INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

DNN bottleneck features for speaker clustering

Jesús Jorrín[1], *Paola García*[1], *Luis Buera*[1]

[1]Nuance Communications, Inc.

{jesus.jorrin,paola.garcia,luis.buera}@nuance.com

Abstract

In this work, we explore deep neural network bottleneck features (BNF) in the context of speaker clustering. A straightforward manner to deal with speaker clustering is to reuse the bottleneck features extracted from speaker recognition. However, the selection of a bottleneck architecture or nonlinearity impacts the performance of both systems. In this work, we analyze the bottleneck features obtained for speaker recognition and test them in a speaker clustering scenario. We observe that there are deep neural network topologies that work better for both cases, even when their classification criteria (senone classification) is loosely met. We present results that outperform a traditional MFCC system by 21% for speaker recognition and between 20% and 37% in clustering using the same topology.
Index Terms: Speaker clustering, DNN features, AHC.

1. Introduction

Neural networks have become popular in the last years. Several are the fields in which they have been successfully applied; for example, computer vision, speech recognition, or natural language processing. The outstanding improvements in the performance of speech recognition systems have influenced other fields such as language or speaker recognition [1, 2, 3, 4].

Recently, the research efforts in speaker identification (SID) are focused on building end-to-end system based entirely on deep neural networks (DNNs). It is not an easy task due to the small amount of data per speaker. However, indirect ways to include DNNs are now part of hybrid systems that combine DNNs and the traditional approach. Two of the most suitable solutions, in this sense, are to extract deep neural network bottleneck features (BNF) or to compute GMMs posterior probabilities using a DNN. In this work, we explore the former case. The performance of BNF has shown to overcome mel frequency cepstral coefficient (MFCC) systems [4]. Then, the question that arises is: *are these features also suitable for speaker clustering?* Hence, the final goal is to select a DNN configuration that outperforms the current results in SID and clustering.

In this work, we deal with the design of a speaker clustering solution that uses a speaker identification system as an input of the clustering process. At this point, several DNN features will be presented, and we will compare their results with the performance obtained by a traditional MFCC system. We investigated the inner characteristics of the DNN topologies that signify an improvement or a deterioration of the results. Furthermore, the analysis will not only help to decide the BNF, but exemplify the different DNN architectures that are suitable for SID and speaker clustering.

This paper is organized as follows. Section 2 and 3 give an overview of the clustering algorithm and the DNN features we are considering. Section 4 describes the system focusing on the different DNN configuration we explored. Section 5 depicts the experiments, the audio database and also the performance measures concentrating on the differences between the two evaluated tasks. The results are presented in Section 6. Finally, Section 7 summarizes the conclusions of the experiments.

2. Clustering algorithm

Speaker clustering means to classify speech segments into groups such that each group contains speech from only one speaker, and ensures that the speech from the same speaker is grouped into the same cluster. We can find many approaches that solve the clustering problem; for example, sequential, hierarchical, or cost function optimization implementations [5]. We consider a bottom-up Agglomerative Hierarchical Clustering (AHC), as in [6]. This is a greedy algorithm that starts with a number of clusters equal to the number of speech segments and it merges the closest clusters iteratively until a stopping criterion is met. There are several ways of implementing these kind of algorithms, such as those based on matrix theory or graph theory [5]. For our purpose, we use an approach based on matrix theory. The input for these kind of implementations is a proximity matrix $P(X)$, where $X = \{x_i\}, i = 1...N$ is the set of elements to be clustered. $P(X)$ is an $N \times N$ matrix with (i, j) component equal to the similarity $s(x_i, x_j)$, or dissimilarity $d(x_i, x_j)$ between the elements x_i and x_j. In this paper we only consider similarity measures. At each iteration t, of the clustering process, the clusters with the highest value in the similarity matrix are merged. After this merger, a new similarity matrix P_t is computed. The dimension of P_t is equal to the dimension of the similarity matrix from the previous iteration P_{t-1} minus one. In this way, $P_0 = P(X)$ with dimension $N \times N$ and P_t will have dimensions $(N - t) \times (N - t)$. To obtain the new matrix P_t, we start from P_{t-1} and we proceed as follows: 1) delete the two columns and rows associated with the merged clusters, and 2) add a new row and column with the distances between the new cluster and the old ones. We can find many approaches to compute these new distances such as single/complete/average linkages or centroids methods [5]. In this work we consider two different linkage algorithms: a) single linkage algorithm, in which the similarity between two clusters is

