Available online at www.sciencedirect.com

ScienceDirect

Computer Speech and Language 40 (2016) 1–22

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

Semantic language models with deep neural networks

Ali Orkan Bayer *, Giuseppe Riccardi

Signals and Interactive Systems Lab, Department of Information Engineering and Computer Science, University of Trento, Italy

Received 22 October 2015; received in revised form 29 February 2016; accepted 17 April 2016
Available online 4 May 2016

Abstract

In this paper we explore the use of semantics in training language models for automatic speech recognition and spoken language understanding. Traditional language models (LMs) do not consider the semantic constraints and train models based on fixed-sized word histories. The theory of frame semantics analyzes word meanings and their constructs by using "semantic frames". Semantic frames represent a linguistic scene with its relevant participants and their relations. They are triggered by target words and include slots which are filled by frame elements. We present semantic LMs (SELMs), which use recurrent neural network architectures and the linguistic scene of frame semantics as context. SELMs incorporate semantic features which are extracted from semantic frames and target words. In this way, long-range and "latent" dependencies, i.e. the implicit semantic dependencies between words, are incorporated into LMs. This is crucial especially when the main aim of spoken language systems is understanding what the user means. Semantic features consist of low-level features, where frame and target information is directly used; and deep semantic encodings, where deep autoencoders are used to extract semantic features. We evaluate the performance of SELMs on publicly available corpora: the Wall Street Journal read-speech corpus and the LUNA human–human conversational corpus. The encoding of semantic frames into SELMs improves the word recognition performance and especially the recognition performance of the target words, the meaning bearing elements of semantic frames. We assess the performance of SELMs for the understanding tasks and we show that SELMs yield better semantic frame identification performance compared to recurrent neural network LMs.

Keywords: Language modeling; Recurrent neural networks; Frame semantics; Semantic language models; Deep autoencoders

1. Introduction

Statistical language models (LMs) are one of the main knowledge sources in language processing systems such as statistical machine translation, information retrieval, automatic speech recognition (ASR), and spoken language understanding (SLU). They play a crucial role in searching for the best hypothesis by estimating the likelihood of each hypothesis in that language.

Traditional LMs are based on n-gram models, where the probability of a word is only dependent on the previous $(n-1)$ words. Although currently the state-of-the-art performance for LMs is obtained by using recurrent neural networks (RNNs), n-gram LMs are still widely used because of their simple computational architecture and also because they scale well (Chelba et al., 2012, 2013) on large data. However, since n-gram LMs are trained by considering fixed

* Corresponding author at: Signals and Interactive Systems Lab, Via Sommarive, 5 38123 Povo, Trento, Italy. Tel.: +39 3209714240; fax: +39 0461283166.
E-mail address: aliorkan.bayer@unitn.it (A.O. Bayer), giuseppe.riccardi@unitn.it (G. Riccardi).

http://dx.doi.org/10.1016/j.csl.2016.04.001
0885-2308/

size histories of words, they suffer from the "locality problem" (Bellegarda, 2000a). As suggested by Bellegarda (2000a), this problem can be solved by "span extension". Span extension can be performed either by using syntactic dependencies or semantic relations. This paper explores the use of lexical semantics for improving the performance of LMs.

One of the approaches that incorporates semantic information into LMs is the *topic model* (Gildea and Hofmann, 1999; Schwartz et al., 1997). 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.

Another approach for semantic span extension is to use trigger pairs. Trigger pairs capture semantic relations by using the correlation between words (Rosenfeld, 1996). 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 (1996) 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. However, selection of trigger pairs is an issue.

Bellegarda (2000a, 2000b) uses *latent semantic analysis* (LSA) to extend the trigger pairs approach. LSA (Berry et al., 1995; Deerwester et al., 1990) is used as an indexing mechanism in information retrieval, and it maps the discrete space of words and documents to the same continuous space. Therefore, each word and each 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 the 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 (Bellegarda, 2000a, 2000b).

Traditional LMs also suffer from "curse of dimensionality" (Bengio et al., 2003), i.e. they consider words as sequence of symbols and do not model the semantic relationships between these words. This problem is approached by learning distributed representations (also known as *word embeddings*) of words (Bengio et al., 2003), i.e. words are mapped onto a continuous space. Neural network LMs (NNLMs) are first introduced in Bengio et al. (2003). NNLMs learn and use distributed representations of words in language modeling. NNLMs are reported to reduce the perplexity significantly (Bengio et al., 2003). Also Schwenk (2007) applies NNLMs to ASR which reports significant improvements in word error rate (WER) by linearly interpolating NNLMs with back-off n-gram LMs. The first approach to NNLMs were feed-forward NNLMs which are based on fixed size histories; therefore they also suffer from the problems related to fixed size histories. Recurrent NNLMs (RNNLMs) (Mikolov, 2012; Mikolov et al., 2010), overcome this problem by using recurrent connections, which feed the activation of the hidden layer at the previous time step as input. This can be thought of as a short-term memory which enables the network to model long-range histories. RNNLMs are shown to improve perplexities and WERs better than any feed-forward NNLM (Mikolov et al., 2011a). Mikolov and Zweig (2012) add a feature layer to RNNLMs, where topic features are used as additional context to the NNLM. In addition, Mikolov et al. (2011b) present the training of maximum entropy features jointly with the RNNLM which are referred to as RNNME.

When building LMs for a specific application, LMs are tuned with respect to the performance metric of the target application. This may lead to problems especially for spoken dialog systems, where one of the main goals of these systems is to extract user intentions and the meaning of utterances. Spoken dialog systems most often use a cascaded approach, where the output of the ASR is fed into the SLU module. The LMs that are used in ASR and SLU are optimized with respect to the component that they are trained for. LMs for ASR are optimized to lower the WER. LMs are optimized to lower the concept error rate (CER) in SLU. In the literature, it has been argued that LMs should be optimized jointly, since the best recognition performance does not yield the best understanding performance (Deoras et al., 2013; Riccardi and Gorin, 1998; Wang et al., 2003). Therefore, it is important that LMs are trained by considering semantic constraints in that language.

As we have mentioned traditional n-gram LMs suffer from the "locality problem" and "curse of dimensionality". The semantic span extension approaches like the topic model (Gildea and Hofmann, 1999; Schwartz et al., 1997) can only be used through a combination of the model with an n-gram LMs. Trigger pairs (Rosenfeld, 1996) offer an

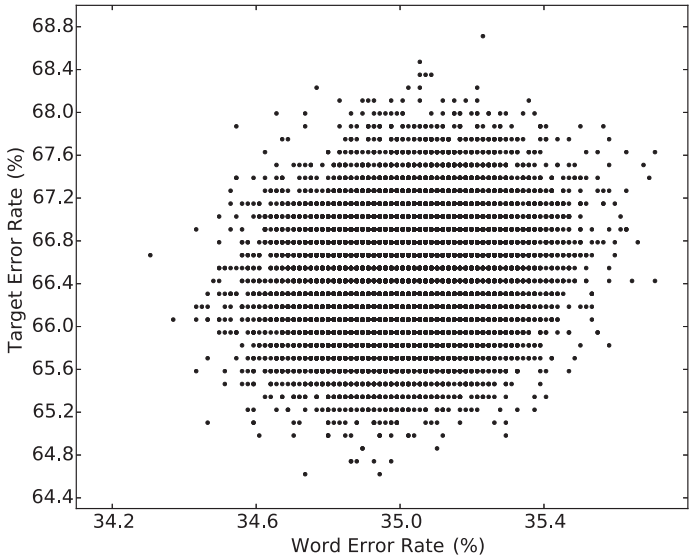

Fig. 1. The scatter plot of WER versus TER for random selections of hypotheses from the 100-best list of the test set of LUNA HH corpus.

effective way to handle long-range dependencies; however, selection of trigger pairs is the main issue. LMs built by using LSA use distributed representations for each document and word; however, these representations are extracted statistically and are not able to incorporate any real world knowledge into LMs. Neural network LMs deal with "curse of dimensionality"; however, they do not use any explicit semantic constraints on LMs. This paper presents semantic LMs (SELMs) that are built over lexical semantics resources; therefore, they incorporate semantics and world knowledge that is present in these resources into LMs. In addition, SELMs use the structure of RNNLMs to handle "curse of dimensionality".

Considering the discussions related to the evaluation of LMs (Deoras et al., 2013; Riccardi and Gorin, 1998; Wang et al., 2003), in this paper we evaluate LMs both on WER and target error rate (TER) for the recognition performance, and on frame identification accuracy for the understanding performance. TER measures the errors that are made on the main meaning bearing elements of semantic frames; therefore TER is a good proxy for the understanding performance. The computation of TER is similar to WER except that it is computed only on the target words. Fig. 1 shows the scatter plot of TER and WER of random selection of hypotheses from the n-best hypotheses on LUNA HH corpus. It is possible to select the hypotheses in various ways during re-scoring. It can be seen that some selections that would lower WER, could make TER higher. In this way the system improves its transcription performance; however, it fails to recognize the meaning bearing words that are important in defining the meaning of the utterance. We argue that LMs need to be optimized also for recognizing the meaning bearing elements, hence for understanding, by considering semantic constraints and the evaluation of LMs needs to include both metrics and the understanding performance.

