A study of continuous space word and sentence representations applied to ASR error detection

Sahar Ghannay, Yannick Estève, Nathalie Camelin

▶ **To cite this version:**

Sahar Ghannay, Yannick Estève, Nathalie Camelin. A study of continuous space word and sentence representations applied to ASR error detection. Speech Communication, 2020. hal-02501943

HAL Id: hal-02501943
https://hal.science/hal-02501943

Submitted on 8 Mar 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A study of continuous space word and sentence representations applied to ASR error detection

Sahar Ghannay[a,*], Yannick Estève[a,*], Nathalie Camelin[a,*]

[a] *LIUM - University of Le Mans, France*

Abstract

This paper presents a study of continuous word representations applied to automatic detection of speech recognition errors. A neural network architecture is proposed, which is well suited to handle continuous word representations, like word embeddings. We explore the use of several types of word representations: simple and combined linguistic embeddings, and acoustic ones associated to prosodic features, extracted from the audio signal. To compensate certain phenomena highlighted by the analysis of the error average span, we propose to model the errors at the sentence level through the use of sentence embeddings. An approach to build continuous sentence representations dedicated to ASR error detection is also proposed and compared to the Doc2vec approach.

Experiments are performed on automatic transcriptions generated by the LIUM ASR system applied to the French ETAPE corpus. They show that the combination of linguistic embeddings, acoustic embeddings, prosodic features, and sentence embeddings in addition to more classical features yields very competitive results. Particularly, these results show the complementarity of acoustic embeddings and prosodic information, and show that the proposed sentence embeddings dedicated to ASR error detection achieve better results than generic sentence embeddings.

Keywords: ASR error detection, neural networks, prosodic features, linguistic embeddings, acoustic embeddings, sentence embeddings.

1. Introduction

Recent advances in the field of speech processing have led to significant improvements in speech recognition performance. However, recognition errors are still unavoidable, whatever the quality of the ASR system. This reflects the

☆

*Corresponding author. Permanent address: LIUM-University of Le Mans, France. Tel.: +33 243833858

Email addresses: sahar.ghannay@limsi.fr (Sahar Ghannay), yannick.esteve@univ-avignon.fr (Yannick Estève), nathalie.camelin@univ-lemans.fr (Nathalie Camelin)

system sensitivity to variability, *e.g.*, to acoustic conditions, speaker, language style, *etc.* These errors may have a considerable impact on applications based on the use of automatic transcriptions, like information retrieval, speech-to-speech translation, spoken language understanding, *etc.*

Error detection aims at improving the exploitation of ASR outputs by downstream applications, but it is a difficult task because there are several types of errors, which can range from simple mistakes on word morphology, such as number agreement, to the insertion of irrelevant words, which affect the overall understanding of the word sequence.

For two decades, many studies have focused on the ASR error detection task. Usually, the best ASR error detection systems are based on the use of Conditional Random Fields (CRF) [1]. In [2], the authors detect error regions generated by Out Of Vocabulary (OOV) words. They propose an approach based on a CRF tagger, which takes into account contextual information from neighboring regions instead of considering only the local region of OOV words. A similar approach for other kinds of ASR errors is presented in [3]: the authors propose an error detection system based on a CRF tagger using various ASR-derived, lexical and syntactic features.

Recent approaches leverage neural network classifiers. A neural network trained to locate errors in an utterance using a variety of features is presented in [4]. Some of these features are gathered from forward and backward recurrent neural network language models in order to capture long distance word context within and across previous utterances. The other features are extracted from two complementary ASR systems. In [5], authors propose to use a neural network classifier furnished by stacked auto-encoders (SAE), that helps to learn the error word representations. In [6, 7], the authors investigated three types of ASR error detection tasks, *e.g.* confidence estimation, out-of-vocabulary word detection and error type classification (insertion, substitution or deletion), based on deep bidirectional recurrent neural networks.

In our previous researches [8, 9, 10] we studied the use of several types of continuous word representations. In [8], we proposed a neural approach to detect errors in automatic transcriptions, and to calibrate confidence measures provided by an ASR system. In addition, we studied different word embeddings combination approaches in order to take benefit from their complementarity. The proposed ASR error detection system integrates several information sources: syntactic, lexical, ASR-based features, prosodic features as well as linguistic embeddings.

We proposed as well to enrich our ASR error detection system with acoustic information which is obtained through acoustic embeddings. We showed in [10] that acoustic word embeddings capture additional information about word pronunciation in comparison to the information supported by their spelling. We showed that these acoustic embeddings are better than orthographic embeddings to measure the phonetic proximity between two words. Moreover, the use of these acoustic embeddings in addition to other features improved the performance of the proposed ASR error detection system [9].

In this paper, we first propose a summary of our previous studies, we report:

- the performance obtained by the combined linguistic embeddings
- the approach we used to build acoustic embeddings
- and the evaluation of the combination of linguistic and acoustic embeddings in the framework of the ASR error detection task

Then, we present new contributions on the combination of prosodic features and acoustic embeddings, and about sentence embeddings to characterize the level of reliability of entire recognition hypotheses in order to better predict erroneous words. Finally, in order to show that results presented on these experiments are portable on current state of the art ASR systems, we also present, results applied on the outputs produced by a Kaldi-based TDNN/HMM ASR system [11, 12].

The paper is organized as follows. Section 2 presents our ASR error detection system based on a neural architecture. This system is designed to be used with word embeddings as part of the input features: different types of word embeddings are used and each one is examined alone on the ASR error detection task. Section 3 recalls the performance of the simple and combined linguistic embeddings and their comparison to a state of the art approach. The description of the approach we used to build the acoustic embeddings, and the experimental results that concern the evaluation of acoustic embeddings, as well as the impact of their combination with prosodic features, are reported in section 4. Then, section 5 presents the study of modeling recognition errors at the sentence level. Finally, section 5.5 presents the application of the proposed approach on the outputs produced by a Kaldi-based TDNN/HMM ASR system ,just before the conclusion.

2. ASR error detection system

The proposed error detection system has to attribute the label *correct* or *error* to each word in the ASR transcript. Each decision is based on a set of heterogeneous features. In our approach, this classification is performed by analyzing each recognized word within its context. The context window size used in this study is two on each side of the current word.

This system is based on a feed-forward neural network and it is designed to be fed by different kinds of features, including word embeddings.

2.1. Architecture

This ASR error detection system is based on a multi-stream strategy to train the network, named multilayer perceptron multi-stream (MLP-MS). The MLP-MS architecture is used in order to better integrate the contextual information from neighboring words. This architecture is inspired by [13] where words and semantic features are integrated for topic identification in telephone conversations. The training of the MLP-MS is based on a pre-training of the hidden layers separately followed by a fine tuning of the whole network. The proposed architecture, depicted in Figure 1, is detailed as follows: three feature

vectors are used as input to the network – feature vectors are described in the next section. These vectors are respectively the feature vector representing the two left words (L), the feature vector representing the current word (W) and the feature vector for the two right words (R). Each feature vector is used separately in order to train a multilayer perceptron (MLP) with a single hidden layer. Formally, the architecture is described by the following equations:

$$H_{1,X} = f(P_{1,X} \times X + b_{1,X}), \tag{1}$$

where X represents one of the three feature vectors (L,W and R), $P_{1,X}$ is the weight matrix and $b_{1,X}$ is the bias vector.

The resulting vectors $H_{1,L}$, $H_{1,W}$ and $H_{1,R}$ are concatenated to form the first hidden layer H_1. The H_1 vector is presented as the input of the second *MLP-MS* hidden layer H_2 computed according to the equation:

$$H_2 = g(P_2 \times H_1 + b_2), \tag{2}$$

Finally, the output layer is a vector O_k of k=2 nodes corresponding to the 2 labels *correct* and *error*:

$$O_k = q(P_O \times H_2 + b_O). \tag{3}$$

Based on our previous experiments, the functions f and g are respectively rectified linear units ($ReLU$) and hyperbolic tangent ($tanh$) activation functions, and q is the $Softmax$ function.