$$s(C_{pq}, C_r) = max\{s(C_p, C_r), s(C_q, C_r)\}, \quad (1)$$

and b) average linkage algorithms in which

$$s(C_{pq}, C_r) = \frac{1}{2}(s(C_p, C_r) + s(C_q, C_r)), \quad (2)$$

where C_{pq} is the new cluster formed by merging clusters C_p and C_q and C_r is an old cluster. This process is repeated until no more mergers are available. The selection of the linkage algorithm is critical while designing a clustering algorithm; it will condition the mergers along the clustering process. In this sense, the single linkage algorithm has a tendency to favor elongated clusters, while average linkage favors to create smaller compact clusters. This information may help to select the best

http://dx.doi.org/10.21437/Interspeech.2017-144

linkage method if there exists prior knowledge about the clusters structure.

2.1. Clustering performance measures

Some common quality measures to evaluate the performance of a clustering task are: the rand index, F-measure, the mutual information or purity measures [7]. In this work, we consider the latter. The purity is a measure of cluster cleanness: clusters containing data from a single speaker. Particularly, we will consider speaker impurity and cluster impurity measures defined in [6]. The first one measures how spread the speakers are between the different clusters in a single partition. The second one measures to what extent clusters contain audios from different speakers. We will use these two measures, since they evaluate the trade-off between grouping audios from common speakers and having pure clusters; *i.e.*, clusters containing audios from one unique speaker.

3. Bottleneck feature extractor

DNNs have shown to be effective for SID. One of the main contributions is a new set of features called: bottleneck features (BNF). They are DNNs that have a hidden layer with less neurons (bottleneck - BN) compared to the rest. For SID these type of DNNs work in an indirect way. DNNs goal is to classify among senones at softmax output layer. Once trained, the network model is truncated at the bottleneck. The feature vectors are extracted from this output layer. It is expected that these new vectors provide good alignments and also preserve the information of the speaker [8]. The BNFs are treated in the same manner as the MFCC throughout the SID process; *i.e.*, training of the UBM, ivector extractor and PLDA deal with these new features in the traditional way.

Depending on the selection of a DNN topology, these new features impact the training, enrollment, and verification. It is not evident how these nonlinearities and architectures work in speaker recognition and clustering. However, we can analyze their behavior from the perspective of i-vector construction and explain the good or bad performance in the final speaker clustering performance.

Five architectures and nonlinearities were used for this research:

- **tanh**: It is a DNN similar to sigmoid. The lower layers of the DNN have gradients close to zero causing higher layers to present vanishing gradient. This effect causes slow optimization convergence and the possibility to converge to local minimum [9]. The learning of this net preserves speaker info, even when the accuracy to senones is not optimal (see Table 1 and Figure 3a).
- **pnorm**: It is denoted by $y = ||x||_p$, a not bounded nonlinearity, which is a good property for obtaining accurate results and mitigating the effect of vanishing gradient. However, it requires a normalization layer to prevent instability [10]. To our knowledge, $p = 2$ has shown the best results for speaker and speech recognition (for clarity, we will refer to pnorm with p=2 as pnorm along this document). It shows better alignment from ASR point of view and preserves the speaker information (see Table 1 and Figure 3a).
- **LSTM** (Long Short Time Memory): It is a recurrent neural network (RNN) mainly used in ASR, with very successful results [11]. It is more accurate due to its ability to learn long time dependencies. However, being the purpose to recognize senones, the speaker information is not kept at the iterations (see Table 1 and Figure 3a).

Apart from the architectures and non-linearities and inspired by the work in [12], the position of the bottleneck was explored for our best nonlinearity: pnorm.

- **BN closer to the input layer:** pnorm-2 (bottleneck in the second layer) preserves speaker information; although the alignment is diminished (see Table 1 and Figure 4a).
- **BN closer to the output layer:** pnorm-4 (bottleneck in the forth layer), the senone alignment is better; but the speaker information is not retained (see Table 1 and Figure 4a).

4. System set-up

We employ a SID traditional system for our experiments in which the MFCC are substituted by BNFs. The SID system is trained and tested in the usual way.

4.1. DNN training

DNNs input are MFCCs with short-time normalization. Five hidden-layers were employed for tanh and pnorm; LSTM uses three hidden-layers. All of the DNNs were trained using Kaldi [13]. Training set includes 300 hours of Fisher database. 60-dimensional BNF vectors are extracted and are then the input for the traditional GMM ivector-PLDA system.