In this paper, we introduce semantic LMs (SELMs) that are trained with lexical and semantic constraints. Lexical constraints are imposed by using the words as the units of the LM, and semantic constraints are imposed by the semantic context of utterances that is extracted by using the target words and semantic frames. The structure of the paper is as follows. Section 2 describes the structure of SELMs. Section 3 presents the features used in SELMs. Section 4 defines the experimental setting. The experimental work is described in Section 5 for the Wall Street Journal (WSJ) speech recognition corpus, and in Section 6 for the Italian LUNA human–human (LUNA HH) corpus. Finally, Section 7 concludes the paper.

2. Semantic language models

SELMs were first introduced in Bayer and Riccardi (2014) for incorporating semantic information in LMs. The aim of exploiting semantic information in LMs is to model the implicit semantic dependencies between words, which

also involves long-range dependencies that cannot be handled by local structures. Also, incorporating semantic constraints would yield LMs that are optimized for the recognition of meaning bearing words, which would have a better understanding performance.

In this respect, we choose to use the theory of frame semantics, which is a theory of lexical semantics to construct the linguistic scene. The linguistic scene that is constructed over frame semantics constitutes the pragmatic information about the utterance. In the commercial event scenario example by Fillmore (1976); the *commercial event* frame contains roles or slots like the *buyer*, the *seller*, the *goods*, and the *money*. Therefore, the identification of a frame predicts the existence of its relevant roles which are closely related to the relevant words for that frame.

2.1. Frame semantics

Frame semantics is the theory of lexical meaning (Fillmore et al., 2003). In the theory of frame semantics word meanings are defined in the context of semantic *frames* which are evoked by *target words* or *targets* (Fillmore et al., 2003). Each frame has elements called *frame elements* that complete the meaning of that frame. FrameNet project (Fillmore et al., 2003) is a resource that describes the semantic frames and their relationship between each other. An example sentence taken from the FrameNet project is presented below:

[*Lee*] **sold** [*a textbook*] [*to Abby*]

This sentence is an example for the "Commerce Sell" frame. The target word "**sold**" evokes the "Commerce Sell" frame. The "Commerce Sell" frame has the frame elements *Seller*, *Goods*, and *Buyer* filled by *Lee*, *a textbook*, and *to Abby* respectively. Hence, frame semantics predicts the existence of related roles (*Seller*, *Goods*, and *Buyer*) for the semantic frame, which can be thought as the world knowledge about lexical items.

Fig. 2 presents the intuition behind SELMs with an example from Penn-Treebank (Marcus et al., 1993). In this example, the frames "Being at Risk", "Commerce Scenario" and "Commerce Sell" create a linguistic scene where the non-target word "market" is an *expected* relevant word to this scene. Semantic frames create a linguistic scene and raise the expectation of relevant words in this linguistic scene. The frames that create this linguistic scene can occur anywhere in the utterance, therefore the whole utterance should be considered.

FrameNet is a public lexical resource for employing world knowledge in natural language understanding systems. The project is on-going and the coverage is improving. The FrameNet version 1.5 contains 1019 frames with around 12,000 lexical entries. However, FrameNet is still limited because of the manual effort needed to annotate the frames. FrameNet has also been ported to other languages. Also, it is possible to augment the coverage of FrameNet by considering domain specific frames or considering the style of the linguistic text. Tonelli and Riccardi (2010) present the methodology used to extend the FrameNet frames to Italian and to a target application domain. In addition Tonelli and Riccardi (2010) extended the FrameNet resource frames to cover the case of oral communication.

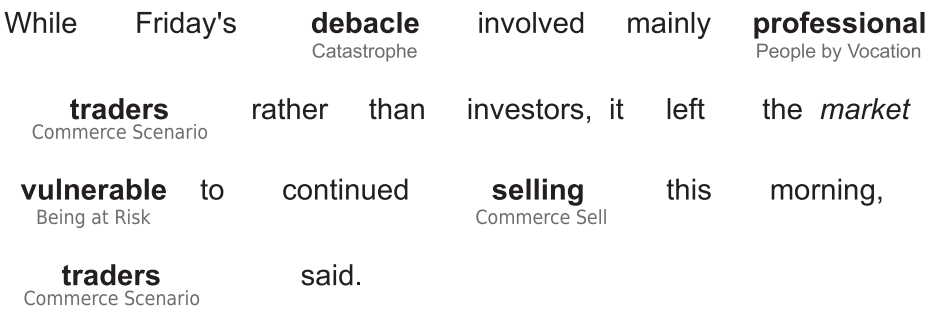

Fig. 2. An example sentence from Penn-Treebank. Some of the target words are shown in bold, the frames that are evoked are shown in gray. The linguistic scene constructed by the frames "Being at Risk", "Commerce Scenario" and "Commerce Sell" helps in predicting the relevant non-target word "market", which is shown in italics.

2.2. Model structure

SELMs are neural network LMs that are based on the RNNLM (Mikolov et al., 2010) architecture. This section first presents the RNNLMs and then describes the SELM architecture.

2.2.1. RNN language models

RNNLMs (Mikolov et al., 2010) use recurrent connections to propagate the state of the network through time to handle long-range dependencies. The major computational complexity of RNNLMs is the size of the vocabulary. This complexity can be reduced by using a class-based approach at the output layer. In this approach, the words in the vocabulary are clustered based on their unigram frequencies. The joint probability of a word and its class is factorized into the class membership probability and the class probability as given in Mikolov et al. (2011c).

It has been shown that RNNLMs can be trained jointly with maximum entropy features. Mikolov (2012) presents the training of n-gram maximum entropy features jointly with an RNNLM model using a hash based implementation. This approach uses hash functions that map all the n-grams for the history of the current word (from unigram to a given *n*-gram degree) to a binary vector of a given size.[1] The maximum entropy features on these vectors of n-gram histories are modeled by using direct connections. In the class-based approach, half of these connections are reserved for class probabilities and half of these connections are used for class membership probabilities and two different hash functions are used for classes and words. These models are referred to as RNNME models and shown to outperform RNNLMs (Mikolov, 2012).

Class-based RNNME models have the following parameters[2]:

- $\mathbf{W_w}$: The weights between the input layer and the hidden layer.
- $\mathbf{W_r}$: The weights between the previous hidden and the current hidden layer (recursive connections).
- $\mathbf{W_{hc}}$: The weights between the hidden layer and the class layer.
- $\mathbf{W_{hw}}$: The weights between the hidden layer and the word layer.
- $\mathbf{W_{dc}}$: The direct connections between the hashed n-grams and the class layer (direct connections to classes).
- $\mathbf{W_{dw}}$: The direct connections between the hashed n-grams and the word layer (direct connections to words).
- $\mathbf{b_h}$: Biases of the hidden layer.
- $\mathbf{b_c}$: Biases of the class layer.
- $\mathbf{b_w}$: Biases of the word layer.

In addition class-based RNNMEs have the following inputs:

- $\mathbf{w_t}$: 1-of-n encoding of the word at time *t* (the current word).
- $\mathbf{s_{t-1}}$: The previous state of the network.
- $\mathbf{hc_t}$: The output of the class hash function for n-gram histories at time *t*.
- $\mathbf{hw_t}$: The output of the word hash function for n-gram histories at time *t*.

We define $\sigma(\mathbf{x})$ as the sigmoid function given by Equation 1, and $\alpha(\mathrm{x})$ as the softmax function. The *ith* component of the softmax function for a layer that has *N* units is defined by Equation 2.

$$\sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}} \tag{1}$$

$$\alpha(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \tag{2}$$

The recursive state of the class-based RNNME model at time *t* is defined by Equation 3.

[1] During this mapping some n-grams may overlap.

[2] We use bold uppercase symbols for matrices, and bold lowercase symbols for vectors. All the vectors defined are column vectors.

