Learning word embeddings: unsupervised methods for fixed-size representations of variable-length speech segments

Nils Holzenberger1,*, Mingxing Du2, Julien Karadayi2, Rachid Riad1,2,3, Emmanuel Dupoux2

1CLSP, Johns Hopkins University, USA
2ENS, EHESS, PSL Research University, CNRS, INRIA, France
3INSERM, NPI, UPEC, France

nholzen1@jhu.edu, mingxing.du@polytechnique.edu, \{julien.karadayi, riadrachid3, emmanuel.dupoux\}@gmail.com

Abstract

Fixed-length embeddings of words are very useful for a variety of tasks in speech and language processing. Here we systematically explore two methods of computing fixed-length embeddings for variable-length sequences. We evaluate their susceptibility to phonetic and speaker-specific variability on English, a high resource language, and Xitsonga, a low resource language, using two evaluation metrics: ABX word discrimination and ROC-AUC on same-different phoneme n-grams. We show that a simple downsampling method supplemented with length information can be competitive with the variable-length input feature representation on both evaluations. Recurrent autoencoders trained without supervision can yield even better results at the expense of increased computational complexity.

Index Terms: unsupervised speech processing, audio word embeddings, ABX discrimination, same-different classification, representation learning.

1. Introduction

Techniques to efficiently embed words into a vector space [1, 2] have led to spectacular successes in several Natural Language Processing (NLP) tasks such as language modeling [3], translation [4], topic modeling [5] and anaphora resolution [6]. In speech, similar techniques could be very useful, and are already used for several applications such as query by example [7], spoken term discovery [8] and unsupervised representation learning [9, 10]. However, before applications inspired by NLP can be deployed, a complication specific to speech has to be addressed. In text, the embedding for two instances of the same word will be the exact same vector. With audio input, due to phonetic and speaker variability, they won’t be identical. Instead, we are dealing with a distribution over vectors, potentially overlapping with the distribution of another word. Therefore, a prerequisite for developing useful vector representations for spoken words is to minimize the impact of this variability.

Here, we propose to systematically explore the phonetic variability of audio word embeddings using two widely different approaches. The first one has been proposed by [11]. It relies on the idea of downsampling the time sequence of acoustic information using a fixed number of samples. This is a simple idea, which requires no training, and is worth investigating, at least as a baseline. The second idea is to use Recurrent Neural Networks (RNN) as autoencoders, by training them to encode a sequence of frames into a vector, and to decode it back to the original sequence of frames. Contrary to downsampling, this technique requires training, but no labels are needed, making it therefore applicable to a variety of speech signals.

We propose to evaluate the phonetic variability of word embeddings using two metrics, based on the computation of the within-word and between-word distance of embeddings. The first one, ABX, tests the average discriminability between pairs of words. The second one, AUC, tests the separation, across the entire corpus, of the distribution of cosine or Euclidean distance for pairs of same phoneme n-grams versus the distribution of that distance for pairs of different phoneme n-grams.

2. Related work

The framework of segmental speech recognition [12] involves an explicit segmentation of speech utterances and a fixed-size representation for each acoustic segment. [13] use mean pooling and temporal derivatives over subparts of each segment, duration and energy, while [14] warp segments into a fixed length. Recent approaches represent segments with RNNs [15], or Deep Neural Networks (DNN) together with subsampling, pooling and duration information [16, 17]. Fixed-size embeddings are also used for speaker verification and extracted with factor analysis [18] or DNNs [19]. DNNs have been used to derive embeddings by training them with a supervised objective, although not for speech recognition per se: [20, 21] alternate convolutional and pooling layers, finishing with a fixed length vector.

In [22], word embeddings are obtained unsupervisedly by first memorizing a reference set of N words, and calculating the Dynamic Time Warping [23] (DTW) distance of a new target word with each of these references. This set of distances is then projected into a subspace by dimensionality reduction. This method obtains good quality embeddings with a large reference set but suffers from a high computational cost. In [11], this approach is replaced by a simpler procedure, where a fixed-size embedding is extracted from a sequence of frames by picking 10 equally-spaced frames from the sequence. In this paper, we will explore improvements of this technique.