4.2. DNN frame accuracy

A simple way to describe the DNN's performance in ASR is in terms of frame accuracy. The classification criterion based on senone distinction give us an idea how good the DNN is able to recognize acoustic units such as triphonemes. This accuracy is close related to the WER and it is a previous hint on the alignment information of our data. However, due to the intrinsic process of the DNN architecture it is not guaranteed that the speaker information is preserved over iterations or layer position.

Table 1: *Frame accuracy per topology.*

Topology	tanh	pnorm-2	LSTM	pnorm-4
Accuracy	39.3%	43.6%	51.3%	43.0%

As shown in Table 1. The best accuracy is given by the LSTM, The pnorm family gives similar results and tanh is the lowest. In the following sections we will link this behavior to our findings.

4.3. Similarity measures

So far we have described the clustering algorithm without going into details on the similarity measures used to compute the proximity matrix $P(X)$. For this purpose, we consider as similarity measure the raw scores provided by a PLDA system [14]:

$$s(x_i, x_j) = score_{PLDA}(x_i, x_j), \quad (3)$$

Two classifier schemes are tested depending on the features:

- **MFCC-PLDA:** Our baseline system is based on a full covariance Universal Background Model (UBM) of 256

GMM-component, using MFCCs. 400 dimensional i-vectors are extracted consequently. Scoring between i-vectors is achieved by using gender dependent PLDA. SRE04, SRE05, SRE06 and SRE08 data were used to train the UBM, i-vector extractor and the PLDA.

- **DNN-PLDA:** This system is similar to the previous one, except that BNFs are used instead of MFCC. UBM is re-trained according to the new features. Several neural networks are considered in this work: LSTM, tanh, pnorm-2 (BN closer to the input layer, second hidden layer) and pnorm-4 (BN closer to the output layer, fourth hidden layer)

5. Experiments

In the following experiment we consider a baseline system based on MFCC features and BNF. We compare the results obtained with each set of features while solving a traditional SID task and a clustering task. To do so, we select a pool of audios and we compute all versus all scores. The obtained scores are used for both building the similarity matrix required by the clustering schemes and building a traditional SID task in which N models are evaluated against N audios.

5.1. Audio database

We consider a subset of audios from NIST SRE10 coreext-coreext condition. The subset has 1335 audios (634 males and 701 females) and 408 speakers (187 males and 221 females). The audios were selected to have an audios per speaker distribution as the one in Figure 1. The distribution models a generic use case in which the number of speaker with certain amount of audios decreases as the number of audios belonging to those speakers increases.

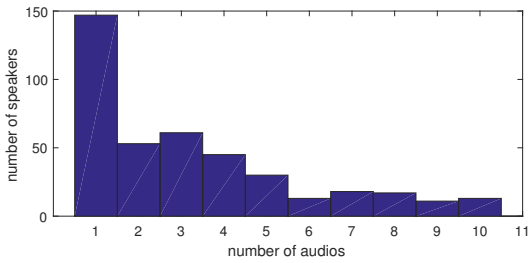


Figure 1: *Audios per speaker distribution.*

5.2. Performance measures

We evaluate clustering results in terms of the cluster impurity (CI) and speaker impurity (SI) measures. We compute CI and SI at each iteration of the clustering process and plot them in a single curve, see Figure 2. The point at which the SI and the CI are equal is denoted as equal impurity rate (EI). The EI measures how close we get to the partition in which there is exactly one cluster per speaker and each cluster contains audios from one single speaker. Additionally, we plot the DET curve associated with the scores used to build the similarity matrix.

6. Results

Two analysis are presented in this section. The first one focuses on the differences between DNN architectures. The second one analyses the impact of the bottleneck position.

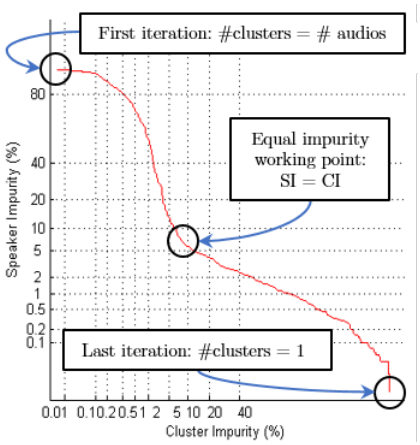


Figure 2: *Impurity tradeoff (IT) curve.*

Table 2: *MFCC baseline SI values and SI relative improvement of the MFCC baseline for the studied BNFs.*