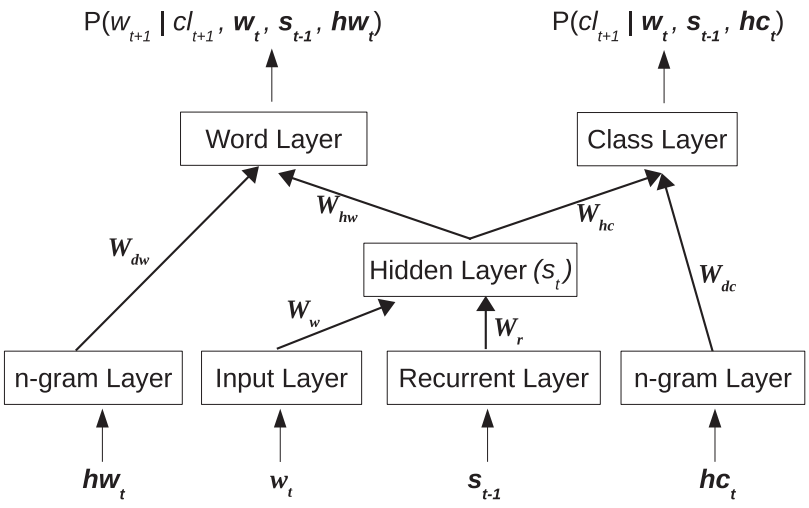

Fig. 3. The class-based RNNME architecture. The network takes the current word w_t, previous state $\mathbf{s_{t-1}}$, and the output of hash functions of n-gram histories $\mathbf{hw_t}$ and $\mathbf{hc_t}$ as input. The output layer estimates the joint probability of the next word w_{t+1} and its class cl_{t+1} by factorizing into class probabilities and class-membership probabilities. The maximum entropy features on n-grams are implemented as direct connections from the hash functions of n-gram histories to either the class layer or the word layer. The weights between each connection is shown in bold as $\mathbf{W_x}$.

$$\mathbf{s_t} = \mathbf{W_w w_t} + \mathbf{W_r s_{t-1}} + \mathbf{b_h} \tag{3}$$

The class probability of the next word is given by Equation 4.

$$P(cl_{t+1} | \mathbf{s_{t-1}}, \mathbf{w_t}, \mathbf{hc_t}) = \alpha(\mathbf{W_{hc}}\sigma(\mathbf{s_t}) + \mathbf{W_{dc}hc_t} + \mathbf{b_c}) \tag{4}$$

The class membership probabilities of words are given by Equation 5, where $\underset{cl_{t+1}}{\alpha}$ defines the softmax function only over the words that belong to the class cl_{t+1}.

$$P(w_{t+1} | cl_{t+1}, \mathbf{s_{t-1}}, \mathbf{w_t}, \mathbf{w_t}) = \underset{cl_{t+1}}{\alpha}(\mathbf{W_{hw}}\sigma(\mathbf{s_t}) + \mathbf{W_{dw}hw_t} + \mathbf{b_w}) \tag{5}$$

The joint probability of the words and their classes can be calculated by the multiplication of Equation 4 and Equation 5. The class-based RNNME architecture is depicted in Fig. 3.

2.2.2. SELM structure

The computational architecture of SELMs is based on the class-based RNNME structure, and it is shown in Fig. 4. In Fig. 4 the n-gram maximum entropy features are not shown for simplicity. SELMs have the following additional parameters with respect to class-based RNNMEs:

- $\mathbf{W_{sh}}$: The weights between the context layer and the hidden layer. These weights are only employed when low-level semantic encodings are used.
- $\mathbf{W_{sc}}$: The weights between the context layer and the class layer.
- $\mathbf{W_{sw}}$: The weights between the context layer and the word layer.

SELMs, compared to RNNMEs, use the semantic context of the utterance when estimating the word probabilities. The semantic context, **sc**, is computed for each utterance and fed into the SELM at the *context layer*. As discussed in the next section, semantic context can be a low-level representation that shows which frames or targets occur in that utterance; or a high-level encoding which is computed by an encoder network. In the case where the low-level representation is used, we connect the context layer to the hidden layer, the class layer, and the word layer. On the other hand, when the high-level encoding is used, we do not employ the connections between the context layer and the

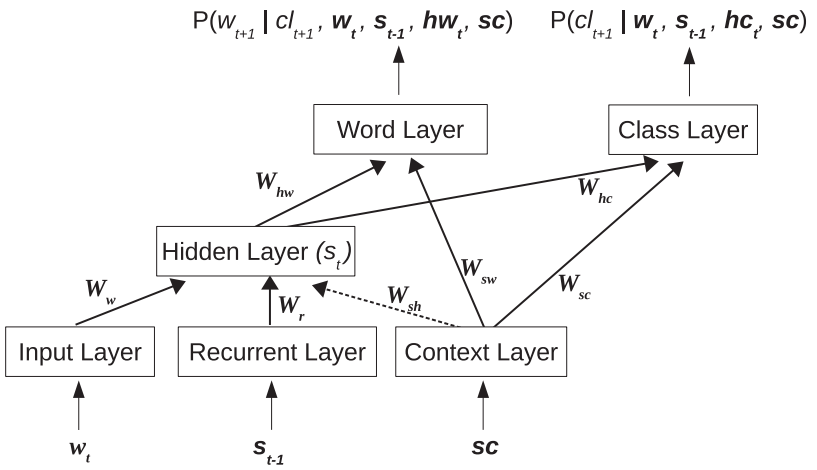

Fig. 4. The SELM structure, the direct connections for n-gram maximum entropy modeling are not shown. The network takes the current word w_t, the previous state $\mathbf{s_{t-1}}$, the output of hash functions for n-gram histories (not shown), and the semantic context **sc** for the current utterance as input. The output layer estimates the joint probability of the next word w_{t+1} and its class cl_{t+1}. When low-level semantic context is used the context layer is connected also to the hidden layer (dashed arrow). However, when high-level semantic context is used, there is no connection between the context layer and the hidden layer.

hidden layer since this encoding is computed by a deep encoder, and it is already a non-linear function of the frame or target information. Therefore, when the low-level representation is used the state of the SELM at time, s_t, is given by Equation 6a. However, when the high-level encodings are used the state s_t is computed the same way as RNNME which is repeated in Equation 6b.

$$\mathbf{s_t} = \mathbf{W_w w_t} + \mathbf{W_r s_{t-1}} + \mathbf{W_{sh} sc} + \mathbf{b_h} \tag{6a}$$

$$\mathbf{s_t} = \mathbf{W_w w_t} + \mathbf{W_r s_{t-1}} + \mathbf{b_h} \tag{6b}$$

The class probabilities and the class membership probabilities are given by Equation 7 and Equation 8 respectively, provided that the corresponding state equation is used for each semantic context type.

$$P(cl_{t+1} | s_{t-1}, \mathbf{w_t}, \mathbf{hc_t}, \mathbf{sc}) = \alpha(\mathbf{W_{hc}}\sigma(\mathbf{s_t}) + \mathbf{W_{dc}hc_t} + \mathbf{W_{sc}sc} + \mathbf{b_c}) \tag{7}$$

$$P(w_{t+1} | cl_{t+1}, s_{t-1}, \mathbf{w_t}, \mathbf{hw_t}, \mathbf{sc}) = \underset{cl_{t+1}}{\alpha}(\mathbf{W_{hw}}\sigma(\mathbf{s_t}) + \mathbf{W_{dw}hw_t} + \mathbf{W_{sw}sc} + \mathbf{b_w}) \tag{8}$$

3. Semantic feature extraction

This section presents the semantic features that are used during the training of SELMs. We present two different semantic features: (a) low-level features that directly use the semantic parse of the utterance (Semantic Context Encodings) and (b) high-level features that are trained by using deep autoencoders (Deep Semantic Encodings).

SELMs use semantic features that are extracted from utterances. To extract these semantic features we use FrameNet semantic parsers. A FrameNet semantic parser usually follows a three-step process (Das et al., 2014). The first step is the target identification task which classifies words as target and non-target words. This step is usually based on a rule-based algorithm. The second step is the frame identification step, where statistical models are used to identify the correct frames that target words evoke. The final step is the frame element identification, which is also performed by a statistical model. This step identifies and finds the span of the frame elements for each identified frame. In this paper we have designed experiments for English and Italian corpora. We have used SEMAFOR (Das et al., 2014), a FrameNet parser trained on general domain data, for English. We have used the LUNA FrameNet semantic parser which is described in Coppola et al. (2009) for the Italian LUNA corpus. The LUNA semantic parser is trained on the frames that are specific to the LUNA spoken conversation domain.

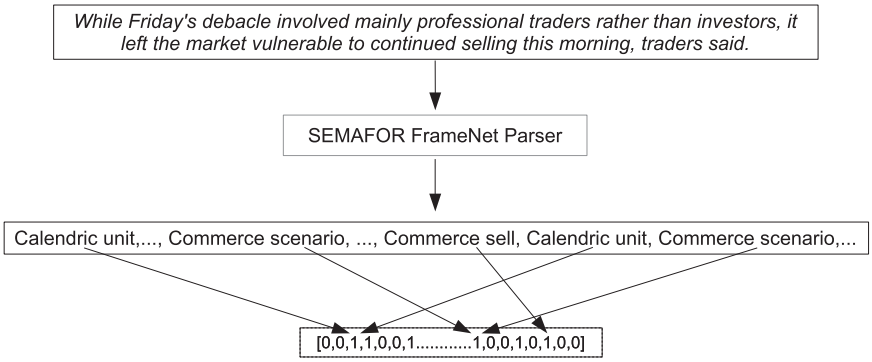

Fig. 5. Semantic feature extraction for frames. The utterance is fed into the semantic parser, and the frames are extracted. The semantic feature is a binary vector over frames, where the component at index i is set to 1, if the frame i occurs.

