs.

Traditional LMs, cannot handle long-range dependencies because of the limited histories they consider. Span extension can be done either by using the syntactic constraints or the semantic constraints. We investigate semantic span extension by using semantic LMs (SELMs). SELMs use semantic features that are based on a well-established theory of lexical semantics, which is called the theory of frame semantics. By incorporating semantic context in the LM, SELMs are able to handle long-range semantic

dependencies.

Joint LMs we have presented are built by using word-concept pairs jointly. We have shown how these models can be optimized either for recognition or for understanding by changing the amount of semantic information we place in the LM. In addition, we have shown how these models can be adapted to the dialog context by means of on-line instance based adaptation. On-line instance based adaptation creates the adaptation data by considering what the ASR recognizes. Therefore, the adaptation of joint models in SLS can be described as "the system understands better what it recognizes". We have achieved significant improvements in concept error rate (CER) by using instance-based adaptation. We have also shown the performance of joint LMs in a cross-language SLU porting task. We have shown that joint LMs can replace domain adaptation and also they improve the SLU 1st-best hypothesis significantly.

SELMs address the problem of long-range dependencies by incorporating the semantic context in the LM by using semantic features extracted from the ASR hypothesis. These semantic features are the frames and the targets that are defined in the theory of frame semantics. The noisy ASR hypothesis affects the accuracy of the semantic information and reduces the potential high performance of these models. To raise the accuracy, we propose to do error pruning on the erroneous frames. The re-scoring experiments on two different corpora show that, error pruning improves the performance of SELMs. However, when the WER of the ASR is very high (as in the LUNA HH corpus), SELMs cannot perform as well as the state-of-the-art LMs (RNNMEs). On the contrary, when ASR noise is low (as in WSJ corpus), SELMs slightly outperform RNNMEs.

We also address the problem of ASR noise on semantic features by using deep autoencoders. We propose to use deep autoencoders that encodes semantic features into binary vectors by introducing noise in the encoding

process. Binary vectors, which are also called semantic encodings, lower the discrepancy between the semantic encodings of the ASR hypotheses and the semantic encodings of the reference transcriptions. We have shown that SELMs that use semantic encodings outperform RNNMEs in the recognition performance (WER). The SELMs, in this manner, can be described as "the system recognizes better what it understands". In addition, we have assessed the understanding performance of SELMs by measuring the WER only on target words i.e. target error rate (TER). Target words are the meaning bearing elements of semantic frames and they constitute a good proxy for assessing the understanding performance. We have shown that SELMs can be jointly optimized for a better recognition and understanding performance and they outperform state-of-the-art RNNMEs in this joint performance.

We believe that apart from the semantic information, LMs for spoken dialog systems may benefit from the dialog context. As a future work we plan to integrate the dialog context altogether with the semantic information. The most important problem regarding SELMs is the problem of noisy semantic features. We plan to apply SELMs to other tasks like statistical machine translation, where there is no noise on the semantic information extracted from the source language (not considering the noise the frame-semantic parser introduces). We believe that, the improvements SELMs achieve in these situations would be relatively higher.

Bibliography

[1] A. Aamodt and E. Plaza. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] Lalit R. Bahl and F. Jelinek. Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition. *Information Theory, IEEE Transactions on*, 21(4):404–411, Jul 1975.

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[4] James K. Baker. The dragon system-an overview. *IEEE Transactions on Acoustics Speech and Signal Processing*, 23(1):24–29, February 1975.

[5] A.O. Bayer and G. Riccardi. Joint language models for automatic speech recognition and understanding. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 199–203. IEEE, Dec 2012.

[6] A.O. Bayer and G. Riccardi. On-line adaptation of semantic models for spoken language understanding. In *Automatic Speech Recognition*

and Understanding (ASRU), 2013 IEEE Workshop on, pages 90–95. IEEE, Dec 2013.

[7] A.O. Bayer and G. Riccardi. Semantic language models for automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 7–12. IEEE, Dec 2014.