More recently, sequence-to-sequence autoencoders [9, 10] have been used on speech to learn fixed-length embeddings in a purely unsupervised fashion. This has the advantage of providing embeddings in low-resource settings, for datasets where a transcript is not available. This is the kind of architecture and training setup we will use in this paper.

3. Proposed approach

Even though both downsampling and recurrent autoencoders have been used in past work, they have never been directly
compared, nor have the hyperparameters of downsampling been thoroughly explored. In addition, we compare these models with a DTW baseline, which aligns raw features to compute the distance between two sequences.

### 3.1. Downsampling

The downsampling techniques used in [11] extract a fixed number of equidistant samples from a time series. Here, we explore variants of this idea.

Formally, let \( x_1, x_2, \ldots, x_T \), with \( x_i \in \mathbb{R}^{40} \) be a sequence of 40-dimensional log mel features. Equidistant downsampling samples \( k \) vectors at interval \( \frac{T}{k-1} \) with proportional interpolation as needed. Let \( \hat{x}_{q_i} \) be the \( i \)-th sample, where \( 1 \leq i \leq k \) and \( 1 = q_1 < q_2 < \ldots < q_k = T \), \( q_{j+1} - q_j = \frac{T}{k-1} \) for \( j = 1, 2, \ldots, k-1 \). Note \( \lfloor \cdot \rfloor \) and \( \lceil \cdot \rceil \) the floor and ceiling functions, then we have:

\[
\hat{x}_{q_i} = x_{\lfloor q_i \rfloor} \cdot (q_i - q_{\lfloor q_i \rfloor}) + x_{\lceil q_i \rceil} \cdot (q_{\lceil q_i \rceil} - q_i)
\]

When \( q_i \) is an integer, we take \( x_{q_i} \) as the sample; otherwise, the sample is a weighted sum of its left and right neighbors. The closer the neighbor, the more weight it contributes. The embedding of \( x_1, x_2, \ldots, x_T \) is the concatenation of \( \hat{x}_{q_1}, \hat{x}_{q_2}, \ldots, \hat{x}_{q_k} \).

The acoustic information in a word is not necessarily distributed uniformly along the time axis; psycholinguistics suggest that it is mostly located at the boundaries. Thus, we introduce non-equidistant downsampling: we assume the space between two samples follows a linear progression and is symmetric with respect to the center, i.e. \( \Delta_j := q_{j+1} - q_j = k(j-1) + b \) and \( q_{j+1} - q_j = kq_{j-1} - kq_j \) for \( j = 1, 2, \ldots, \lfloor k/2 \rfloor \). To illustrate the degree of non-equidistant, instead of setting \( k \) as hyperparameter, we introduce the ratio \( \alpha = \frac{\Delta(k/2)}{\Delta_1} \). Using \( q_1 = 1 \) and \( q_k = T \), \( k \) and \( b \) can be deduced from \( \alpha \). \( \alpha \) represents the ratio of the middle space against the first space. For example if \( \alpha > 1 \) then we take more samples at the two ends, if \( \alpha < 1 \) we take more samples in the middle, if \( \alpha = 1 \), then we come back to equidistant downsampling.

We further extend proportional interpolation to Gaussian weighted interpolation, the idea being to avoid loosing information. Formally, consider the sequence of log mel features as a step function i.e. \( f(t) = x_t \) for \( -0.5 < t < +0.5 \). We then calculate a Gaussian density function centered at \( q_i \) with variance \( \sigma^2 \). We have the following formula and illustrate it in figure 1:

\[
\hat{x}_{q_i} = \frac{\int_{-0.5}^{0.5} g_i(t) f(t) dt}{Z_i}
\]

where \( Z_i \) is the normalization term, \( Z_i = \int_{-0.5}^{0.5} g_i(t) dt \).