Semantic features are only extracted over the targets and frames of the utterances. We do not consider frame elements, since the identification of frame elements is very noisy at the current state-of-the-art.

3.1. Semantic context encodings

The low-level features are extracted directly by using the output of the semantic parser. These features are based on either the targets detected or the frames identified. Low-level features represent the targets or frames with a binary vector, in which a bit is set to 1 if the corresponding target or frame occurs in that utterance or set to 0 otherwise. Therefore, the semantic vector represents each target or frame that occurs in that utterance. For example, 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. The feature extraction step for frames is depicted in Fig. 5.

3.2. Deep semantic encodings

In this section we present deep semantic encodings which aim at reducing the size of the semantic vector space and eliminating the ASR noise in the semantic representations. The low-level feature vectors can be used to represent the linguistic context; however, they are sparse vectors in a high-dimensional space which affects their performance. Hinton and Salakhutdinov (2006) have shown that deep autoencoders can reduce the dimensionality of data with a higher precision than principal component analysis. They have shown that for document similarity tasks deep autoencoders outperform latent semantic analysis. Therefore, high-dimensional low-level semantic context can be represented in a low-dimensional space with a high-level representation by means of autoencoders.

The noise introduced by the ASR when extracting the semantic information reduces the accuracy of semantic representations. Deep semantic encodings may address this issue as well by smoothing the representations of the semantic context. We use a similar approach to *semantic hashing* (Salakhutdinov and Hinton, 2009) for constructing high-level representations of the semantic context.

Semantic hashing (Salakhutdinov and Hinton, 2009), which is used for document retrieval, 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 binary vectors and can be retrieved efficiently. In semantic hashing, a deep autoencoder that is composed of an encoder and a decoder is trained on bag-of-words representation of content words in documents. The training is performed at two steps: unsupervised pretraining by using restricted Boltzmann machines (RBMs) and fine-tuning by unrolling the network and applying back-propagation on the reconstruction error. The middle layer of the deep autoencoder (code layer) is forced to produce a binary vector by adding Gaussian noise to the input of each unit. We apply the same methodology to extract deep semantic encodings, and the details are presented in Section 3.2.1. However, we use stochastic binary units at the code layer to enforce binary vectors rather than adding Gaussian noise to the input of this layer; we apply it to utterances rather than documents; and we use the targets or the frames of the utterance rather than the content words.

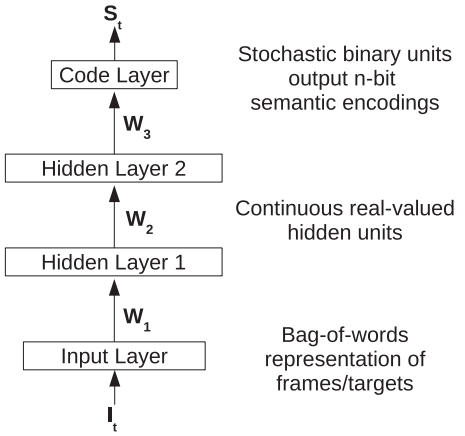

Fig. 6. The structure of the encoder network for the deep autoencoder used for semantic encoding. The input $\mathbf{I_t}$ is the normalized bag-of-words representation of targets or frames at time t. The state of the code layer $\mathbf{S_t}$ is the n-bits binary encoding of $\mathbf{I_t}$ at time t. The state of the code layer is enforced to be binary by using stochastic binary units at that layer. The hidden layers use continuous valued units. The weights between layers i and $i+1$ are represented with $\mathbf{W_i}$.

We train deep autoencoders to construct deep semantic encodings (Bayer and Riccardi, 2015). Autoencoders can be thought of as a combination of an encoder and a decoder. The encoder encodes the input to a hidden representation, whereas the decoder decodes this hidden representation to re-construct the input. The structure of the encoder that is used in this paper is given in Fig. 6. The first layer of this encoder is the input layer, which is a linear layer activated by the bag-of-words representation of the frames or targets of an utterance. The code layer encodes these representations to a binary vector. The binary representation is enforced by using stochastic binary units at that layer. The state of a stochastic binary unit is determined by a random value in the interval [0, 1] that is created at run time. If the activation of the unit is greater than that value, its state is set to 1; otherwise the state is set to 0. The activations at the code layer at time t, $\mathbf{a_t}$, is given by:

$$\mathbf{a_t} = \sigma(\mathbf{W_3}\sigma(\mathbf{W_2}\sigma(\mathbf{W_1}\mathbf{I_t} + \mathbf{b_2}) + \mathbf{b_3}) + \mathbf{b_4}) \tag{9}$$

where $\mathbf{b_i}$ is the biases of the layer i (the input layer has no biases) and $\mathbf{W_i}$ is the weights between the layers i and $i+1$. Then, the state of the ith node at time t, $st_{i,t}$ is computed by:

$$st_{i,t} = \begin{cases} 0, & \text{if} \quad a_{i,t} < rand_{i,t}(0,1) \\ 1, & \text{otherwise} \end{cases} \tag{10}$$

where $a_{i,t}$ refers to the activation of node i at time t and $rand_{i,t}(0,1)$ is a random value in the interval [0, 1] for node i at time t.

3.2.1. Training deep autoencoders

Training deep neural networks that have more than one hidden layer converges to a poor local minima if the weights are randomly initialized and then gradient descent is used to learn the weights (Hinton and Salakhutdinov, 2006). 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 this 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 the weights by greedy layer-by-layer training (Hinton et al., 2006). At the second phase, the backpropagation algorithm is used to train deep neural networks in a supervised way (Hinton and Salakhutdinov, 2006). The input given to the deep autoencoders is the normalized "bag-of-words" (BoW) vectors of frames or targets for each utterance. The deep autoencoder constructs n-bits encodings from these BoW vectors.

The first phase is the unsupervised pretraining phase as shown in Fig. 7. For this purpose, the greedy layer-by-layer training (Hinton et al., 2006) is performed. In this approach, each pair of layers is modeled by restricted Boltzmann

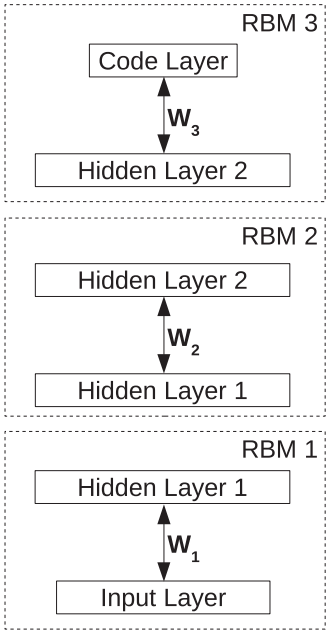

Fig. 7. The unsupervised pretraining procedure. Each pair of layer is considered as a restricted Boltzmann machine, and the whole network is trained in a greedy layer-by-layer approach. The code layer contains the units that will be used for binary encodings. All the layers use binary valued units except for the nodes at the input layer, which use discrete valued units.

machines (RBMs) and each RBM is trained from bottom to top. RBMs have a visible layer and a hidden layer, where the nodes in the same layer are not connected to each other. Parameters of RBMs are defined as follows:

- **W**: The weights between the nodes of the visible layer and the hidden layer (W_{ij} is the weight between the *ith* visible node and *jth* hidden node).
- **bv**: The biases of the visible layer.
- **bh**: The biases of the hidden layer.

During the pretraining phase, the visible layer of the bottom RBM (RBM 1) is modeled by a constrained Poisson model as given in Salakhutdinov and Hinton (2009), where the nodes of the visible layer have discrete values and the nodes of the hidden layer have binary values. The probability of the *ith* node at the visible layer to have the activation value of n is given by:

$$p(v_i = n|\mathbf{h}) = Ps\left(n; \alpha\left(\mathbf{W}^{\mathbf{T}}\mathbf{h} + \mathbf{bv}\right)_i * N\right) \tag{11}$$

where $\alpha(x)_i$ is the softmax function defined by Equation 2, N is the sum of the all activation at the visible layer, **h** is the activations of the hidden layer, and $Ps(n, \lambda)$ is defined by the probability mass function of X that has a Poisson distribution with mean λ.

$$Ps(n; \lambda) = Pr(X = n) = \frac{e^{-\lambda}\lambda^n}{n!} \tag{12}$$

The probability of the *jth* node at hidden layer to have a value of 1 is given by:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(bh_j + \sum_i W_{ij}v_i\right) \tag{13}$$