[8] A.O. Bayer and G. Riccardi. Deep semantic encodings for language modeling. In *Interspeech 2015, 16th Annual Conference of the International Speech Communication Association*, 2015.

[9] Jerome R. Bellegarda. Robustness in statistical language modeling: Review and perspectives. In Jean-Claude Junqua and Gertjan van Noord, editors, *Robustness in Language and Speech Technology*. Springer, 2001.

[10] J.R. Bellegarda. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296, Aug 2000.

[11] J.R. Bellegarda. Large vocabulary speech recognition with multispan statistical language models. *Speech and Audio Processing, IEEE Transactions on*, 8(1):76–84, Jan 2000.

[12] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, Aug 2013.

[13] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, Mar 1994.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March 2003.

[15] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March 1996.

[16] M. W. Berry, S.T. Dumais, G.W. O'Brien, Michael W. Berry, Susan T. Dumais, and Gavin. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.

[17] Nicola Bertoldi and Marcello Federico. Domain adaptation for statistical machine translation from monolingual resources. In *Proceedings of WMT*, 2009.

[18] M. Bisani and H. Ney. Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proceedings of ICASSP*, 2004.

[19] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.

[20] Mikael Bodén. A guide to recurrent neural networks and backpropagation. In *In the Dallas project, SICS Technical Report T2002:03, SICS*, 2002.

[21] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[22] Harry Bunt. A framework for dialogue act specification. *Proceedings of SIGSEM WG on Representation of Multimodal Semantic Information*, 2005.

[23] Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 152–164, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[24] Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. Structure and performance of a dependency language model. In *In Proceedings of Eurospeech*, pages 2775–2778, 1997.

[25] Ciprian Chelba and Frederick Jelinek. Structured language modeling. *Computer Speech & Language*, 14(4):283–332, 2000.

[26] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, October 1999.

[27] Y.-L. Chow and S. Roukis. Speech understanding using a unification grammar. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 727–730 vol.2, May 1989.

[28] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learnng Research*, 12:2493–2537, November 2011.

[29] Bonaventura Coppola, Alessandro Moschitti, and Giuseppe Riccardi. Shallow semantic parsing for spoken language understanding. In *Proceedings of Human Language Technologies: The 2009 Annual Confer-*

ence of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, NAACL-Short '09, pages 85–88, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[30] Mark G. Core and James F. Allen. Coding dialogs with the DAMSL annotation scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Cambridge, MA, November 1997.

[31] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. In *The Annals of Mathematical Statistics*, volume 43, pages 1470–1480, 1972.

[32] D. Das, D. Chen, A. F. T. Martins, N. Schneider, and N. Smith. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56, 2014.

[33] K. H. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642, November 1952.

[34] R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur. Spoken language understanding. *Signal Processing Magazine, IEEE*, 25(3):50–58, May 2008.

[35] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[36] Li Deng, G. Tur, Xiaodong He, and D. Hakkani-Tur. Use of kernel deep convex networks and end-to-end learning for spoken language

understanding. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 210–215, Dec 2012.

[37] Li Deng and Dong Yu. Deep convex network: A scalable architecture for speech pattern classification. In *Proceedings of Interspeech 2011*. ISCA, August 2011.

[38] A. Deoras, G. Tur, R. Sarikaya, and D. Hakkani-Tur. Joint discriminative decoding of words and semantic tags for spoken language understanding. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(8):1612–1621, Aug 2013.

[39] Anoop Deoras and Ruhi Sarikaya. Deep belief network based semantic taggers for spoken language understanding. In *Proceedings of Interspeech*. ISCA, September 2013.

[40] Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. Reranking models based-on small training data for spoken language understanding. In *EMNLP*, pages 1076–1085. ACL, 2009.

[41] 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.

[42] M. Eck, S. Vogel, and A. Waibel. Language model adaptation for statistical machine translation based on information retrieval. In *Proceedings of LREC 2004*, 2004.

[43] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[44] A. Emami and F. Jelinek. Exact training of a neural syntactic language model. In *Acoustics, Speech, and Signal Processing, 2004.*

Proceedings. (ICASSP '04). IEEE International Conference on, volume 1, pages I–245–8 vol.1, May 2004.