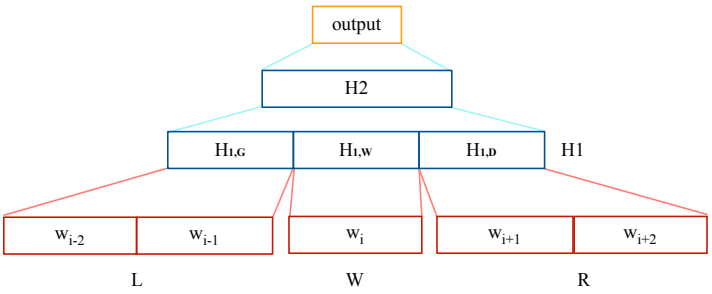

Figure 1: MLP-MS architecture for ASR error detection task.

2.2. Feature vectors

In this section, we describe the features collected for each word and how they are extracted. Some of these features are inspired by the ones presented in [3]. The word feature vector is the concatenation of the following features:

- **ASR confidence scores**: confidence scores are the posterior probabilities generated from the ASR system, computed over confusion networks [14]. Confusion networks are computed from word-lattices and provide better word posteriors than word-lattices since they take into account the different instances of the same hypothesis word that are in concurrence in the lattice, because of different factors (different language model histories, different variants of pronunciation, different temporal frontiers...). By summing all the word posteriors of the different instances of the same word in a confusion bin, a better confidence score for this word is obtained [15, 16]. Size and density of word-lattices used to build the confusion networks were optimized in order to get the lowest WER on the developpment data of the ETAPE corpus. More, during the official participation of the LIUM ASR to the campaign ETAPE, only words with a confidence score greater than 0.3 were kept. In the experiments described in this paper, all the recognized words were kept.

- **Lexical features**: lexical features are derived from the word hypothesis output from the ASR system. They include the word length that represents the number of letters in the word, and three binary features indicating if the 3-grams containing the current word have been seen in the training corpus of the ASR language model.

- **Syntactic features**: we obtain syntactic features by automatically assigning part-of-speech tags (POS tags), dependency labels – such label is a grammatical relation held between a governor (head) and a dependent –, and word governors, which are extracted from the word hypothesis output by using the MACAON NLP Tool chain[1] [17] to process the ASR outputs.

- **Word**: The orthographic representation of a **word** is used in CRF approaches as for instance in [3]. Using our neural approach we can handle different word embeddings, which permits us to take advantage of the generalizations extracted during the construction of the continuous vectors.

2.3. Description of experimental data

Experimental data for ASR error detection is based on the entire official ETAPE corpus [18], composed from audio recordings of French broadcast news shows, with manual transcriptions (reference). This corpus is enriched with automatic transcriptions generated by the LIUM ASR system, which is a multi-pass system based on the CMU Sphinx decoder, using GMM/HMM acoustic models. This ASR system won the ETAPE evaluation campaign in 2012. A detailed description is presented in [19]. Notice that, as written above, in order to show that results presented on these experiments are portable on current state of the art ASR systems, we also present, in section 5.5 final results applied on the outputs produced by a Kaldi-based TDNN/HMM ASR system. Moreover,

[1] http://macaon.lif.univ-mrs.fr

we also published a study showing that our approach can be used to detect errors within ASR outputs produced by an unknown ASR system, including a system based on DNN/HMM acoustic models [20].

The automatic transcriptions have been aligned with reference transcriptions using the *sclite*[2] tool. From this alignment, each word in the corpora has been labeled as correct (C) or error (E). The description of the experimental data, in terms of size, word error rate (WER) as well as percentage of substitution (Sub), deletion (Del) and insertion (Ins), is reported in Table 1. The repartition of the different shows into train/dev/test sub-corpora has been made according to the source of shows (radio and title of the show), in order to balance this in the sub-corpora. The WER information was not known when this distribution has been done. We assume that some speakers or some topics occurring in some shows in the training data were harder to decode than the ones in the Test data: it is probably why the WER is higher on the training data in comparison to the Wer on Test or the Dev data.

Name	**#words ref**	**#words hyp**	**WER**	**Sub**	**Del**	**Ins**
Train	349K	316K	25.3	10.3	12.0	3.1
Dev	54K	50K	24.6	10.3	11.0	3.3
Test	58K	53K	21.9	8.3	10.9	2.7

Table 1: Description of the experimental corpus in terms of size, word error rate (WER) as well as percentage of substitution (Sub), deletion (Del) and insertion (Ins).

3. Linguistic word embeddings

Different approaches have been proposed to build linguistic word embeddings through neural networks. These approaches can differ in the type of architecture and the data used to train the model. Hence, they can capture different types of information: semantic, syntactic, *etc.*

In our previous studies [8, 21], we evaluated different kinds of word embeddings, including:

- **Skip-gram**: This architecture is proposed by [22], and trained using the negative-sampling procedure. The target word w_i is at the input layer, and the context words C are at the output layer. It consists of predicting the contextual words C given the current word w_i.

  The skip-gram model with negative sampling seeks to represent each word w_i and each context C as d-dimensional vectors (V_{w_i},V_C) in order to have similar vector representations for similar words. This is done by maximizing the dot product $V_{w_i}.V_C$ associated with the good word-context pairs

[2] http://www.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm

that occur in the document D and minimizing it for negative examples, that do not necessarily exist in D. These negative examples are created by stochastically corrupting the pairs (w_i, C), thus the name *negative sampling*.

Also, the context is not limited to the immediate context, and training instances can be created by skipping a constant number of words in its context, for instance, $w_{i_{-3}}$, $w_{i_{-4}}$, $w_{i_{+3}}$, $w_{i_{+4}}$, hence the name *skip-gram*.

- **GloVe**: This approach is introduced by [23], and relies on constructing a global co-occurrence matrix X of words, by processing the corpus using a sliding context window. Here, each element X_{ij} represents the number of times the word j appears in the context of word i.

  The model is based on the global co-occurrence matrix X instead of the actual corpus, thus the name GloVe, for Global Vectors.

  This model seeks to build vectors V_i and V_j that retain some useful information about how every pair of words i and j co-occur, such as:

$$V_i^T V_j + b_i + b_j = logX_{ij} \tag{4}$$

  where b_i and b_j are the bias terms associated with words i and j, respectively. This is accomplished by minimizing a cost function J, which evaluates the sum of all squared errors:

$$J = \sum \sum f(X_{ij})(V_i^T V_j + b_i + b_j - logX_{ij})^2 \tag{5}$$

  where f is a weighting function which is used to prevent learning only from very common word pairs. The authors define f as follows [23]:

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{X_{max}})^\alpha & \text{if } X_{ij} < X_{max} \\ 1 & \text{otherwise} \end{cases}$$

- **w2vf-deps**: Levy *et al.* [24] proposed an extension of word2vec, called word2vecf and denoted **w2vf-deps**, which allows to replace linear bag-of-words contexts with arbitrary features. This model is a generalization of the skip-gram model with negative sampling introduced by [22], and it requires labeled data for training. As in [24], we derive contexts from dependency trees: a word is used to predict its governor and dependents, jointly with their dependency labels. This effectively allows for variable window size.

Word embeddings evaluations were carried out on ASR error detection, natural language processing (NLP), analogical and similarity tasks. In addition, we compared three approaches to combine them via an auto-encoder, a principal component analysis (PCA), and a simple concatenation. We revealed that the combination of word embeddings through an auto-encoder yields the best

results compared to the other combination approaches (PCA and simple concatenation).

Based on the results of these studies, we propose to use the best linguistic word embeddings (w2vf-deps, skip-gram and GloVe) retained from the evaluation task [21] and to combine them with an auto-encoder as in [8]. The resulting combined linguistic word embedding is called *Comb_Emb* further in the paper. A detailed description of the combination approaches is presented in [21].

Performance of simple and combined linguistic embeddings is reported in the next sub-section.