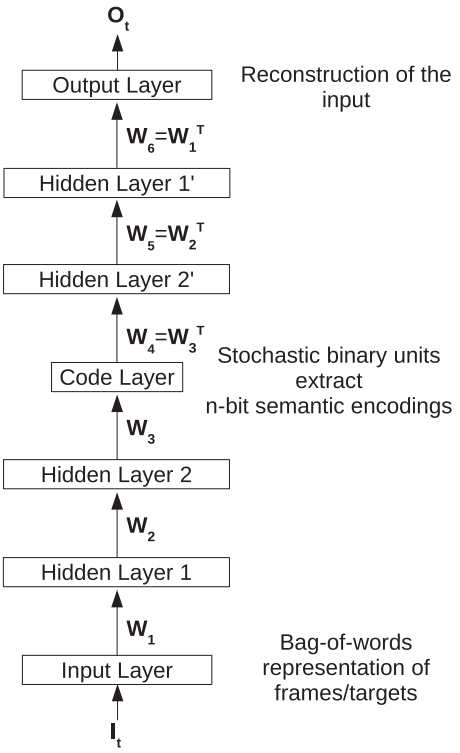

Fig. 8. The fine-tuning phase of the deep autoencoder. The autoencoder unrolled such that $\mathbf{W_i^T}$ corresponds to the transpose of the $\mathbf{W_i}$ weight matrix of the RBM pairs in the pretraining phase. At the output layer, $\mathbf{O_t}$ is the reconstruction of the normalized BoW representation of the input $\mathbf{I_t}$ by using the softmax function. Stochastic binary units are used to enforce binary encodings at the code layer. The network is fine-tuned by using the backpropagation algorithm with the cross-entropy loss function.

where $\mathbf{v}$ is the activation of the visible layer. Therefore during pretraining, unnormalized BoW vectors are used as the input when the activations of the hidden layer are computed. The softmax activation function is used for the reconstruction of the input and multiplied by the total number of frames or targets in the input. The other RBMs use only binary units at both layers, therefore the probability of the *jth* node in the hidden layer to have a value of 1 is given by Equation 13. The probability of the *ith* node in the visible layer to have a value of 1 is given as:

$$p(v_i = 1|\mathbf{h}) = \sigma\left(bv_i + \sum_j W_{ij}h_j\right) \tag{14}$$

The network is pretrained by using the single-step contrastive divergence CD_1 (Hinton, 2002).

In the second phase, the network is unrolled as shown in Fig. 8. Here, $\mathbf{W_i^T}$ corresponds to the transpose of the $\mathbf{W_i}$ weight matrix and only used for initialization. The output layer uses the softmax function and reconstructs the normalized BoW input vector, the code layer uses stochastic binary units and its state is defined by Equation 10, the other hidden layers use the sigmoid activation function. Therefore the output is given by Equation 15 by using the state of the code layer at time t, st_t that is given in Equation 10.

$$\mathbf{O_t} = \alpha(\mathbf{W_6}\sigma(\mathbf{W_5}\sigma(\mathbf{W_4}\mathbf{st_t} + \mathbf{b_5}) + \mathbf{b_6}) + \mathbf{b_7}) \tag{15}$$

where $\mathbf{O_t}$ is the output of the autoencoder at time t, $\mathbf{W_i}$ are the weights between layers i and $i+1$, and $\mathbf{b_i}$ is the biases of the layer i. The backpropagation algorithm is used to fine-tune the weights by using the reconstruction error at the output layer. The error function is the cross-entropy between the input $\mathbf{I}$ and the output $\mathbf{O}$ that is given by Equation 16, where n is the dimension of $\mathbf{I}$ and $\mathbf{O}$.

$$E(\mathbf{I}, \mathbf{O}) = -\sum_n I_n log(O_n) \tag{16}$$

During the fine-tuning phase at the code layer, the state values given in Equation 11 is used for the forward-pass. However, when backpropagating the errors, the actual activation values that are computed by Equation 9 are used.

3.3. Training SELMs

The training of SELMs are performed at two steps. The first step is the semantic feature extraction step that is described before. After the features are extracted, the training of the SELM is performed by first randomizing the training data. Then the current word, hash functions of n-gram histories, and the semantic feature that is extracted for that utterance is fed into the network. The weights and biases of the neural network are learned by using backpropagation through time (BPTT) algorithm, i.e., by unrolling the network for a number of time-steps back (Bodén, 2002). The cross entropy is used as the error function. The network state is reset after the training is done for an utterance. Therefore, SELMs are trained independently for each sentence. The learning rate is adjusted by using the gain on the development set and early stopping is performed to avoid overfitting.

4. Experimental setting

We assess the performance of SELMs by performing re-scoring experiments on two different settings. The first corpus we have used is the Wall Street Journal (WSJ) read-speech corpus (Paul and Baker, 1992). WSJ corpus is a publicly available speech corpus that is designed for the speech recognition task. The corpus consists of read articles from Wall Street Journal. Since it is read speech the performance of ASR is expected to be good. Also the domain of the utterance is generic, because it contains news articles. The second corpus we have used is the LUNA spoken dialogue corpus (Dinarelli et al., 2009). The LUNA corpus consists of human–human Italian spoken conversations between customers and operators in a help-desk call center. The two corpora are very different in terms of speaking styles, recording conditions, and lexica. For this reason they provide a very good performance benchmark for the automatic speech recognition and understanding tasks.

The performance of SELMs is assessed by the re-scoring framework that is depicted in Fig. 9. In this framework each test utterance is fed into the ASR system. The ASR system outputs an n-best list of hypotheses as well as the 1st-best hypothesis. The 1st-best hypothesis is passed to the semantic parser. The output of the semantic parser is used by the feature extraction step which either extracts low-level features (Semantic Context Encodings) or high-level features (Deep Semantic Encodings). These features are used as the semantic context for that utterance. The n-best list (100-best in this paper) is re-scored by the SELM that uses the semantic features, and the best hypothesis of the SELM is found.

The evaluation of the performance is done by considering the recognition and understanding requirements of spoken language systems. The recognition performance is reported on WER for WSJ since we only have reference transcriptions for this corpus. However, for LUNA we report WER and TER for the recognition performance. For the understanding task on LUNA we report the accuracy on the frame identification by using the F1-score. TER is important in measuring the potential understanding performance of the systems by measuring the recognition performance on the main meaning bearing elements of semantic frames. TER can be used as a proxy for the understanding performance because target words are the main meaning bearing elements of semantic frames and the accuracy of frame identification depends on how accurately targets are detected.

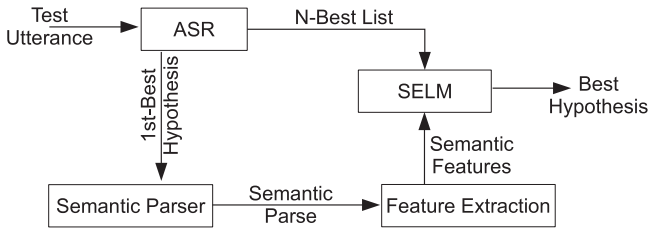

Fig. 9. The re-scoring framework. The ASR 1st-best hypothesis is given to the semantic parser. The semantic features extracted by using the semantic parse. The n-best hypotheses are re-scored by using the SELM that uses these semantic features for that utterance.

5. Wall Street Journal experiments

We have performed n-best re-scoring experiments on Wall Street Journal (WSJ) speech recognition corpus to assess the performance of these models on WER. WSJ experiments are conducted on 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 independent 20K vocabulary are used as the development set ("Dev93" - 503 utterances). The evaluation is done on the November 92 CSR Speaker independent 20k NVP test set ("Test92" - 333 utterances) and on the November 93 CSR HUB 1 test set ("Test93" - 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 are used for LM training are the whole WSJ 87, 88, and 89 sets.

5.1. ASR baselines

The baseline ASR system is built by using the Kaldi (Povey et al., 2011) 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. We have not used advanced acoustic models since we would like to test our approach under high WER condition and address the effect of ASR noise on semantic information. Also our main purpose is to compare the performance of SELMs with RNNMEs.

The ASR baseline performs weighted finite state decoding. We have extracted 100-best lists for the development and each evaluation set. WER and TER performance of the ASR and the oracle hypotheses are given in Table 1. The oracle hypotheses are the hypotheses that give the lowest WER on the 100-best list.

The semantic frames and targets are extracted by using the semantic-frame parser SEMAFOR (Das et al., 2014). WSJ speech recognition corpus has 841 distinct frames and 29043 distinct targets on the LM training data. SEMAFOR recognizes around 40% of the tokens as targets for WSJ corpus.

5.2. Low-level semantic features

This section presents the experiments when low-level semantic features are used for the semantic context. We present two different sets of features; the first one is based on frames that are evoked in the utterance and the second one target words of the utterance. The feature extraction process is performed as given in Section 3.1.