[45] A. Emami, Peng Xu, and F. Jelinek. Using a connectionist model in a syntactical based language model. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 1, pages I–372–I–375 vol.1, April 2003.

[46] Charles J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, 280(1):20–32, 1976.

[47] Charles J. Fillmore and Collin Baker. A frames approach to semantic analysis. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*. Oxford University Press, 2009.

[48] Charles J. Fillmore, Christopher R. Johnson, and Miriam R. L. Petruck. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250, September 2003.

[49] Daniel Gildea and Thomas Hofmann. Topic-based language models using EM. In *Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH-99)*, pages 2167–2170, Budapest, 1999.

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

[51] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

[52] Joshua T. Goodman. A bit of progress in language modeling: Extended version. Technical Report MSR-TR-2001-72, Microsoft Research, 2001.

[53] S. Hahn, M. Dinarelli, C. Raymond, F. Lefevre, P. Lehnen, Renato de Mori, A. Moschitti, H. Ney, and G. Riccardi. Comparing stochastic approaches to spoken language understanding in multiple languages. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(6):1569–1583, Aug 2011.

[54] Dilek Z. Hakkani-Tr, Frdric Bchet, Giuseppe Riccardi, and Gkhan Tr. Beyond asr 1-best: Using word confusion networks in spoken language understanding. *Computer Speech and Language*, 20(4):495–514, 2006.

[55] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

[56] Peter A. Heeman. POS tags and decision trees for language modeling. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 129–137, College Park, MD, 1999. ACL.

[57] G. Hinton, Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, Nov 2012.

[58] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[59] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. Technical Report UTML TR 2010003, Department of Computer Science, University of Toronto, 2010.

[60] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

[61] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[62] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[63] B. Jabaian, L. Besacier, and F. Lefevre. Combination of stochastic understanding and machine translation systems for language portability of dialogue systems. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5612–5615, May 2011.

[64] B. Jabaian, L. Besacier, and F. Lefevre. Comparison and combination of lightly supervised approaches for language portability of a spoken language understanding system. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(3):636–648, March 2013.

[65] Bassam Jabaian, Laurent Besacier, and Fabrice Lefèvre. Investigating multiple approaches for SLU portability to a new language. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *Proceedings of Interspeech 2010*, pages 2502–2505. ISCA, 2010.

[66] F. Jelinek. Self-organized language modeling for speech recognition. In A. Waibel and K. F. Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan-Kaufmann, 1990.

[67] Frederick Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, April 1976.

[68] Frederick Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA, 1997.

[69] Minwoo Jeong and Gary Geunbae Lee. Practical use of non-local features for statistical spoken language understanding. *Computer Speech and Language*, 22(2):148–170, April 2008.

[70] M. I. Jordan. Serial order: A parallel distributed processing approach. Technical Report ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.

[71] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchaman, and N. Morgan. Using a stochastic context-free grammar as a language model for speech recognition. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 189–192 vol.1, May 1995.

[72] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 400–401, 1987.

[73] Dietrich Klakow. Log-linear interpolation of language models. In *The 5th International Conference on Spoken Language Processing, Incorporating The 7th Australian International Speech Science and Tech-*

nology Conference, Sydney Convention Centre, Sydney, Australia, 30th November - 4th December 1998. ISCA, 1998.

[74] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184 vol.1, May 1995.

[75] Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modelling. In *Third European Conference on Speech Communication and Technology, EUROSPEECH 1993, Berlin, Germany, September 22-25, 1993*. ISCA, 1993.

[76] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT.

[77] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computer Linguistics, 2007.

[78] Philipp Koehn and Josh Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[79] S. Kombrink, T. Mikolov, M. Karafiat, and L. Burget. Recurrent neural network based language modeling in meeting recognition. In *Proceedings of Interspeech 2011*, pages 2877–2880. ISCA, 2011.