3.1. Experimental setup

The linguistic word embeddings were computed from a large textual corpus in French, composed of about 2 billions of words. This corpus was built from articles of the French newspaper "Le Monde", the French Gigaword corpus, articles provided by Google News, and manual transcriptions of about 400 hours of French broadcast news. It contains dependency parses used to train *w2vf-deps* embeddings, while the unlabeled version is used to train *skip-gram* and *GloVe*.

3.2. Experimental results

This section reports the experimental results made on the data set using the ASR error detection system *MLP-MS*. These results concern the evaluation of simple and combined linguistic word embeddings impact when adding them to the set of features presented in Section 2.2.

The performance of the proposed approach is compared with a state-of-the-art system based on CRF [3] provided by the *Wapiti* tagger[3] [25] and applied to the set of features described in section 2.2.

The ASR error detection systems (MLP-MS and CRF) are trained on the training corpus (Train) and are applied on the test (Test) set. The development set (Dev) was used to tune all the parameters: the learning rate, the batch size and the hidden layers sizes of MLP-MS, and the features template of CRF, that describes which features are used in training and testing.

The performance is evaluated by using recall (R), precision (P) and F-measure (F) for the misrecognized word prediction and global Classification Error Rate (CER). CER is defined as the ratio of the number of misclassifications over the number of recognized words. Then, the significance of our results is measured using the 95% confidence interval. Finally, based on confidence interval evaluation, the significant relative improvements are underlined in next tables. To check the statistical significance, we used the Student's t-test since we observed that the CER follows a normal law by subsampling some parts of the development data. Note that, in our study we are interested to the detection of insertion and substitution errors.

[3] http://wapiti.limsi.fr

We present in Table 2, the results obtained by using the simple and the combined embeddings in addition to the other features. We also present the results of a baseline system (Base) that assign to all recognized words with the label 'Correct', and the results of a simple error detection that only applies a threshold on the ASR confidence scores to label 'Correct' or 'Error'. In terms of global classification error rate, the proposed neural approach outperforms the CRF, especially by using the combined embeddings *Comb_Emb*, it achieves 6.02% and 5.72% of CER relative reduction on Dev and Test. These results confirm the ones obtained in our previous studies [8, 21]. We refer in the remainder of the paper to the set of features presented in section 2.2 (ASR confidence scores, lexical features, syntactic features) and the combined embedding *Comb_Emb* to represent the word as the baseline features, named **B-Feat**.

Corp.	App.	Word Representation	Label *Error* P	R	F	Global CER
Dev	Base	always correct	-	-	-	14.69
	Simple	ASR conf. score	66.78	49.13	56.61	11.08
	Neural	w2vf-deps	73.01	50.17	59.47	10.06
		Skip-gram	**75.06**	45.58	56.72	10.24
		GloVe	73.81	46.98	57.41	10.26
		Comb_Emb	70.50	**57.56**	**63.38**	**9.79**
	CRF	discrete	68.11	55.37	61.08	10.38
Test	Base	always correct	-	-	-	12.04
	Simple	ASR conf. score	65.43	47.43	55.00	9.30
	Neural	w2vf-deps	71.90	50.98	59.66	8.26
		Skip-gram	**74.45**	46.75	57.44	8.30
		GloVe	72.16	46.97	56.90	8.53
		Comb_Emb	69.66	**57.89**	**63.23**	**8.07**
	CRF	discrete	67.69	54.74	60.53	8.56

Table 2: Comparison of the use of different types of word embeddings in MLP-MS error detection system on Dev and Test corpora.

4. Acoustic word embeddings

Until now, we experimented with several information sources: syntactic, lexical, ASR-based features. However, we did not yet investigate the use of acoustic information. One issue about representing such information is the need of a fixed size representation to be injected in the same way as used to inject the other information sources at the word level in our neural architecture. Acoustic word embeddings are an interesting solution to get a fixed length vector representation. Acoustic word embeddings were successfully used in a query-by-example search system [26, 27] and in a segmental ASR lattice re-scoring system [28]. Thus, we propose to evaluate their performance in the framework of the ASR error detection.

4.1. Building signal and acoustic word embeddings

The approach we used to build acoustic word embeddings is inspired by the study proposed in [28]. Acoustic word embeddings are trained through a deep neural architecture, depicted in Figure 2, which relies on a convolutional neural network (CNN) classifier over words and on a deep neural network (DNN) trained by using a triplet ranking loss [28, 29, 30]. This architecture was proposed in [28] with the purpose to use the scores derived from the CNN word classifier for lattice rescoring. The two architectures are trained using two different inputs: speech signal and orthographic representation of the word.

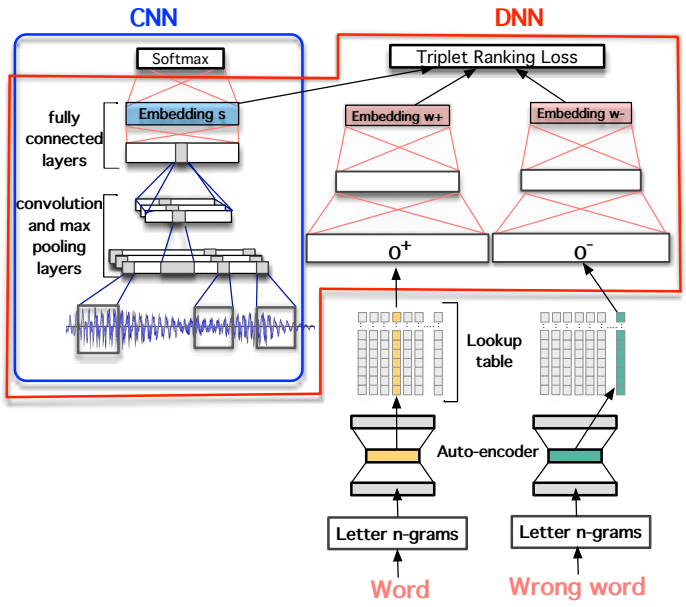

Figure 2: Deep architecture used to train acoustic word embeddings. In the DNN part, all the weights and biases are shared by the two DNNs.

The CNN is trained to predict a word given an acoustic sequence of T frames (log-filterbanks computed every 10ms over a 25ms window yielding a 23-dimensional vector) as input. It is composed of several convolution and pooling layers, followed by several number of fully connected layers that feed into the final $Softmax$ layer. The final fully connected layer just below the $Softmax$ one is denoted $\mathbf{s}$ for signal word embedding (it was denoted $\mathbf{e}$ in [28]). It contains a compact representation of the audio signal. This representation tends to preserve acoustic similarity between words, such that words are close in this space if they sound alike.

The idea behind using the second architecture is to be able to build a unique acoustic word embedding $\mathbf{a}$ from an orthographic word representation, especially in order to get an acoustic word embedding for words not already observed in

an audio speech signal. Above all, a such acoustic word embedding derived from an orthographic representation can be perceived as a unique canonical acoustic representation for a word, since different pronunciations imply different signal word embeddings $\mathbf{s}$.

Like in [28], an orthographic word representation consists on a bag of n-grams ($n \leq 3$) of letters, composed of 10222 trigrams, bigrams, and unigrams of letters, including special symbols $\lceil$ and $\rfloor$ to specify the start and the end of a word. Since we do not access to the same amount of training data as Google in [28], we chose to use an auto-encoder in order to reduce the size of this bag of n-grams vector to d-dimensions to reduce the number of parameters to train. To check the performance of the resulting orthographic representation, a neural network is trained to predict a word given this compressed orthographic representation. It reaches 99.99% of accuracy on the training set composed of the $52k$ words of the vocabulary, showing the richness of this representation.

Similar to [28], a DNN was trained by using the triplet ranking loss [28, 29, 30] in order to project the orthographic word representation into the signal embeddings $\mathbf{s}$ space obtained from the CNN architecture, which is trained independently. It takes as input a word orthographic representation and outputs an embedding vector of the same size as $\mathbf{s}$.

During the training process, this model takes as inputs the signal word embedding $\mathbf{s}$ selected randomly from the training set, the orthographic representation of the matching word $\mathbf{o}^+$, and the orthographic representation of a randomly selected word $\mathbf{o}^-$ different from the first word. These two orthographic representations supply shared parameters in the DNN.