The frame and target vocabulary for the WSJ corpus is 841 and 29043 respectively. To reduce the computational complexity of the SELMs, we have used the frames and targets that cover the 80% of the corpus. With 80% coverage, frame vocabulary becomes 184 and target vocabulary becomes 1184. We train two different SELMs; one uses frame features and the other one uses target features that are constructed on the 80% coverage vocabulary. The semantic features are extracted from the ASR hypothesis; however, to show the potential performance of SELMs we have also extracted features by using the reference transcriptions. We present the results both for the features from the ASR hypothesis (ASR Frames/Targets) and for the features from the reference transcription (Ref Frames/Targets). To compare the performance of SELMs, we have also trained a Kneser-Ney smoothed 5-gram LM (KN5) and an RNNME LM on the same data. The SELMs and the RNNME LM use 200 hidden units and are class-based models over 200 word classes that are determined with respect to the unigram occurrences of the words. They use up to 4-gram maximum entropy features by using 10^9 connections. The results are given in Table 2. As can be seen

Table 1
The WER(%) performance of the ASR baseline system on Test92 and Test93 sets.

	Test92	Test93
ASR 1st-best	10.2	14.0
Oracle on 100-best	5.1	7.3

Table 2
The WER(%) performance on the re-scored 100-best lists.

Model	Test92	Test93
KN5	9.7	13.3
RNNME	8.8	12.7
SELM on frames		
ASR frames	**9.2**	**13.6**
Ref frames	8.5	12.5
SELM on targets		
ASR targets	**9.6**	**13.7**
Ref targets	7.9	11.3

KN5 is a Kneser–Ney 5-gram LM with singleton cut-offs. RNNME and SELMs are class-based models with 200 word classes. The actual performance of SELMs are given in bold, the results with Ref frames/targets show the possible lower bound.

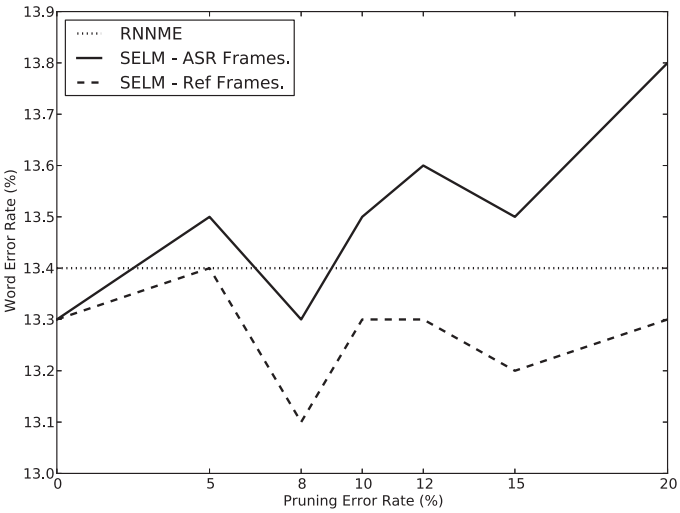

Fig. 10. The WER performance of SELM on frames on the Dev93 set with different pruning error rates.

SELMs have high potential when the correct semantic information is used (Ref Frames/Targets); however, the noise that is present in the ASR hypothesis drops their performance significantly beyond the RNNME LM.

Noise reduction can be performed by using the semantic information that is more accurate. Since frames are more robust to ASR noise we perform error pruning only on the frames for WSJ. For this purpose, we use Dev93 set and align the output of the semantic parser on the ASR 1st-best hypothesis with the output of the semantic parser on the reference transcription. We calculate the error rate of each frame by counting all the errors (substitutions, insertions, deletions) that are made on each frame. We discard the frames that have error rates above a certain threshold when extracting low-level semantic features for the training and the test sets. We train SELMs from scratch by using the semantic contexts after error pruning. And we perform the re-scoring experiments once more. Fig. 10 shows the WER on the development set with various pruning rates. It can be seen that, in general, as the threshold increases the WER increases. We can also observe that the WER is the lowest with pruning error rates of 0% and 8%, which correspond to 37 and 52 distinct frames in the semantic information.

The WER performance of SELMs with error pruning on the test set is given in Table 3. We present the performance of two models, SELMs on frames with the error pruning threshold of 0%, and SELMs on frames with the error pruning threshold of 8%. For Test92 set the model with 0% pruning threshold performs better; however, for Test93 set both models perform the same. Although we have a slight improvement for Test92 set with SELMs, we cannot achieve any improvement for Test93.

Table 3
The WER(%) performance after error pruning.

Model	Test92	Test93
KN5	9.7	13.3
RNNME	8.8	12.7
SELM frames err. = 0%		
ASR frames	**8.7**	**12.7**
Ref frames	8.7	12.7
SELM frames err. = 8%		
ASR frames	**8.9**	**12.7**
Ref frames	8.8	12.5

The actual performance of SELMs are given in bold, the results with Ref frames show the possible lower bound.

5.3. Deep semantic encodings

Error pruning on low-level semantic features helps to improve WER for Test92. The performance of error pruning depends on the set where the pruned frames are selected and may not work well for the unseen data. Also, by pruning erroneous frames, the whole linguistic scene cannot be employed in the language model. This section presents the experiments performed by using deep semantic encodings, where the whole semantic information is encoded by using deep autoencoders. In addition, the semantic encodings handle the noise of ASR frames and targets by encoding the semantic information in a noisy way by using stochastic binary units.

The deep semantic encodings are constructed for the frames and targets separately. Therefore, we have two different sets of 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 are available for LM training, in addition, the development set is used to avoid overfitting. To obtain the frames and 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 to avoid overfitting by adjusting the learning rate and by "early stopping".

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

To investigate the accuracy of the encodings of various sizes, we compare ASR encodings with reference encodings. Semantic encodings are binary vectors; therefore 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. Fig. 11 shows the histograms of frame and target encodings in the interval [1, 8].[3] We observe that in general, as the size of the encodings increase the number of instances with a higher Hamming distance increase. For the frame encodings, the sizes of 8-bits and 12-bits have a similar distribution which have a better performance of suppressing the ASR noise. However, in particular, the sizes of 20-bits and 24-bits are very much affected by the ASR noise and show high discrepancy between the ASR and the reference encodings. For the target encodings, the size of 8-bits has the lowest error distribution and size of 24-bits 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 the same sizes of frame and target encodings are compared with each other).

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

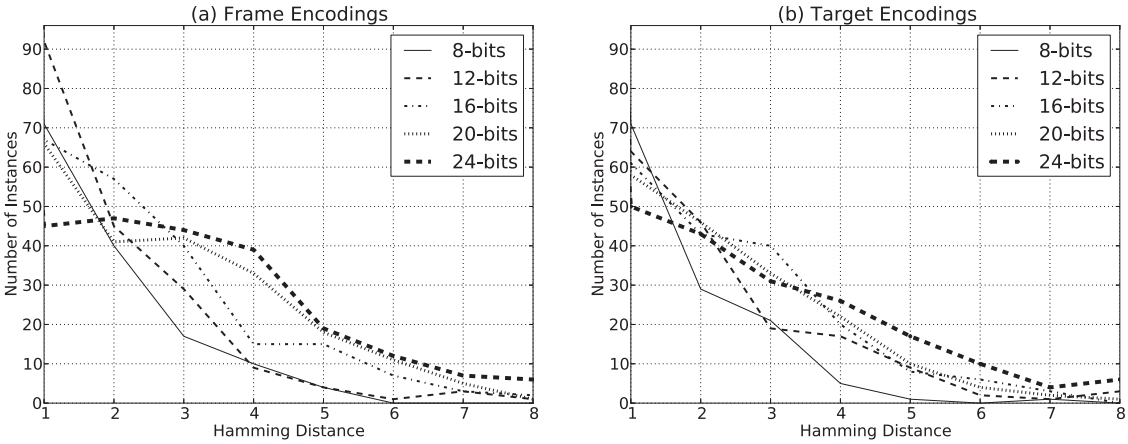

Fig. 11. The histogram of Hamming distance between ASR encodings and reference encodings on the development set: (a) Frame Encodings, (b) Target Encodings. The encodings of 8-bits and 12-bits have a better performance. Target encodings are more robust in general. The histogram is shown for the Hamming distance in the interval [1, 8].

Table 4
The WER(%) performance of SELMs with deep semantic encodings.

Model	Test92	Test93
KN5	9.7	13.3
RNNME	8.8	12.7
SELM – frame enc.		
8-bits	**8.5**	**12.4**
12-bits	8.6	13.0
16-bits	9.1	12.6
20-bits	8.5	12.5
24-bits	9.2	12.8
SELM – target enc.		
8-bits	**8.5**	**12.4**
12-bits	**8.5**	**12.4**
16-bits	8.7	12.5
20-bits	9.0	12.7
24-bits	8.8	13.1

The best performing models for both test sets are given in bold.

The performance of the SELMs that are trained on deep semantic encodings are evaluated on WER performance. The performance of these models are presented with the KN5 and RNNME model for comparison. The WER performance of these models are presented in Table 4.

We can see in Table 4 that the SELMs with 8-bits and 12-bits target encodings, and 8-bits and 20-bits frame encodings always perform better than the RNNME. We also see that frame encodings are more noisy, because of the performance differences between Test92 and Test93; but for target encodings the situation is more stable, the 8-bits and 12-bits 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.