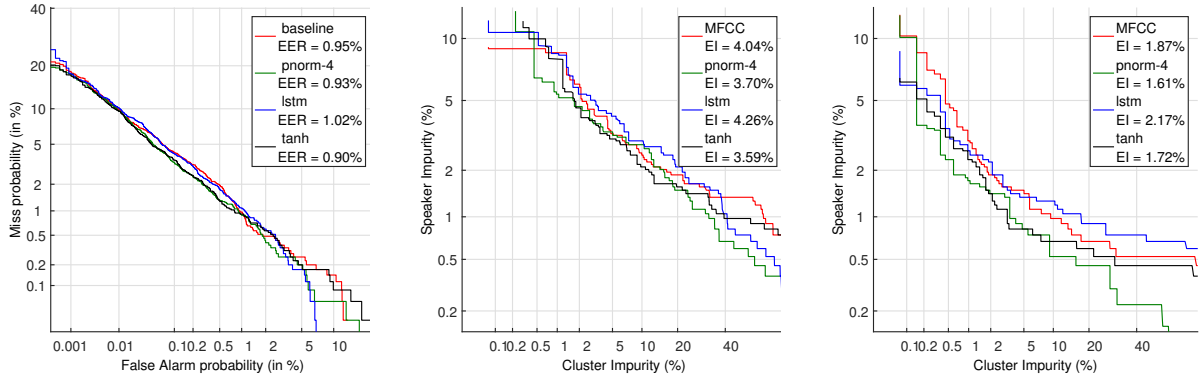
Feature	EI	SI@CI=1%	SI@CI=10%
Single linkage - Baseline			
MFCC	4.04%	8.6%	2.47%
Single linkage - Relative improvement			
pnorm-4	8.41%	37.32%	-15.38%
pnorm-2	**37.12%**	**51.27%**	**39.27%**
LSTM	-5.44%	0.81%	-21.45%
tanh	11.13%	6.04%	14.97%
Average linkage - Baseline			
MFCC	1.87%	2.99%	0.97%
Average linkage - Relative improvement			
pnorm-4	13.9%	**42.47%**	**46.39%**
pnorm-2	**20.32%**	32.44%	17.52%
lstm	-16.04%	12.37%	-30.92%
tanh	8.02%	22.40%	30.92%

6.1. BNF comparison

Figure 3a shows that all features perform similar for very low FA rates (lower than 0.01%). Tanh presents better performance than the baseline for FA rates between 0.01% and 1%, and pnorm-4 even for rates higher than 1%. Lastly, LSTM only improved the baseline for high FA working points.

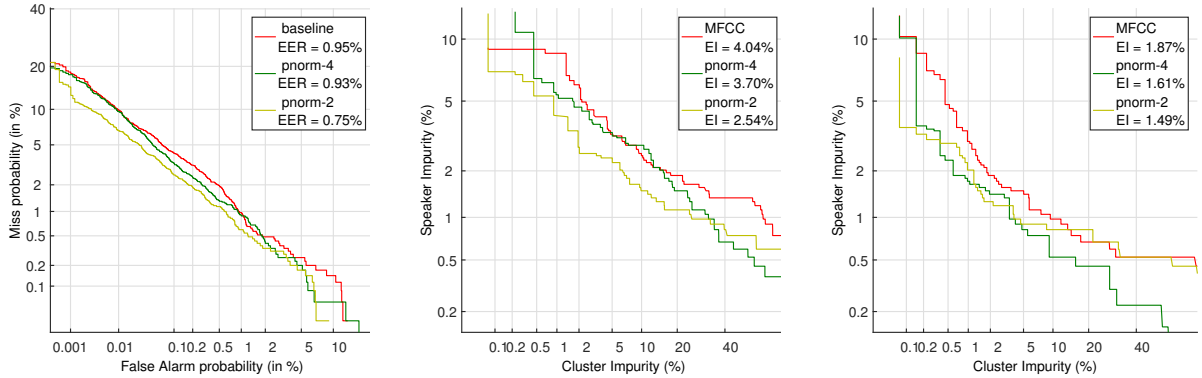
Considering the results for the clustering tasks (see Figures 3b and 3c and Table 2), we find that pnorm-4 and tanh outperform the baseline results for both linkage algorithms. A relative improvement at the EI of 8.4% (pnorm-4) and 11.1% (tanh) was measured for single linkage. The average linkage presented 13.9% (pnorm-4) and 8% (tanh) of relative improvement. We notice that single linkage curves show trends analogous to the DET curves. Comparable performance is obtained with tanh and pnorm-4 for the whole curves, both improving or keeping baseline results. LSTM performs similar to the baseline. On the other hand, for average linkage, pnorm-4 and tanh curves are not so close. pnorm-4 shows higher improvement with respect to the baseline compared with tanh, except for working points near EI. Results are summarized in Table 2.