[80] Stefan Kombrink, Tomas Mikolov, Martin Karafiát, and Lukás Burget. Recurrent neural network based language modeling in meeting recognition. In *Interspeech 2011, 12th Annual Conference of the International Speech Communication Association*, pages 2877–2880, 2011.

[81] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

[82] R. Lau, R. Rosenfeld, and S. Roukos. Trigger-based language models: a maximum entropy approach. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 45–48 vol.2, April 1993.

[83] Hai-Son Le, I. Oparin, A. Allauzen, J. Gauvain, and F. Yvon. Structured output layer neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5524–5527, May 2011.

[84] Hai Son Le, Ilya Oparin, Abdelkhalek Messaoudi, Alexandre Allauzen, Jean-Luc Gauvain, and Franois Yvon. Large vocabulary soul neural network language models. In *INTERSPEECH*, pages 1469–1472. ISCA, 2011.

[85] G. Leech and A. Wilson. EAGLES recommendations for the morphosyntactic annotation of corpora. Technical report, ILC-CNR, 2006.

[86] Fabrice Lefèvre, Franois Mairesse, and Steve Young. Cross-lingual spoken language understanding from unaligned data using discriminative classification models and machine translation. In *Proceedings of Interspeech 2010*, pages 78–81. ISCA, 2010.

[87] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, pages 171–185, 1995.

[88] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[89] Michael F. McTear. *Spoken dialogue technology: toward the conversational user interface.* Springer, 2004.

[90] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of Interspeech 2013*. ISCA, August 2013.

[91] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký. Strategies for training large scale neural network language models. In *Proceedings of ASRU*, pages 196–201. IEEE, 2011.

[92] T. Mikolov and G. Zweig. Context dependent recurrent neural network language model. In *Proceedings of SLT*, pages 234–239. IEEE, 2012.

[93] Tomas Mikolov. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[94] Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukás Burget, and Jan Černocký. Empirical evaluation and combination of advanced language modeling techniques. In *Interspeech 2011, 12th Annual Conference of the International Speech Communication Association*, pages 605–608, 2011.

[95] Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753, 2014.

[96] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, 2010.

[97] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Černocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of ICASSP*, pages 5528–5531. IEEE, 2011.

[98] Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. Hidden understanding models of natural language. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 25–32, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[99] Marvin Minsky. A framework for representing knowledge. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[100] Andriy Mnih and Geoffrey E. Hinton. A scalable hierarchical distributed language model. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088. Curran Associates, Inc., 2008.

[101] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Citeseer, 2005.

[102] T.R. Niesler, E.W.D. Whittaker, and P.C. Woodland. Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 177–180 vol.1, May 1998.

[103] Douglas B. Paul and Janet M. Baker. The Design for the Wall Street Journal-based CSR Corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 357–362, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[104] Roberto Pieraccini and Esther Levin. A learning approach to natural language understanding. In *In Speech Recognition and Coding, New Advances and Trends, NATO ASI Series*, pages 139–155. Springer Verlag, 1995.

[105] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky,

G. Stemmer, and K. Vesely. The Kaldi speech recognition toolkit. In *Proceedings of ASRU*. IEEE, 2011.

[106] P. J. Price. Evaluation of spoken language systems: The ATIS domain. In *Proceedings of the Speech and Natural Language Workshop*, pages 91–95, Hidden Valley, PA, 1990.

[107] Christian Raymond and Giuseppe Riccardi. Generative and discriminative algorithms for spoken language understanding. In *Interspeech 2007, 8th Annual Conference of the International Speech Communication Association*, pages 1605–1608, 2007.

[108] G. Riccardi and A. L. Gorin. Stochastic language models for speech recognition and understanding. In *ICSLP, Sydney, Nov. 1998*, 1998.

[109] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, Aug 2000.

[110] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228, 1996.

[111] Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer Speech and Language*, 15(1):55–73, 2001.

[112] D.E. Rumelhart, G.E. Hintont, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[113] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July 2009.

[114] Richard M. Schwartz, Toru Imai, Francis Kubala, Long Nguyen, and John Makhoul. A maximum likelihood model for topic classification of broadcast news. In *Proceedings of EUROSPEECH*. ISCA, 1997.