We call $t = (\mathbf{s}, \mathbf{a}^+, \mathbf{a}^-)$ a triplet, where $\mathbf{s}$ is the signal embedding, $\mathbf{a}^+$ is the embedding obtained through the DNN for the matching word, while $\mathbf{a}^-$ is the embedding obtained for the wrong word.

The triplet ranking loss is defined as:

$$Loss = \max(0, m - Sim_{dot}(s, a^+) + Sim_{dot}(s, a^-)) \tag{6}$$

where $Sim_{dot}(x, y)$ is the dot product function used to compute the similarity between two vectors x and y, and m is a margin parameter that regularizes the margin between the two pairs of similarity $Sim_{dot}(\mathbf{s}, \mathbf{a}^+)$ and $Sim_{dot}(\mathbf{s}, \mathbf{a}^-)$. This loss is weighted according to the rank in the CNN output of the word matching the audio signal.

The resulting trained model can then be used to build an acoustic word embedding ($\mathbf{a}^+$) from any word, as long as one can extract an orthographic representation from it.

Performance of signal embeddings $\mathbf{s}$ and word embeddings $\mathbf{a}$ applied to ASR error detection is summarized in the next section.

4.2. Experimental setup

4.2.1. Data

The training set for the CNN consists of 488 hours of French Broadcast News with manual transcriptions. This dataset is composed of data coming

from the ESTER1 [31], ESTER2 [32] and EPAC [33] corpora. It contains $52k$ unique words that are seen at least twice each in the corpus. All of them corresponds to a total of 5.75 millions occurrences. In French language, many words have the same pronunciation without sharing the same spelling, and they can have different meanings; *e.g.* the sound [so] corresponds to four homophones: *sot* (fool), *saut* (jump), *sceau* (seal) and *seau* (bucket), and twice more by taking into account their plural forms that have the same pronunciation: *sots*, *sauts*, *sceaux*, and *seaux*. When a CNN is trained to predict a word given an acoustic sequence, these frequent homophones can introduce a bias to evaluate the recognition error. To avoid this, we merged all the homophones existing among the $52k$ unique words of the training corpus. As a result, we obtained a new reduced dictionary containing $45k$ words and classes of homophones.

As written before, acoustic features provided to the CNN are log-filterbanks, computed every 10ms over a 25ms window yielding a 23-dimensional vector for each frame. A forced alignment between manual transcriptions and speech signal was performed on the training set in order to detect word boundaries. The statistics computed from this alignment reveal that 99% of words are shorter than 1 second. Hence we decided to represent each word by 100 frames, thus, by a vector of 2300 dimensions. When words are shorter they are padded with zero equally on both ends, while longer words are cut equally on both ends.

4.2.2. Architectures

To build the signal and acoustic word embeddings, the CNN and DNN deep architectures are trained on 90% of the training set and the remaining 10% are used for validation.

- CNN: This architecture predicts a word given by a sequence of 100 frames. It contains two convolution and max-pooling layers followed by two fully-connected layers, which feed into the final $softmax$ layer over $45k$ words and classes of homophones. The convolution layers have respectively 15 and 10 filters over 8 frames.

  The max pooling layers perform over 4 units. The two fully connected layers are composed with 500 and 100 units respectively. The hyperbolic tangent ($Tanh$) function is used as an activation function for all the layers. This model achieves 61.51% of accuracy (*i.e.* word recognition precision) on the validation set. It is used to extract the signal word embedding $\mathbf{s}$.

- DNN: This architecture has to map the word orthographic representation into the signal embeddings space obtained by the CNN model. It is composed of two fully connected $Tanh$ layers of 300 and 100 units each. When replacing in the CNN the signal word embedding $\mathbf{s}$ (*Cf.* Figure 2) by the acoustic word embedding $\mathbf{a}^+$ computed by the DNN model from the orthographic embedding $\mathbf{o}^+$, the accuracy on the validation set becomes 50.15%. Since acoustic word embeddings are expanded in the acoustic space from the orthographic notation, it is by nature more difficult to expand orthographic transcription into acoustic space than to expand the audio signal

to this space. However, this result shows that acoustic word embeddings are able to capture relevant acoustic information in a continuous space to model a word.

4.3. Experimental results

With the purpose to evaluate the performance of acoustic word embeddings on ASR error detection, they were used as additional features into the MLP-MS system.

The signal and acoustic word embeddings (respectively $\mathbf{s}$ and $\mathbf{a}^+$) of the current word were processed in a specific stream as shown in Figure 3. The additional hidden layer $H3$ takes as input the concatenation of the two layers $H2$-1 and $H2$-2-AC .

Three settings are compared corresponding to different types of features used in MLP-MS: baseline features (**B-Feat**) alone, then adding the signal embeddings $\mathbf{s}$, and finally adding the acoustic word embeddings $\mathbf{a}^+$. Experimental results reported in Table 3 show the usefulness of signal embeddings $\mathbf{s}$, that are able to characterize some suspicious acoustic segments. It yields a relative improvement in terms of CER reduction compared to the results of the neural baseline system (**B-Feat**). An additional slight relative improvement is observed by adding the acoustic embedding $\mathbf{a}^+$.

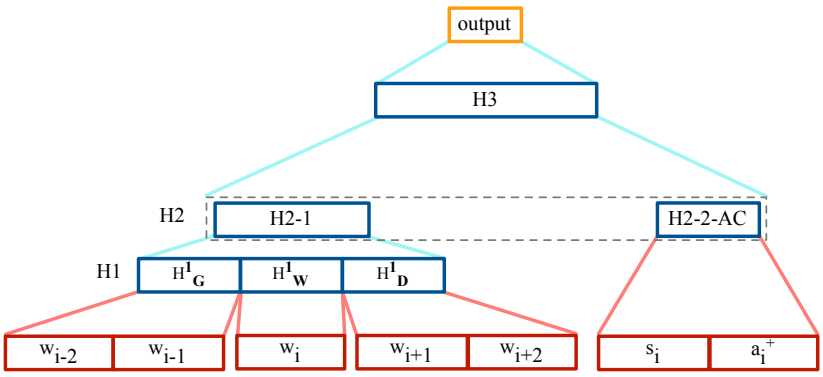

Figure 3: MLP-MS architecture for ASR error detection task, that integrates signal and acoustic word embeddings in addition to the baseline features.

To improve further these results, we propose to add other informations extracted from the audio signal, which are prosodic features.

4.4. Performance of prosodic features

Our use of prosodic features is motivated by state of the art studies [34, 35, 36]. Those studies reveal that misrecognized words have extreme prosodic values [36], and prosodic features are useful for misrecognized utterance detection [35], whereas their combination with syntactic features is helpful to localize

Corp.	Word Representation	Label *Error*			Global
		P	**R**	**F**	**CER**
Dev	B-Feat	70.50	57.56	63.38	9.79
	B-Feat $\oplus$ **s**	**71.98**	57.63	64.01	9.54
	B-Feat $\oplus$ **s** $\oplus$ $\mathbf{a}^+$	71.70	**58.25**	**64.28**	**9.53**
Test	B-Feat	69.66	57.89	63.23	8.07
	B-feat $\oplus$ **s**	69.64	**59.13**	63.95	7.99
	B-feat $\oplus$ **s** $\oplus$ $\mathbf{a}^+$	**70.09**	58.92	**64.02**	**7.94**

Table 3: Performance of signal and acoustic embeddings for ASR error detection task on Dev and Test corpora.

misrecognized words in a speaker turn [34]. We show in our previous study [8], the usefulness of prosodic features when they are combined with syntactic ones. We aim in this study to evaluate their impact when they are combined with the acoustic embeddings.