Combining the downsampling methods with Gaussian weight, we have tested 5 variants with equidistant downsampling:

- **(EPI) Proportional interpolation (baseline):** no hyperparameters.
- **(EAG) Absolute Gaussian weight:** one hyperparameter \( \sigma \), as described previously.
- **(ERG) Relative Gaussian weight:** one hyperparameter \( \beta \), setting \( \sigma \) proportional to the sequence length, i.e. \( \sigma = \beta T \).
- **(EALG) Absolute linear Gaussian weight:** two hyperparameters \( k, \sigma \). Instead of setting \( \sigma \) for all samples, we set \( \sigma_j \) for each sample to be a linear function and symmetric with respect to the center, i.e. \( \sigma_j = k(j-1) + \sigma_1 \) and \( \sigma_j = \sigma_{k-j+1} \) for \( j = 1, 2, \ldots, \lceil k/2 \rceil \).
- **(ERLG) Relative linear Gaussian weight:** two hyperparameters \( k, \beta \). The only difference from the previous one is that we set the intersection proportional to the sequence length, i.e. \( \sigma_j = k(j-1) + \beta T \) and \( \sigma_j = \sigma_{k-j+1} \) for \( j = 1, 2, \ldots, \lceil k/2 \rceil \).

• (ERLG) Relative linear Gaussian weight: two hyperparameters \( k, \beta \). The only difference from the previous one is that we set the intersection proportional to the sequence length, i.e. \( \sigma_j = k(j-1) + \beta T \) and \( \sigma_j = \sigma_{k-j+1} \) for \( j = 1, 2, \ldots, \lceil k/2 \rceil \).

• (ERLG) Relative linear Gaussian weight: two hyperparameters \( k, \beta \). The only difference from the previous one is that we set the intersection proportional to the sequence length, i.e. \( \sigma_j = k(j-1) + \beta T \) and \( \sigma_j = \sigma_{k-j+1} \) for \( j = 1, 2, \ldots, \lceil k/2 \rceil \).

5 non-equidistant downsampling methods:

- **(NPI) Proportional interpolation:** one hyperparameter \( \alpha \), as described previously.
- **(NAG) Absolute Gaussian weight:** two hyperparameters \( \alpha, \sigma \). \( \alpha \) for non-equidistant downsampling, \( \sigma \) for Gaussian weight.
- **(NRG) Relative Gaussian weight:** two hyperparameters \( \alpha, \beta \). \( \alpha \) for non-equidistant downsampling, setting \( \sigma \) proportional to the sequence length, i.e. \( \sigma = \beta T \).
- **(NALG) Absolute linear Gaussian weight:** two hyperparameters \( \alpha, \sigma \). \( \alpha \) for non-equidistant downsampling, setting Gaussian to the same slope calculated by non-equidistant downsampling, i.e. \( \Delta_j := q_{j+1} - q_j = k(j-1) + b \), \( \alpha = \frac{\Delta(k/2)}{\Delta_1} \), \( q_{j+1} - q_j = kq_{j-1} - kq_j \), \( \sigma_j = k(j-1) + \sigma_1 \), \( \sigma_j = \sigma_{k-j+1} \) for \( j = 1, 2, \ldots, \lceil k/2 \rceil \).

• (NRLLG) Relative linear Gaussian weight: two hyperparameters \( \alpha, \beta \). The only difference from the previous one is that we set the intersection proportional to the sequence length, i.e. \( \sigma_j = k(j-1) + \beta T \), \( \sigma_j = \sigma_{k-j+1} \) for \( j = 1, 2, \ldots, \lceil k/2 \rceil \).

Finally, preliminary results suggest that including length information in the embeddings can further improve the quality of the embeddings. To this end, we compute the L2 norm (approximate the energy for each frame) for each log mel feature vector, and cut off the sequence to 150 frames if larger, pad with 0 if smaller than 150 frames. After scaling the L2 norms with a trade-off hyperparameter \( \gamma \), we concatenate them to the downsampling embedding. Our results suggest that with this L2 norm, we achieve an average gain on ABX error of 0.09% and 0.01% for Xitsonga and English respectively. Downsampling each sequence of log mel features to 20 samples (each the output of 40-dimensional log mel filter banks), then concatenating these samples along with the L2 norm, leads to a final embedding size of 950.