[115] Holger Schwenk. Efficient training of large neural networks for language modeling. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 3059–3064 vol.4, July 2004.

[116] Holger Schwenk. Continuous space language models. *Computer Speech and Language*, 21(3):492–518, July 2007.

[117] C. Servan, N. Camelin, C. Raymond, F. Bechet, and Renato de Mori. On the use of machine translation for spoken language understanding portability. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5330–5333, March 2010.

[118] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In David E. Rumelhart, James L. McClelland, and PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 194–281. MIT Press, Cambridge, MA, USA, 1986.

[119] E.A. Stepanov, I. Kashkarev, A.O. Bayer, G. Riccardi, and A. Ghosh. Language style and domain adaptation for cross-language slu porting. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 144–149. IEEE, Dec 2013.

[120] Evgeny Stepanov, Giuseppe Riccardi, and Ali Orkan Bayer. The development of the multilingual LUNA corpus for spoken language system porting. In *Proceedings of the Ninth International Confer-*

ence on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, may 2014.

[121] David Traum and Universite De Geneve. Conversational agency: The Trains-93 dialogue manager. In *Susann LuperFoy, Anton Nijholt, and Gert Veldhuijzen van Zanten, editors, Proceedings of Twente Workshop on Language Technology, TWLT-11*, pages 1–11, 1996.

[122] G. Tur, Li Deng, D. Hakkani-Tur, and Xiaodong He. Towards deeper understanding: Deep convex networks for semantic utterance classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5045–5048, March 2012.

[123] Gokhan Tur, Anoop Deoras, and Dilek Hakkani-Tur. Semantic parsing using word confusion networks with conditional random fields. In *Proceedings of Interspeech*. ISCA, September 2013.

[124] A. M. Turing. Computing machinery and intelligence, 1950.

[125] Jos J. A. van Berkum, Colin M. Brown, Pienie Zwitserlood, Valesca Kooijman, and Peter Hagoort. Anticipating Upcoming Words in Discourse: Evidence From ERPs and Reading Times. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 31:443–467, 2005.

[126] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, December 2010.

[127] Ye-Yi Wang, A. Acero, and C. Chelba. Is word error rate a good indicator for spoken language understanding accuracy. In *Automatic*

Speech Recognition and Understanding, 2003. ASRU '03. 2003 IEEE Workshop on, pages 577–582, Nov 2003.

[128] Ye-Yi Wang and Alex Acero. Discriminative models for spoken language understanding. In *the International Conference on Spoken Language Processing*, pages 1766–1769. International Speech Communication Association, 2006.

[129] Ye-Yi Wang, Li Deng, and A. Acero. Spoken language understanding. *Signal Processing Magazine, IEEE*, 22(5):16–31, Sept 2005.

[130] Ye-Yi Wang, Li Deng, and Alex Acero. Semantic frame-based spoken language understanding. In Gokhan Tur and Renato De Mori, editors, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Wiley, 2011.

[131] Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *Proceedings of ASRU*, pages 78–83. IEEE, 2013.

[132] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. Spoken language understanding using long short-term memory neural networks. In *Proceedings of SLT*. IEEE, December 2014.

[133] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. Recurrent neural networks for language understanding. In *Proceedings of Interspeech 2013*. ISCA, August 2013.

[134] Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[135] S. Young. A review of large-vocabulary continuous-speech. *Signal Processing Magazine, IEEE*, 13(5):45–57, Sept 1996.

[136] F. Zamora-Martinez, S. Espana-Boquera, J. Castro-Bleda, M., and R. De-Mori. Cache neural network language models based on long-distance dependencies for a spoken dialog system. In *Proceedings of ICASSP*. IEEE, 2012.

[137] V. Zue, J. Glass, D. Goodine, H. Leung, M. Phillips, J. Polifroni, and S. Seneff. Integration of speech recognition and natural language processing in the MIT Voyager system. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 713–716 vol.1, Apr 1991.