4.4.1. Prosodic features extraction

Automatic forced alignment is carried out using the LIMSI ASR system [37]. As well as extra-lexical events such as silences, breath and hesitations, words and corresponding phone segments are located (time-stamped) in the speech signal. Forced alignments are carried out twice: (i) using the manual / reference transcription and (ii) using the hypothesis of the LIUM ASR system [19]. As a result, we get among others word and phone segment durations that are relevant to prosody, rhythm and speech rate. The system's pronunciation dictionary includes major pronunciation variants (*e.g.* optional schwas in word-internal and word-final positions, optional liaison consonants, *etc.*) and the best matching pronunciation variant is selected during alignments. The audio signal is processed using `Praat` [38] (standard analysis parameters) to measure fundamental frequency (f_0) every 5 milliseconds, depending on the method used in [39]. f_0 is the acoustic parameter corresponding to the perceived voice frequency. Speech f_0 variations are highly relevant to prosody, in particular if they can be linked to phone segments and their embedding words and phrases [40]. The phone timecodes delivered during forced alignment are used to retrieve the corresponding f_0 values both for reference and hypothesis transcriptions, on three points for each vocalic segment. Since the properties of central segments show more acoustic stability, we use measures taken at the middle of vocalic segments in order to calculate f_0 values at the word level: (i) f_0 mean of the word (in Hertz) and (ii) delta between f_0 of the first vocalic segment of the word and f_0 of the last vocalic segment of the word (in both Hertz and semitones).

4.4.2. Experimental results

Experimental results reported in Table 4 show the usefulness of prosodic features, that yield a relative improvement in terms of CER reduction compared to the results in Tables 3 and 2 for neural and CRF systems.

Corpus	Approach	**Word Representation**	**Label *Error***			**Global CER**
			P	**R**	**F**	
Dev	CRF	discret	68.11	55.37	61.08	10.38
	CRF $\oplus$ *pros*		69.81	55.07	61.57	10.12
	Neural	B-Feat (baseline)	70.50	57.56	63.38	9.79
		B-Feat $\oplus$ pros	69.26	**63.46**	**66.23**	9.52
		B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$	71.70	58.25	64.28	9.53
		B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$ $\oplus$ pros	71.70	59.96	65.31	**9.38**
Test	CRF	discret	67.69	54.74	60.53	8.56
	CRF $\oplus$ pros		68.61	54.14	60.52	8.46
	Neural	B-Feat (baseline)	69.66	57.89	63.23	8.07
		B-Feat $\oplus$ pros	68.12	**63.16**	**65.55**	7.96
		B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$	70.09	58.92	64.02	7.94
		B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$ $\oplus$ pros	**70.27**	61.32	65.49	**7.75**

Table 4: Performance of the combination of prosodic features and acoustic embeddings in addition to the baseline features on Dev and Test corpora.

For neural systems, we observe that the integration of prosodic features (B-Feat $\oplus$ pros) or acoustic embeddings (B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$) obtains similar results in terms of CER. The combination of prosodic features, linguistic and acoustic embeddings in addition to the other features (B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^+$ $\oplus$ pros) yields the most interesting results, and achieves an interesting relative improvement in comparison to the baseline (B-Feat). These results show the complementarity of acoustic and prosodic information and this best system is named *MLP-MS-AC* further in the paper.

We observe as well, that the CER improvements are not fully consistent with changes in F-measure. For reminder, the CER score is evaluated on both *correct* and *error* labels whereas the F-measure score focuses only on the error label. Hence, the error variations seems not to be proportional between the two labels.

For reminder, the CER score is evaluated on both *correct* and *error* labels whereas the F-measure score focuses only on the error label. Hence, the error variations seems not to be proportional between the two labels.

Furthermore, we investigated the choice of extending the size of left- and right- context windows. We experimented the use of a context window of three words (three at left, and three at right). In a such way, performances decrease to a CER of 7.87 (compared to 7.75 for the baseline). This result confirms what we observed in preliminary experiments carried on to choose the optimal size of the context window. A size of two words was also optimal when using only single linguistic word embeddings and part of speech information.

Last, we also investigated the use of acoustic embeddings for context words, leading again to a loss in CER: 7.82 on test set.

Note that, increasing the context size from 2 to 3 words, leads to increase

the context window size from 5 to 7. Assuming, that the word features vector size is equal to 446d without taking into account the acoustic embeddings (200d), the input of the network is equal to 3122d (446*7). The augmentation of context size may require the use of more data to train the network and obtain better performance when adding such additional information like the acoustic embeddings in the context words.

These two complementary experiments confirm us in our two parameter choices.

4.5. Average span analysis of the ASR error detection system outputs

In this section, we are interested in the analysis of the outputs generated by our best system, *MLP-MS-AC*, in order to perceive the errors that are hard to detect. We do not use the Test corpus for this part of our work in order to avoid to introduce some biases in our future experiments. This analysis is based on the average error segment size (average span), since we know that our system takes only local decisions and is not designed to perform optimally sequence predictions.

Results summarized in Table 5 present the average span and the standard deviation for the ground truth, the predictions (classifier outputs) and the correct predictions for *MLP-MS-AC* and CRF classifiers. The average span of the correct predictions is defined as the average error segment of the contiguous errors correctly detected.

Corpus	**Approach**		**Average span**	**Standard deviation**
Train		Ground truth	3.03	1.72
Dev			3.24	2.15
Test			3.10	
Dev	CRF⊕*pros*	predictions	3.28	1.77
		correct predictions	2.88	1.34
	MLP-MS-AC	predictions	2.82	1.28
		correct predictions	2.66	1.05
Test	CRF⊕*pros*	predictions	2.92	1.21
		correct predictions	2.46	0.98
	MLP-MS-AC	predictions	2.79	1.28
		correct predictions	2.62	0.75

Table 5: The average span and the standard deviation for the ground truth, the predictions, and the correct predictions for CRF and *MLP-MS-AC* systems.

We observe that the average span of CRF outputs is nearly the same as the ground truth. However, for the *MLP-MS-AC* system the average span of the predictions is smaller by 12.9% relative compared to the ground truth, with a smaller standard deviation by 40.5%. Regarding the correct predictions, the average spans for correct predictions of both systems are smaller than the ground truth, even if the one of CRF is closer to the ground truth.

The gap related to the error segment size between the ground truth, the predictions and the correct predictions is due to the architecture of the proposed ASR error detection system ($MLP\text{-}MS\text{-}AC$). This one takes only local decisions and is not currently designed to perform optimally sequence predictions while CRF seems to be able to better capture such global information.

The results of this analysis provided us useful information in order to improve the performance of the proposed ASR error detection system. For this purpose, we propose to explore the integration of global information, at the sentence level, and evaluate their impact by using the same neural architecture.

5. Global decision: sentence embeddings

In this section, we focus on the integration of global information to enrich our ASR error detection system, through the use of sentence embeddings (Sent-Emb). These representations have been successfully used in sentence classification and sentiment analysis tasks [41, 42, 43]. Sentence embeddings can be built in a general context by using the tool Doc2vec [41], or they can be adapted to a specific task like for the sentiment analysis task as in [44].

For the error detection task, we propose to build sentence embeddings that carry information about the confidence of the recognition hypothesis at the sentence level: whether the sentence is almost correct or highly erroneous. Then, we compare the performance of the proposed sentence embeddings to the DBOW (Distributed bag of words) embeddings provided by Doc2vec [41].

5.1. DBOW embeddings

The DBOW approach consists on predicting the words randomly sampled from the paragraph in the output: *i.e.* at each iteration of stochastic gradient descent, we sample a text window, then sample a random word from the text window and form a classification task given the paragraph vector [41]. This model is also similar to the Skip-gram model in word vectors [45]. In our experiments, this model is trained on the ETAPE corpus to build 100-dimensional embeddings, named Emb_{DBOW}, for each automatic transcription (utterance).

5.2. Task-specific embeddings

The sentence embeddings Emb_{DBOW} carry semantic information held in the automatic transcriptions, but probably do not carry specific information for ASR error detection task. Thus, we propose to build specific sentence embeddings for the error detection task.