### 3.2. Recurrent autoencoders

RNNs, in particular Long-Short Term Memory networks (LSTM) [24], are the basis of state of the art solutions for numerous speech processing tasks. LSTMs can learn fixed-size...
representations of variable-length sequences, in text [25, 26] and speech processing [9, 10]. Our proposed model, shown in figure 2, follows the encoder-decoder architecture used in [9], frequently used in machine translation [27], and borrows some ideas from segmental ASR.

Let \( x_t \) be the \( t \)-th 40-dimensional log mel feature vector, concatenated with 3 context frames on either side, a total of 7 concatenated frames (thus \( x_t \in \mathbb{R}^{280} \)). The encoder, a multilayer bidirectional LSTM [28], reads a sequence \( x_1, \ldots, x_T \) and encodes it into a sequence of hidden states \( h_1, \ldots, h_T \), where \( h_t = [h^\text{forward}_t, h^\text{backward}_t] \) and \( [a, b] \) is the concatenation of vectors \( a \) and \( b \). We extract 3 representations from the encoder. The last hidden state of the forward LSTM \( h^\text{forward}_T \), the last hidden state of the backward LSTM \( h^\text{backward}_1 \); an element-wise maxpooling along the time axis of the sequence of hidden states, \( v = \text{maxpool}(h_t) \) so that \( |v| = \max_i |h_t[i]| \). These are concatenated and used as a representation of the full sequence: \( c = [h^\text{forward}_T, h^\text{backward}_1, v] \). The vector \( c \) is fed to an LSTM decoder at every time step \( t \), along with a learned embedding \( e_t \in \mathbb{R}^{250} \) of the integer \( t \). The decoder’s input at time step \( t \) is thus \( [c, e_t] \), and its target output is \( \hat{x}_t \). We train the network with Mean Squared Error (MSE) loss: \( \sum_t ||x_t - \hat{x}_t||^2 \). The network is trained on segments of no more than 150 frames.

The training data has a very large amount of samples (English: 1.7M, Xitsonga: 653K), making it impractical to evaluate the neural networks on the timescale of epochs. Instead, we set aside 10% of the training data as a validation set. During training, we randomly take 100 samples from the training data and perform gradient descent using MSE loss and the Adam optimizer [29] with standard settings on this batch. After 100 training batches (i.e. \( 10^4 \) training samples), we randomly take 1000 samples from the validation set and use them to measure validation MSE. We use the neural net with the lowest validation MSE as our final model. The larger the model, the lower the MSE, so we must use the ABX score to compare architectures.

### 3.3. Evaluation

Given the broad scope of applications for low-resource speech technologies, and to develop methods fairly independent of their final application, we chose to evaluate the quality of embeddings intrinsically. We evaluate the learned embeddings with ABX discrimination tasks [30] across speakers. We sample triplets of (sequence \( A \), sequence \( B \), sequence \( X \)) where each sequence is a sequence of acoustic features; \( A \) and \( B \) were said by speaker 1; \( X \) was said by speaker 2, and the label associated with \( X \) (e.g. “cat”, or “ow-k”) is identical to that of either \( A \) or \( B \). In this paper, the label is either a word or an \( n \)-gram of phonemes. Using the method we wish to evaluate, we compute embeddings \( a, b \) and \( x \) from sequences \( A, B \) and \( X \). Using distance metric \( d \), we compute pairwise distances between \( a, b \) and \( x \). We report two different metrics for this task.

- In the first setting, denoted ABX, sequences \( A \), \( B \) and \( X \) are words. We look at the classification task of predicting whether \( X \) shares its label \( y_X \) with \( A \) or \( B \). We predict
  \[
  \hat{y}_X = \begin{cases} 
  y_A & \text{if } d(a, x) < d(b, x) \\
  y_B & \text{otherwise}
  \end{cases}
  \]
  and report error on this classification task. Triplets are sampled such that the performance of random guessing is 50%.
- In the second setting, denoted AUC, there is no \( X \); sequences \( A \) and \( B \) correspond to phoneme \( n \)-grams where \( n = 2 \ldots 12 \). We are still looking at a classification task, with
  \[
  \hat{y} = \begin{cases} 
  \text{“same”} & \text{if } d(a, b) < \theta \\
  \text{“different”} & \text{otherwise}
  \end{cases}
  \]
  We look at the ROC curve obtained by measuring precision and recall for various values of \( \theta \). The score reported is the area under the ROC curve. Higher is better, and means less sensitivity with respect to parameter \( \theta \) [31].