Note that average linkage performs better than single linkage. From previous works [15], we know that the audios per speaker distribution of the database strongly affects the clustering task results. The audio database used in our experiments



(a) *DET curve.* (b) *IT curve. Single linkage algorithm.* (c) *IT curve. Average linkage algorithm.*

Figure 3: *BNF comparison*



(a) *DET curve.* (b) *IT curve. Single linkage algorithm.* (c) *IT curve. Average linkage algorithm.*

Figure 4: *Bottleneck position analysis*

does not present predominant speakers; *i.e.*, speakers with much more audios than all the others, which is in fact the most suitable case for single linkage. Hence, average linkage performs better.

6.2. Bottleneck position analysis

We describe our findings on our best nonlinearity, pnorm, when changing the position of the BN. Two configurations were tested: a) pnorm-4, with the BN closer to the output layer (fourth hidden layer); and b) pnorm-2, with the BN closer to the input layer (second hidden layer).

SID results (Figure 4a) show higher improvement when the BN is closer to the input. Improvement with respect to the baseline is observed along the whole DET curve, in contrast to pnorm-4.

For the clustering results (see Figures 4b and 4c), we notice again that single linkage produces analogous behavior as the DET curve. pnorm-2 shows higher performance than pnorm-4, 8.41% against 37.12% of relative improvement in the EI. On the other hand, this difference is not found for average linkage; even for high CI rates, pnorm-4 outperforms pnorm-2 results. This may result counterintuitive if we consider information extracted from the DET curve. However, both pnorm configurations outperform the baseline.

7. Conclusions

In this work we compared the results provided by different DNN solutions when solving a speaker clustering task. Two approaches of an AHC algorithm were tested. When considering single linkage approach, the conclusions were consistent with those observed in the SID task: the higher the performance in a DET curve, the higher the performance in the clustering task. The nonlinearity that performed the best was pnorm with a relative improvement of 8.41% in the EI compared to the baseline. We additionally observed that if the BN position was moved closer to the input of the DNN this improvement grew up to 37.12%. When average linkage was considered, pnorm also showed better performance than the other topologies, with relative improvement in the EI of 13.9%. In contrast to single linkage, the gain obtained when moving the BN position was not so evident. Both pnorms outperform the baseline at almost every point, but have several crosses. We observed that DET curves are not always the best way to evaluate scores for clustering purposes. For future research it may be interesting to study which score properties must be evaluated to measure the goodness of a system when solving a clustering task.

8. References

[1] M. McLaren, L. Ferrer, and A. Lawson, "Exploring the role of phonetic bottleneck features for speaker and language recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5575–5579.

[2] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[3] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černockỳ, "Multilingual bottleneck features for language recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] P. Matějka, O. Glembek, O. Novotnỳ, O. Plchot, F. Grézl, L. Burget, and J. H. Cernockỳ, "Analysis of dnn approaches to speaker identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5100–5104.

[5] S. Theodoridis and K. Koutroumbas, *Pattern recognition, Fourth Edition*. Academic Press, 2008.

[6] D. A. V. Leeuwen, "Speaker linking in large datasets," in *Odyssey2010, the Speaker Language and Recognition Workshop, Brno, Czech Republic*, 2010, pp. 202–208.

[7] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[8] A. Lozano-Diez, A. Silnova, P. Matejka, O. Glembek, O. Plchot, J. Pešán, L. Burget, and J. Gonzalez-Rodriguez, "Analysis and optimization of bottleneck features for speaker recognition," in *Odyssey 2016: The Speaker and Language Recognition Workshop*, 2016, pp. 21–24.

[9] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, no. 1, 2013.

[10] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 215–219.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[12] S. H. Ghalehjegh and R. C. Rose, "Deep bottleneck features for i-vector based text-independent speaker verification," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 555–560.

[13] 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," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[14] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[15] J. Jorrín-Prieto[1], C. Vaquero, and P. García[1], "Analysis of the impact of the audio database characteristics in the accuracy of a speaker clustering system."