For this reason, we propose to use the embeddings extracted from a convolutional neural network, named Emb_{CNN}, trained to predict whether an automatic transcription (utterance) is slightly erroneous (SE) or very erroneous (VE). The resulting sentence embeddings may capture information about the error. To build those embeddings we need to use a labeled corpus: each utterance in the ETAPE corpus was tagged to "slightly erroneous" or "very erroneous". In this study, we arbitrarily consider a sentence as very erroneous if 20% of the

words that it compose are incorrect. Indeed, as our goal is to build embeddings with information about errors in the utterances, the choice of the 20% seems reasonable to predict the confidence of the utterances. The utterances with less than 20% of incorrect words are then considered as slightly erroneous (including fully correct utterances). Table 6 presents the description of the data used to train the convolutional neural network.

Corpus	**#Ref. Utt.**	**# Hyp. Utt.**	**#SE Utt.**	**#VE Utt.**
Train	22K	21.3K	13.3K	8.3K
Dev	3.7K	3.5K	2.2k	1.3k
Test	3.6K	3.5K	2.3K	1.1K

Table 6: Description of the data used to build the Emb_{CNN} embeddings in terms of number of reference and hypotheses utterances and the number of slightly erroneous (SE) and very erroneous (VE) utterances.

The CNN takes as input an utterance represented by a vector of features and has as output two labels "slightly erroneous" or "very erroneous". The CNN architecture is composed of two convolution and max pooling layers followed by two fully connected layers. From the hidden layer just before the Softmax layer we extracted the 100-dimensional sentence embeddings (Emb_{CNN}) for each utterance. Note that, the CNN classifier achieves 13.5% of classification error rate on Test corpus transcriptions.

The utterance feature vector corresponds to the concatenation of the feature vectors of the words composing the utterance. The word feature vector is composed of those used in $MLP\text{-}MS\text{-}AC$ (B-Feat $\oplus$ $\mathbf{s}$ $\oplus$ $\mathbf{a}^{+}$ $\oplus$ pros).

The size of the utterances is set to 50 words, since 98.37% of them have a size that varies between 1 and 50 words. When the utterances are shorter they are padded with zero equally on both ends, while longer utterances are cut equally on both ends.

5.3. Experimental results

This section summarizes the comparison results between both sentence embeddings: Emb_{DBOW} and Emb_{CNN}. The performance reached by using those embeddings is compared to the ones got by the $MLP\text{-}MS\text{-}AC$. The sentence embedding Emb_{DBOW} or Emb_{CNN} was processed by a specific stream as shown in the fig. 4. In this case the last hidden layer $H3$ takes as input the concatenation of the three hidden layers $H2$-1, $H2$-2-AC and $Sent\text{-}EmbH$.

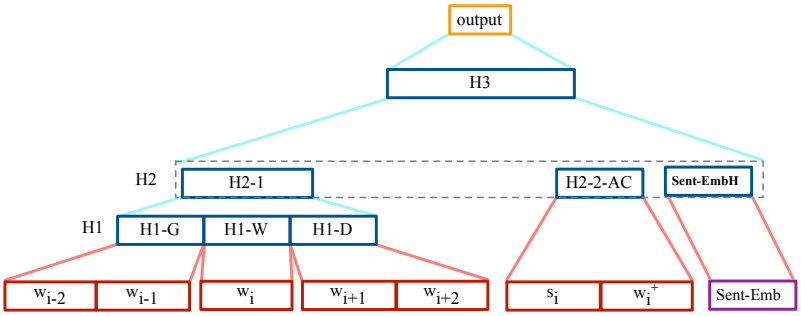

Figure 4: MLP-MS architecture for ASR error detection task, that integrates acoustic and sentence embeddings in addition to the baseline features.

Experimental results, summarized in table 7, show that the integration of sentence embeddings was helpful and yields to some relative improvements in comparison to the results of *MLP-MS-AC*, especially when using the Emb_{CNN} embedding, which is better than the Emb_{DBOW} embedding. The Emb_{CNN} embedding yields to 1.27% and 0.77% of CER relative reduction in comparison to *MLP-MS-AC*, respectively on Dev and Test. This system is named *MLP-MS-AC-*Emb_{CNN} further in the paper. From these results we can reveal that the Emb_{CNN} have captured information about the error useful for our targeted task.

Comparing to the results obtained by the best simple embeddings *w2vf-deps*, our entire neural approach using the combined linguistic embeddings, acoustic and sentence embeddings in addition to the prosodic and syntactic features achieves statistically significant relative improvements, by respectively 7.95% and 6.9% in terms of CER reduction on Dev and Test.

Corpus	**Sentence Embed.**	**Label *Error***			**Global**
		P	**R**	**F**	**CER**
Dev	- (baseline)	71.70	59.96	65.31	9.38
	Emb_{DBOW}	**72.74**	58.24	64.69	9.36
	Emb_{CNN}	72.49	59.80	65.53	**9.26**
Test	- (baseline)	70.27	61.32	65.49	7.75
	Emb_{DBOW}	**72.48**	57.32	64.01	7.72
	Emb_{CNN}	**72.49**	57.81	64.32	**7.69**

Table 7: Performance of sentence *embeddings* Emb_{DBOW} and Emb_{CNN} in comparison to the results of MLP-MS-AC system on Dev and Test corpora

We revealed that the addition of the information extracted at the sentence level improves the performance of our system. In order to confirm the hypothesis discussed in section 4.5 on improving the performance of the error detection through a better global modeling, that would indirectly better models error

span, we report in this section the results of the average span analysis performed on the *MLP-MS-AC-Emb$_{CNN}$* system outputs. Table 8 presents the average spans and the standard deviations for the ground truth, the predictions and the correct predictions for CRF, *MLP-MS-AC* and *MLP-MS-AC-Emb$_{CNN}$* systems. The results show that sentence embeddings have captured information about the error propagation: indeed, the addition of these embeddings has improved the average span compared to the *MLP-MS-AC* system. Thus, the *MLP-MS-AC-Emb$_{CNN}$* system obtains results very close to the CRF, in terms of average span for both predictions and correct prediction.

Approach		**Average span**	**Standard deviation**
	Ground truth	3.24	2.15
CRF⊕*pros (baseline)*	predictions	3.28	1.77
	correct predictions	2.88	1.34
MLP-MS-AC	predictions	2.82	1.28
	correct predictions	2.66	1.05
MLP-MS-AC-Emb$_{CNN}$	predictions	3.15	1.70
	correct predictions	2.84	1.22

Table 8: The average span and the standard deviation for the ground truth, the predictions, and the correct predictions for CRF, *MLP-MS-AC* and *MLP-MS-AC-Emb$_{CNN}$* systems.

5.4. Comparison to bidirectional LSTM system

These last experiments revealed the usefulness of the sentence embeddings integration in our MLP-MS architecture. Since some neural architectures showed recently to be effective to process sequence to sequence tasks [46], it could be interesting to compare the neural approach used until now in our experiments to measure the impact of continuous representations to the use of a bidirectional LSTM architecture. Such an architecture is designed to learn how to integrate relevant long distant information, and was successfully used for the ASR error detection task in [6, 7].

In our experiments, the bidirectional LSTM architecture is composed of two hidden layers of 512 hidden units each, *i.e.* 256 units in each forward and backward sides and a fully connected layer of 128 units. It integrates the same features as the MLP-MS-AC system: ASR confidence scores, lexical features, syntactic features, prosodic features, linguistic embeddings and acoustic embeddings. This system is called BLSTM-AC. For the BLSTM-AC system we did include sentence embeddings that capture global information about the sentence, because we suppose the BLSTM-AC by nature, should be able to capture this kind of information. Results summarized in table 9 show that BLSTM-AC and MLP-MS-AC systems obtain comparable results. Notice that BLSTM-AC system obtains better results on Dev but not on Test corpus. It seems that the BLSTM architecture does not generalize well in these experiments. This

behaviour is not due to the number of parameters to train: there are about 3.5 millions of parameters (weight + biases) to train in the BLSTM while they are 3.6 millions in the MLP-MS. To explain the better generalization observed on MLP-MS, we assume that the topology of the MLP-MS architecture helps during the training process. For instance the use of an optimized neighbor size, or the use of specific streams in relation to the nature of the input features have learned during the training process by the BLSTM while they are known before the training, by nature, in the MLP-MS architecture. Maybe with more training data, the BLSTM can improve its generalization capacity. Moreover, as we can see the addition of sentence embeddings to the BLSTM architecture (BLSTM-AC$_{CNN}$), yields to almost the same results on Dev, but degraded the results on Test, this mainly due to the lack of training data as we mentioned before.