For DTW, embedding \( z \) is actually \( X \) itself, and \( d \) is the DTW cosine distance with or without length normalization. For downsampling, \( d \) is the cosine distance, shown to work best for log mel features. For neural networks, \( d \) is the Euclidean distance, as the most natural distance metric over a vector space.

### 3.4. Datasets

We used the English and Xitsonga datasets provided by the Zero Resource Speech Challenge 2015 [32] to design and evaluate our models. The English corpus is a subset of the Buckeye corpus [33] (denoted as “English” in the following), and the Xitsonga corpus is the NCHLT Xitsonga Speech Corpus [34] (denoted as “Xitsonga” in the following). Speech is represented as a sequence of 40-dimensional log mel feature vectors, computed on a sliding window of 25 ms, offset by 10 ms. The filter banks were mean and variance normalized per file (excluding silences), to help remove background noise and reduce across-speaker variability. Here are, in ms, average and standard deviation for the duration of English words: 236.2 and 153.6; English 2-to-12-grams: 547.7 and 500.1; Xitsonga words: 450.8 and 290.4; Xitsonga 2-to-12-grams: 617.4 and 327.2.

### 4. Results and discussion

For each downsampling method, we have conducted several grid searches. We kept the parameters achieving the best average ABX error on both corpora, since we would like parameters which can generalize well enough across different corpora. As shown in Table 1, although the best methods for English and for Xitsonga are different, the Equidistant downsampling with Absolute Linear Gaussian weight (EALG) method, which achieves the best average ABX, performs close enough to the best method on both corpora.

We investigated varying the depth and the number of hidden units of the recurrent autoencoders. Note that the size of the embedding is equal to twice the number of hidden units. Results from table 1 show, across both languages, that the larger the size
of the embedding, the higher the ABX score and AUC. Depth of the network does not correlate with better scores.

On the AUC task, on both corpora, we find that recurrent autoencoder embeddings perform best, and downsampling outperforms normalized DTW by a comfortable margin. For the ABX task, normalized DTW and our methods perform some-

Figure 4: ABX error (%) and AUC score. "AE" means RNN autoencoder, with the architecture of the encoder indicated as the number of hidden units per layer. The decoder’s architecture mirrors the encoder’s.

Table 1: ABX error (%) and AUC score. "AE" means RNN autoencoder, with the architecture of the encoder indicated as the number of hidden units per layer. The decoder’s architecture mirrors the encoder’s.

5. Conclusion

In this paper, we propose two parametric methods to extract fixed-size embeddings from variable-length sequences of acoustic frames. The geometry of these embeddings captures linguistic properties, sometimes better than the geometry of the original acoustic space, as measured by DTW. We have thoroughly explored different strategies of downsampling, a method with few parameters, fast to compute and robust across languages. RNNs, trained with an unsupervised objective, and without any gold word-level segmentation of the input, manage to capture acoustic regularities, and further improve over downsampling. Using larger embeddings allows these models to fit the acoustic regularities even better, with higher ABX and AUC scores for larger networks. However, these embeddings come at a higher computational cost than downsampling.

Interestingly, one can find, for each method, hyperparam-
eters which give very good results for two very different languages. The RNN autoencoders even show some ability to generalize across languages after training. This suggests a possibility to precompile language-general spoken word embedders. In future work, it would be interesting to further improve the RNN autoencoders by training them as denoising autoencoders, using data augmentation such as adding reverberation, and encouraging the autoencoder to ignore speaker information, e.g. with a well-designed loss function.

6. Acknowledgements

This project is funded by the European Research Council (ERC-2011-AdG-295810 BOOTPHON), the Agence Nationale pour la Recherche (ANR-10-LABX-0087 IEC, ANR-10-IDEX-0001-02 PSL* ), Microsoft Research (joint MSR-INRIA center) and a Google Award Grant.
7. References