5.4. Discussion

The re-scoring experiments on WSJ corpus show that SELMs can be optimized for WER jointly by using either low-level semantic features or deep semantic encodings. We achieve a more stable improvement by using deep semantic encodings compared to error pruning. Error pruning cannot utilize all the semantic context and most often depends on the data that the error pruning is performed. Deep semantic encodings, on the other hand, utilize all the available semantic information. We observe that target encodings are more robust.

6. LUNA HH experiments

This section presents the re-scoring experiments that are conducted on the Italian LUNA human–human conversational speech corpus (LUNA HH). LUNA HH corpus consists of human–human spoken dialogs that are recorded between a customer and an operator at a call center. Therefore, the corpus consists of domain specific dialogs. The dialogs on LUNA HH corpus is annotated at multiple levels (Dinarelli et al., 2009). These levels include the word level annotation and predicate argument annotation. The predicate argument annotation is based on the FrameNet model. In addition, the dialogs are recorded on a single channel. Because of the nature of human–human conversations these dialogs contain overlapped segments which are discarded. The number of dialogs in LUNA HH corpus is 415, 53, 84 for the training, the development and the test set respectively. The FrameNet annotations, which are used for the training of the semantic parser, are only available for 94, 11, 20 dialogs in the training, development and the test set. In the experiments conducted on LUNA HH we have used the whole training and the development sets. However, for reporting both the recognition and the understanding performance, the results are reported on the subset of the test set that has semantic annotations. We will refer to this subset as the test set from now on. The statistics on LUNA HH corpus is given in Table 5.

6.1. Semantic parsing on LUNA HH

The FrameNet annotation on LUNA HH corpus is domain specific, i.e., only the frames that are semantically relevant to the domain are annotated (Dinarelli et al., 2009). Therefore, the LUNA semantic parser only extracts domain specific target words and frames, whereas SEMAFOR parser extracts all of the target words and frames that are defined under the FrameNet project. In this respect, the targets and frames are sparse for LUNA HH corpus. The frame vocabulary is 203.

The LUNA semantic parser (Coppola et al., 2009) is mainly focused on frame element recognition and outputs multiple hypotheses at the frame identification step. The frames are scored by using a very simple model. To improve the frame identification performance of the LUNA semantic parser we have re-scored the multiple hypotheses by using another semantic model that is trained by using conditional random fields (CRFs). The target identification and frame identification performance of the semantic parser with re-scoring for the reference transcription of the test set is given in Table 6.

6.2. ASR baseline

The ASR baseline for LUNA HH is constructed by using the Kaldi (Povey et al., 2011) speech recognition toolkit. The ASR uses mel-frequency cepstral coefficients (MFCC) that are transformed by linear discriminant analysis (LDA)

Table 5
Statistics of the whole training split and the subset of the test set that has semantic annotations of LUNA HH corpus.

	Training	Test
No. of utterances	14465	900
No. of tokens	116178	6273
Vocabulary size	6840	1287
OOV rate (%)	–	3.5

Table 6
The target identification and frame identification performance of the LUNA semantic parser with re-scoring on the transcriptions of the test set.

	Precision (%)	Recall (%)	F1-score (%)
Target identification	79.3	70.4	74.6
Frame identification	63.6	56.5	59.8

Table 7
The WER(%) and TER(%) performance of the baseline ASR on LUNA HH corpus.

	WER(%)	TER(%)
ASR	35.7	66.7
Oracle on 100-best	23.0	60.5

The oracle performance is given for 100-best lists.

Table 8
The target identification and frame identification accuracy of the ASR baseline on the test set of LUNA HH corpus.

	Precision (%)	Recall (%)	F1-score (%)
Target identification	60.3	50.9	55.2
Frame identification	50.4	44.0	47.0

Table 9
The WER(%) performance of the re-scoring experiments on LUNA HH corpus by using the SELMs that are trained over frames and targets.

	WER(%)	TER(%)
RNNME	34.6	66.7
SELM on frames		
ASR frames	**35.5**	**66.4**
Ref frames	34.6	64.4
SELM on targets		
ASR targets	**35.1**	**67.0**
Ref targets	34.0	61.4

The actual performance of SELMs are given in bold, the results with Ref frames/targets show the possible lower bound.

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) (Leggetter and Woodland, 1995). 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 7. Because of the nature of the corpus, i.e., human–human conversations over the telephone, the corpus is very noisy and has a high WER.

We present the performance of the ASR baseline on frame accuracy and target accuracy by using the Italian LUNA semantic-frame parser (Coppola et al., 2009). The semantic parser recognizes around 10% of the tokens as targets; therefore the targets for LUNA are more sparse than WSJ. The target recognition and frame identification accuracy is given in Table 8. The high WER on LUNA HH also affects the semantic parsing performance; therefore, we have very low accuracy on the ASR hypothesis.

6.3. 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. Thus we have used 151 distinct frames and 536 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 initialized with the same random weights. The results of the re-scoring experiments are given in Table 9. In this table, "ASR Frames/Targets" means that we have given the ASR 1st-best hypothesis to the semantic parser, and "Ref 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". "Ref Frames/Targets" shows a lower bound to WER.

We observe a similar situation for LUNA. When the accurate semantic information (Ref Frames/Targets) is used, the SELMs have a significant improvement on WER. However, although the SELMs with "ASR Frames" perform

Table 10
The WER(%) and TER(%) performance with error pruning at 0%, 20%, and 40% on LUNA HH.

	WER(%)	TER(%)
RNNME	34.6	66.7
SELM on frames – err. = 0% (15 frames)		
ASR frames	**34.7**	**66.1**
Ref frames	34.7	66.1
SELM on frames – err. = 20% (24 frames)		
ASR frames	**35.1**	**66.4**
Ref frames	35.0	66.3
SELM on frames – err. = 40% (47 frames)		
ASR frames	**36.1**	**66.9**
Ref frames	35.9	65.9
SELM on targets – err. = 0% (20 targets)		
ASR targets	**35.0**	**65.5**
Ref targets	35.0	65.5
SELM on targets – err. = 20% (47 targets)		
ASR targets	**35.0**	**66.4**
Ref targets	35.0	66.7
SELM on targets – err. = 40% (173 targets)		
ASR targets	**34.9**	**67.1**
Ref targets	34.8	65.1

The number of distinct frames and targets with the corresponding thresholds are given in parenthesis. The actual performance of SELMs are given in bold, the results with Ref frames/targets show the possible lower bound.

better than the ASR baseline, it fails to outperform the RNNME model. Therefore, we apply the error pruning methodology and prune erroneous targets and frames.

6.4. Error pruning

We perform error pruning on the training data by measuring the target and frame errors that are present in the ASR hypotheses with respect to the reference transcription. Therefore, as done for WSJ, we have aligned the output of the semantic parser (both on targets and frames) for the ASR hypothesis with the reference transcription. All the errors are computed (substitutions, insertions, and deletions) for each target and frame. The targets and frames which have errors more than a threshold are discarded from all the semantic context (both for the train and test sets). The thresholds for pruning are 0%, 20%, and 40%. The SELMs are trained from scratch and re-scoring experiments are performed once more. The performance of SELMs are presented in Table 10.

The results show that error pruning improves the results compared to training with all the semantic features. We observe that as we add targets and frames that have errors, the performance has a tendency to drop. The best performance is obtained by using the frames and targets that do not have any errors. In terms of WER, the SELMs trained for LUNA HH corpus have a lower performance compared to RNNME. However, we achieve a lower TER for SELMs with error pruning up to 20%. Therefore, although the high WER on the ASR hypothesis prevents SELMs to be trained effectively, they still have a better performance on the meaning bearing elements.

6.5. Understanding performance

We present the understanding performance of the SELMs by performing a detailed analysis on TER and frame identification accuracy. We perform this detailed analysis with the RNNME model, the SELM on frames with error pruning at 0%, the SELM on targets with error pruning at 0%, and the linear interpolation of these two SELMs with equal weights. The detailed analysis of TER and frame identification accuracy is given in Fig. 12.