In this paper we focus on word and sentence continuous word representations, and evaluate them for the ASR error detection task through the use of a feedforward neural architecture. These results with the BLSTM architecture, recently proposed for this task, validate our previous experiments, and show that they cannot be questioned in relation to the use of a more sophisticated neural architecture.

Moreover, these results confirm our hypothesis about the integration of global information in MLP-MS system in order to take better local decisions, since the MLP-MS-AC-Emb$_{CNN}$ system achieves better results than the BLSTM-AC system.

Corpus	**System**	**Label *Error***			
		P	**R**	**F**	**CER**
Dev	MLP-MS-AC	71.70	59.96	65.31	9.38
	MLP-MS-AC-Emb$_{CNN}$	72.49	59.80	65.53	**9.26**
	BLSTM-AC	70.49	**63.60**	**66.87**	9.28
	BLSTM-AC$_{CNN}$	**73.56**	57.84	64.76	9.27
Test	MLP-MS-AC	70.27	61.32	65.49	7.75
	MLP-MS-AC-Emb$_{CNN}$	**72.49**	57.81	64.32	**7.69**
	BLSTM-AC	68.90	**63.26**	**65.96**	7.83
	BLSTM-AC$_{CNN}$	71.09	56.82	63.16	7.95

Table 9: Comparison of the proposed MLP-MS architecture to a BLSTM architecture.

5.5. Application to neural ASR outputs

This section reports the results of the application of our approach on the outputs produced by a Kaldi-based TDNN/HMM ASR system as we mentioned in section 2.3. The acoustic models of this ASR system were trained with the same training data as the one previously used. The same vocabulary and the same language models were used. Acoustic models are based on chain model, based on a sub-sampled time-delay neural network (TDNN) [12]. This kind of model is trained with a sequence-level objective function (the log probability of the

correct phone sequence) [47]. It can be viewed as training with the maximum mutual information (MMI) criterion, implemented without lattices, by doing a full forward-backward on a decoding graph derived from a phone n-gram language model and using a three times smaller frame rate at the output of the neural network. Training this model was done by using high-resolution MFCC features (without dimensionality reduction, keeping the 40 cepstra) concatenated with 100-dimensional i-vectors for speaker adaptation, accounting for an input dimension of 140.

For these experiments, we recomputed all the features presented in this paper for each new recognized words, including acoustic word embeddings, sentence embeddings,.. and we retrained all the different systems presented in this paper: CRF, MLP-MS and BLSTM.

The description of the experimental data, in terms of size, word error rate (WER) as well as percentage of substitution (Sub), deletion (Del) and insertion (Ins), is reported in Table 10.

Name	**#words ref**	**#words hyp**	**WER**	**Sub**	**Del**	**Ins**
Train	347k	328k	21.2	6.9	9.9	4.5
Dev	53k	51k	21.6	6.7	10.0	5.0
Test	57k	54k	19.0	5.7	9.2	4.1

Table 10: Description of the neural ASR outputs in terms of size, word error rate (WER) as well as percentage of substitution (Sub), deletion (Del) and insertion (Ins).

To build sentences embeddings for neural ASR outputs we used the same experimental protocol as mentioned in section 5.2. So, we arbitrarily consider a sentence as very erroneous if 20% of the words that it compose are incorrect. Table 11 presents the description of the data used to train the convolution neural network to train the sentence embeddings.

Corpus	**#Ref. Utt.**	**# Hyp. Utt.**	**#SE Utt.**	**#VE Utt.**
Train	22k	21k	16k	5k
Dev	3.7	3.5	2.6k	0.9k
Test	3.6	3.5	2.6k	0.9k

Table 11: Description of the data used to build the Emb_{CNN} embeddings in terms of number of reference and hypotheses utterances and the number of slightly erroneous (SE) and very erroneous (VE) utterances.

Note that, the CNN classifier achieves 16.11% of classification error rate on Test corpus transcriptions.

Table 12 summarizes the different experimental results obtained by applying the different systems presented in this paper: CRF, MLP-MS and BLSTM on the outputs produced by the neural ASR system. For neural systems, both results without and with sentence embeddings Emb_{CNN} are presented. As we can see, these results confirm the ones got on the GMM/HMM based-ASR

system. The MLP-MS system obtains better results than BLSTM and CRF systems. In addition, results obtained with BLSTM match the results obtained in section 5.4, since the BLSTM architecture does not generalize well in these experiments too.

Corpus	**System**	**Label *Error***			
		P	**R**	**F**	**CER**
Dev	MLP-MS-AC	**69.01**	45.15	54.59	8.13
	MLP-MS-AC-Emb$_{CNN}$	68.75	**46.38**	**55.39**	**8.09**
	BLSTM-AC	65.22	36.65	46.93	8.64
	BLSTM-AC-Emb$_{CNN}$	67.50	42.02	51.80	8.15
	CRF	65.60	37.99	48.12	8.87
Test	MLP-MS-AC	**69.12**	49.27	57.53	6.32
	MLP-MS-AC-Emb$_{CNN}$	69.04	**50.04**	**58.03**	**6.29**
	BLSTM-AC	65.31	39.69	49.38	7.08
	BLSTM-AC-Emb$_{CNN}$	64.48	46.36	53.94	6.88
	CRF	66.58	39.44	49.53	6.99

Table 12: Performance of the proposed MLP-MS architecture, BLSTM architecture and CRF on Kaldi outputs.

6. Conclusions and future work

This paper presents a study that focuses on the use of different types of continuous representations applied to the ASR error detection task. An important objective in this task is to locate possible linguistic or/and acoustic incongruities in automatic transcriptions. For this, we focused on the use of different types of embeddings that are able to capture information from different levels: linguistic word embeddings, acoustic word embeddings, and sentence embeddings.

Experiments, that were performed on the French ETAPE corpus, show that the combination of linguistic embeddings, acoustic embeddings, and prosodic features in addition to other more classical features yields very competitive results. Particularly, these new results show the high complementarity of acoustic word embeddings and prosodic information, and show that the proposed task-specific sentence embeddings achieve better results than the general ones proposed by Doc2vec.

Finally, the use, in a simple feed forward neural architecture, of combined linguistic embeddings, prosodic features, acoustic and sentence embeddings as additional sources of evidence, strongly and significantly improves the results in comparison to the use of the best single linguistic word embeddings, while this last approach was yet better than the use of Conditional Random Fields that were the state-of-the-art for this task until very recently. In addition, we show that using a BLSTM architecture does not improve the ASR error detection: the MLP-MS architecture fed by a sliding window of relevant continous representations is particularly effective. Last, we show that results presented

on these experiments are portable on current state of the art ASR systems, by applying the proposed approach on the outputs produced by a Kaldi-based TDNN/HMM ASR.

In a future work, we expect to develop new approaches in order to automatically classify the nature (acoustic, linguistic, phonetic...) of a recognition error if such an error is detected, and we also work on improving the ASR system performances by exploiting and injecting in the loop this good quality detection.

Last, we will explore the use and the adaptation of word and sentence continuous representations in other tasks, *e.g.* spoken language understanding.

Acknowledgement

This work was partially funded by the European Commission through the EUMSSI project, under the contract number 611057, in the framework of the FP7-ICT-2013-10 call. This work was also partially funded by the French National Research Agency (ANR) through the VERA project, under the contract number ANR-12-BS02-006-01.

References

[1] Lafferty, John D. and McCallum, Andrew and Pereira, Fernando C. N., Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 282–289.

[2] C. Parada, M. Dredze, D. Filimonov, F. Jelinek, Contextual Information Improves OOV Detection in Speech, in: Human Language Technologies: Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'10), 2010.

[3] F. Béchet, B. Favre, ASR error segment localization for spoken recovery strategy, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013, 2013, pp. 6837–6841. doi:10.1109/ICASSP.2013.6638986.

[4] T. Yik-Cheung, Y. Lei, J. Zheng, W. Wang, ASR error detection using recurrent neural network language model and complementary ASR, in: Proceedings of Acoustics, Speech and Signal Processing (ICASSP 2014), 2014, pp. 2312–2316.

[5] S. Jalalvand, D. Falavigna, Stacked auto-encoder for ASR error detection and word error rate prediction, in: 16th Annual Conference of the International Speech Communication Association (INTERSPEECH), Dresden, Germany, September 6-10, 2015, pp. 2142–2146.

[6] A. Ogawa, T. Hori, ASR error detection and recognition rate estimation using deep bidirectional recurrent neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference, IEEE, 2015, pp. 4370–4374.

[7] A. Ogawa, T. Hori, Error detection and accuracy estimation in automatic speech recognition using deep bidirectional recurrent neural networks, Speech Communication 89 (2017) 70–83.

[8] S. Ghannay, Y. Estève, N. Camelin, C. Dutrey, F. Santiago, M. Adda-Decker, Combining continous word representation and prosodic features for ASR error prediction, in: 3rd International Conference on Statistical Language and Speech Processing (SLSP 2015), Budapest (Hungary), 2015.

[9] S. Ghannay, Y. Estève, N. Camelin, P. Deleglise, Acoustic word embeddings for ASR error detection, in: Interspeech 2016, San Francisco (CA, USA), 2016.

[10] S. Ghannay, Y. Estève, N. Camelin, P. Deléglise, Evaluation of acoustic word embeddings, in: Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP, 2016, pp. 62–66.

[11] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al., The Kaldi speech recognition toolkit, Tech. rep., IEEE Signal Processing Society (2011).

[12] V. Peddinti, D. Povey, S. Khudanpur, A time delay neural network architecture for efficient modeling of long temporal contexts, in: Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[13] Y. Estève, M. Bouallegue, C. Lailler, M. Morchid, R. Dufour, G. Linarès, D. Matrouf, R. D. Mori, Integration of word and semantic features for theme identification in telephone conversations, in: 6th International Workshop on Spoken Dialog Systems (IWSDS 2015), 2015.

[14] L. Mangu, E. Brill, A. Stolcke, Finding consensus among words: Lattice-based word error minimization, in: Sixth European Conference on Speech Communication and Technology, 1999.

[15] G. Evermann, P. Woodland, Posterior probability decoding, confidence estimation and system combination, in: Proc. Speech Transcription Workshop, Vol. 27, Baltimore, 2000, p. 78.

[16] F. Wessel, R. Schluter, K. Macherey, H. Ney, Confidence measures for large vocabulary continuous speech recognition, IEEE Transactions on speech and audio processing 9 (3) (2001) 288–298.

[17] A. Nasr, F. Béchet, J.-F. Rey, B. Favre, J. Le Roux, Macaon: An NLP tool suite for processing word lattices, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations, 2011, pp. 86–91.

[18] G. Gravier, G. Adda, N. Paulsson, M. Carr, A. Giraudel, O. Galibert, The ETAPE corpus for the evaluation of speech-based TV content processing in the French language, in: Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12), 2012.

[19] P. Deléglise, Y. Estève, S. Meignier, T. Merlin, Improvements to the LIUM French ASR system based on CMU Sphinx: what helps to significantly reduce the word error rate?, in: Interspeech, Brighton, UK, 2009.

[20] S. Ghannay, Y. Estève, N. Camelin, Word embeddings combination and neural networks for robustness in ASR error detection, in: European Signal Processing Conference (EUSIPCO 2015), Nice (France), 2015.

[21] S. Ghannay, B. Favre, Y. Estève, N. Camelin, Word embedding evaluation and combination, in: 10th edition of the Language Resources and Evaluation Conference (LREC 2016), Portorož (Slovenia), 2016.

[22] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: arXiv preprint arXiv:1301.3781, 2013.

[23] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation., in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Vol. 14, 2014, pp. 1532–1543.

[24] O. Levy, Y. Goldberg, Dependency-based word embeddings, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Vol. 2, 2014, pp. 302–308.

[25] Lavergne, Thomas and Cappé, Olivier and Yvon, François, Practical Very Large Scale CRFs, in: Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL), 2010, pp. 504–513.

[26] H. Kamper, W. Wang, K. Livescu, Deep convolutional acoustic word embeddings using word-pair side information, in: arXiv preprint arXiv:1510.01032, 2015.

[27] K. Levin, K. Henry, A. Jansen, K. Livescu, Fixed-dimensional acoustic embeddings of variable-length segments in low-resource settings, in: Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop, IEEE, 2013, pp. 410–415.

[28] S. Bengio, G. Heigold, Word embeddings for speech recognition., in: INTERSPEECH, 2014, pp. 1053–1057.

[29] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu, Learning fine-grained image similarity with deep ranking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1386–1393.

[30] Weston, Jason and Bengio, Samy and Usunier, Nicolas, Wsabie: Scaling up to large vocabulary image annotation, in: IJCAI, Vol. 11, 2011, pp. 2764–2770.

[31] S. Galliano, E. Geoffrois, D. Mostefa, K. Choukri, J.-F. Bonastre, G. Gravier, The ESTER phase II evaluation campaign for the rich transcription of French Broadcast News., in: Interspeech, 2005, pp. 1149–1152.

[32] S. Galliano, G. Gravier, L. Chaubard, The ESTER 2 evaluation campaign for the rich transcription of French radio broadcasts., in: Interspeech, Vol. 9, 2009, pp. 2583–2586.

[33] Y. Estève, T. Bazillon, J.-Y. Antoine, F. Béchet, J. Farinas, The EPAC Corpus: Manual and Automatic Annotations of Conversational Speech in French Broadcast News., in: LREC, Citeseer, 2010.

[34] S. Stoyanchev, P. Salletmayr, J. Yang, J. Hirschberg, Localized detection of speech recognition errors, in: Spoken Language Technology Workshop (SLT), 2012 IEEE, 2012, pp. 25–30.

[35] J. Hirschberg, D. Litman, M. Swerts, Prosodic and other cues to speech recognition failures, Speech Communication 43 (1) (2004) 155–175.

[36] S. Goldwater, D. Jurafsky, C. D. Manning, Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates, Speech Communication (2010) 181–200.

[37] J.-L. Gauvain, G. Adda, L. Lamel, F. Lefvre, H. Schwenk, Transcription de la parole conversationnelle, Traitement Automatique des Langues 45 (3) (2005) 35–47.

[38] P. Boersma, D. Weenink, Praat, a system for doing phonetics by computer, Glot International 5 (9) (2001) 341–345.

[39] M. Adda-Decker, C. Gendrot, N. Nguyen, Contributions du traitement automatique de la parole l'étude des voyelles orales du franais, Traitement Automatique des Langues 49 (3) (2008) 13–46.

[40] R. Nemoto, M. Adda-Decker, J. Durand, Investigation of lexical F0 and duration patterns in French using large broadcast news speech corpora, in: Proceedings of Speech Prosody, 2010.

[41] Q. V. Le, T. Mikolov, Distributed Representations of Sentences and Documents., in: ICML, Vol. 14, 2014, pp. 1188–1196.

[42] Y. Lin, H. Lei, J. Wu, X. Li, An empirical study on sentiment classification of chinese review using word embedding, arXiv preprint arXiv:1511.01665.

[43] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, M. Zhou, Sentiment embeddings with applications to sentiment analysis, IEEE Transactions on Knowledge and Data Engineering 28 (2) (2016) 496–509.

[44] Y. Ren, R. Wang, D. Ji, A topic-enhanced word embedding for Twitter sentiment classification, Information Sciences 369 (2016) 188–198.

[45] Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[46] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[47] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, S. Khudanpur, Purely sequence-trained neural networks for asr based on lattice-free mmi., in: Interspeech, 2016, pp. 2751–2755.