In terms of TER we achieve a better performance with all of the SELMs at every coverage of target words compared to RNNME. In addition, we observe that the SELMs on targets perform the best. We also observe that the performance gain is greater for the target words that occur more frequently. On the frame identification, we observe that

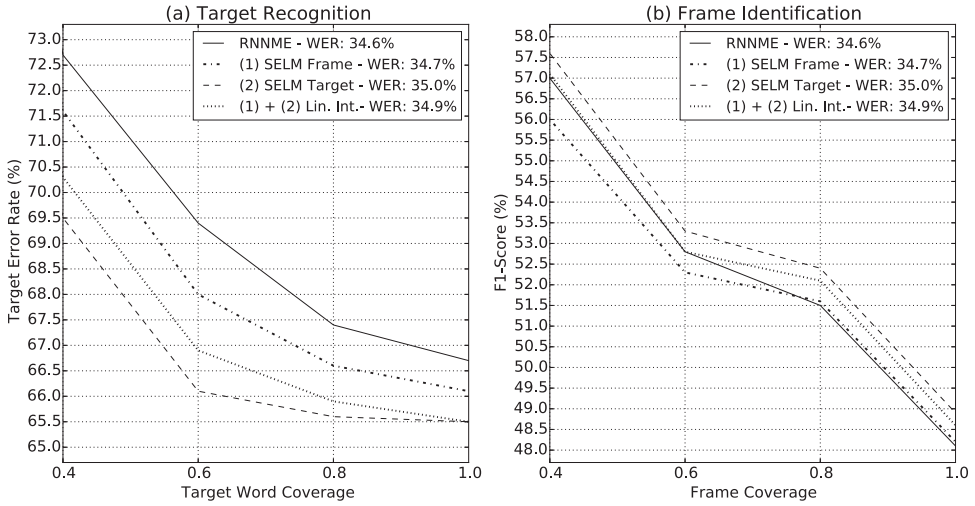

Fig. 12. Target recognition (TER) and frame identification (F1-score) performance at different coverages of target words and frames for LUNA HH: (a) target recognition, (b) frame identification.

SELMs on frames perform worse than RNNME up to 80% coverage, and for the full coverage it performs slightly better than RNNME. SELMs on targets, on the other hand, outperform RNNMEs.

6.6. Discussion

The experiments on LUNA HH, which is a domain specific conversational corpus, have a noisy setting. The ASR 1st-best performance is very noisy in terms of recognition and the understanding performance. In addition, the semantic frame annotation only includes domain specific frames which lead to sparsity of the semantic information.

The sparse semantic information and the small size of LUNA HH prevents us to train deep semantic encodings. Therefore, we have only presented the SELMs with low-level semantic features. Under these conditions, we observe that SELMs fail to outperform RNNMEs on WER; however, they achieve better performance in terms of TER and frame identification accuracy. The SELM on targets has the best performance for both TER and frame identification accuracy.

7. Conclusion

In this paper we have presented SELMs that integrate semantic constraints into language modeling by using the theory of frame semantics. SELMs incorporate semantic features into LMs and enable them to be optimized with respect to lexical and semantic constraints jointly.

We have performed re-scoring experiments on WSJ and LUNA HH to assess the performance of SELMs. The performance is evaluated on WER for WSJ corpus and on WER, TER, and frame identification accuracy for LUNA HH corpus. We have used two different types of semantic features: low-level features which are extracted directly from the output of the semantic parser and high-level features (deep semantic encodings) which are extracted by using deep autoencoders. Low-level features are affected by the noise in the ASR hypothesis, and they can only be employed after erroneous frames and targets are pruned. Deep semantic encodings, on the other hand, suppress the ASR noise by smoothing the encodings. Deep semantic encodings are more robust and they outperform low-level features on WER for WSJ. On the other hand, low-level features may not be optimized for WER; however, they still perform well for the understanding performance.

The experiments are performed on two different settings with low ASR noise and dense (in terms of triggered frames per token) semantic features (WSJ), and high ASR noise and sparse semantic features (LUNA HH). We observe that in the case where the ASR noise is low and the semantic features are dense, SELMs can be optimized better with respect to WER. The high ASR noise prevents SELMs to be optimized for WER; however, they still outperform RNNMEs on the understanding performance.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 610916 – SENSEI.

References

Bayer, A.O., Riccardi, G., 2014. Semantic language models for automatic speech recognition. In: Spoken Language Technology Workshop (SLT), 2014 IEEE. IEEE, pp. 7–12.

Bayer, A.O., Riccardi, G., 2015. Deep semantic encodings for language modeling. In: Interspeech 2015, 16th Annual Conference of the International Speech Communication Association, pp. 1448–1452.

Bellegarda, J., 2000a. Exploiting latent semantic information in statistical language modeling. P. IEEE 88 (8), 1279–1296.

Bellegarda, J., 2000b. Large vocabulary speech recognition with multispan statistical language models. IEEE Trans. Speech Audio Process. 8 (1), 76–84.

Bengio, Y., Ducharme, R., Vincent, P., Janvin, C., 2003. A neural probabilistic language model. J. Mach. Learn. Res. 3, 1137–1155.

Berry, M.W., Dumais, S., O'Brien, G., 1995. Using linear algebra for intelligent information retrieval. SIAM Rev. 37, 573–595.

Bodén, M., 2002. A guide to recurrent neural networks and backpropagation. In: In the Dallas project, SICS Technical Report T2002:03, SICS.

Chelba, C., Bikel, D., Shugrina, M., Nguyen, P., Kumar, S., 2012. Large scale language modeling in automatic speech recognition. Tech. rep., Google.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., 2013. One billion word benchmark for measuring progress in statistical language modeling. CoRR abs/1312.3005. <http://arxiv.org/abs/1312.3005> (accessed 03.15).

Coppola, B., Moschitti, A., Riccardi, G., 2009. Shallow semantic parsing for spoken language understanding. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers. NAACL-Short '09. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 85–88.

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

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., 1990. Indexing by latent semantic analysis. J. Am Soc. Inf. Sci. 41 (6), 391–407.

Deoras, A., Tur, G., Sarikaya, R., Hakkani-Tur, D., 2013. Joint discriminative decoding of words and semantic tags for spoken language understanding. IEEE Trans. Audio Speech Lang. Process. 21 (8), 1612–1621.

Dinarelli, M., Quarteroni, S., Tonelli, S., Moschitti, A., Riccardi, G., 2009. Annotating spoken dialogs: from speech segments to dialog acts and frame semantics. In: Proceedings of SRSL 2009 Workshop of EACL. Athens, Greece.

Fillmore, C.J., 1976. Frame semantics and the nature of language. Ann. N. Y. Acad. Sci. 280 (1), 20–32.

Fillmore, C.J., Johnson, C.R., Petruck, M.R.L., 2003. Background to Framenet. Int. J. Lexicogr. 16 (3), 235–250.

Gildea, D., Hofmann, T., 1999. Topic-based language models using EM. In: Proceedings of the 6th European Conference on Speech Communication and Technology (EUROSPEECH-99). Budapest, pp. 2167–2170.

Hinton, G.E., 2002. Training products of experts by minimizing contrastive divergence. Neural Comput. 14 (8), 1771–1800.

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313 (5786), 504–507.

Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (7), 1527–1554.

Leggetter, C.J., Woodland, P.C., 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Comput. Speech Lang. 9 (2), 171–185.

Marcus, M.P., Santorini, B., Marcinkiewicz, M.A., 1993. Building a large annotated corpus of English: The Penn Treebank. Comput. Ling. 19 (2), 313–330.

Mikolov, T., 2012. Statistical language models based on neural networks. (Ph.D. thesis), Brno University of Technology.

Mikolov, T., Zweig, G., 2012. Context dependent recurrent neural network language model. In: Proceedings of SLT. IEEE, pp. 234–239.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S., 2010. Recurrent neural network based language model. In: Interspeech 2010, 11th Annual Conference of the International Speech Communication Association, pp. 1045–1048.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., Cernocký, J., 2011a . Empirical evaluation and combination of advanced language modeling techniques. In: Interspeech 2011, 12th Annual Conference of the International Speech Communication Association, pp. 605–608.

Mikolov, T., Deoras, A., Povey, D., Burget, L., Cernocký, J., 2011b. Strategies for training large scale neural network language models. In: Proceedings of ASRU. IEEE, pp. 196–201.

Mikolov, T., Kombrink, S., Burget, L., Cernocký, J., Khudanpur, S., 2011c. Extensions of recurrent neural network language model. In: Proceedings of ICASSP. IEEE, pp. 5528–5531.

Paul, D.B., Baker, J.M., 1992. The design for the Wall Street Journal-based CSR corpus. In: Proceedings of the Workshop on Speech and Natural Language. HLT '91. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 357–362.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., et al., 2011. The Kaldi speech recognition toolkit. In: Proceedings of ASRU. IEEE.

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

Rosenfeld, R., 1996. A maximum entropy approach to adaptive statistical language modeling. Comput. Speech Lang. 10, 187–228.

Salakhutdinov, R., Hinton, G., 2009. Semantic hashing. Int. J. Approx. Reason. 50 (7), 969–978.

Schwartz, R.M., Imai, T., Kubala, F., Nguyen, L., Makhoul, J., 1997. A maximum likelihood model for topic classification of broadcast news. In: Proceedings of EUROSPEECH. ISCA.

Schwenk, H., 2007. Continuous space language models. Comput. Speech Lang. 21 (3), 492–518.
Tonelli, S., Riccardi, G., 2010. Guidelines for annotating the LUNA corpus with frame information. In: Technical Report DISI-10-017, University of Trento.
Wang, Y.-Y., Acero, A., Chelba, C., 2003. 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, pp. 577–582.