Influence of speaker pre-training on character voice representation

Mathias Quillot, Jarod Duret, Richard Dufour, Mickael Rouvier, Jean-François Bonastre

Influence of speaker pre-training on character voice representation

Mathias Quillot[0000−0002−5858−2416], Jarod Duret[0000−0002−5755−2653], Richard Dufour[0000−0003−1203−9108], Mickael Rouvier[0000−0003−3541−3385], and Jean-François Bonastre[0000−0001−7741−3346]

Université d'Avignon, Avignon 84140, France
{firstname}.{lastname}@univ-avignon.fr

Abstract. Finding professional voice-actors for cultural productions is performed by a human operator and suffers from several difficulties. Researchers have therefore been interested for several years in mimicking the process of vocal casting to help human operators find new voices. However, voice casting appears to be an underdefined task with many difficulties. The main issue is that no label is available to accurately assess the performance of voice casting systems. To tackle these problems, recent works have focused on building a speech representation of acted voices able to highlight the character dimension. The proposed approach relies on an initial sequence extractor issued from a speaker recognition system which is able to represent a time variable speech sequence by a unique fixed-size vector, followed by a dedicated neural network where the character-based embedding, called *p*-vector, is extracted. It is legitimate to wonder if the sequence extractor is not guiding *p*-vectors too much towards speaker information. We then propose to study the impact of the speaker pre-training on the character representation learning. In comparison to a directly trained character representation, the results show that the use of a speaker pre-training provides more character information while retaining the speaker-independent part.

Keywords: voice casting · speaker recognition · character information · speaker-independent character information · speaker information.

1 Introduction

Voice Casting consists in finding professional voice-actors for cultural productions. It is carried out by a human operator and suffers from several difficulties. Indeed, the cinema market is evolving rapidly with the appearance of streaming platforms such as Netflix, Amazon Prime Video or Disney+. More and more audiovisual productions are emerging and it is becoming difficult to find the actors to dub them, and this, within increasingly tight deadlines (due in particular to the rise of series). Moreover, industrialized productions have a strong appetite for discovering new vocal talents. Nonetheless, experienced operators usually do not have the time to perform a large number of auditions, and they do not have

the memory to keep them all in mind. Of great interest is to come up with an automatic system capable of assisting them by selecting a small number of the best candidates from a large database for dubbing a specific voice. Some researchers have been investigating this question in recent years [6, 7, 16, 17] with the objective of proposing a system to measure vocal similarity, mimicking the decision of a professional operator.

But voice casting is a high level intellectual task difficult to automatize, due to its *underdefined* nature [1]: professional operators themselves find it difficult to precisely define the goals and fundamentals of voice casting, even though they do the job on a day-to-day basis. Working on this type of problem is also complicated because the only sure available knowledge on the vocal casting process concerns the choices previously made by the operators, that is to say the productions already dubbed. We would like to inform the reader that the articles [5–7, 15–17] cited in this paper are the only ones dealing with the speech-based automation of Vocal Casting in the literature to our knowledge.

In a recent experiment, authors from WarnerBros. proposed to evaluate a recommendation system by using decision data from Artistic Directors. These data are sensitive and their acquisition is not trivial since it requires working on the critical voice casting process. We therefore position ourselves in a task quite different from [15] since we do not use the Director Decisions — neither for training nor for evaluating our systems. We use the final works (video games) where we have the English and French voices from which we can deduce part of the decision criteria of the Artistic Directors in the form of what we call *character information*.

The work presented in this article follows [5–7] and shares with them the same strong hypothesis: if we do not know the precise criteria that go into the decisions of the operators, we assume that this process involves implicit, high level and commonly shared factors conveyed through the voice. These factors can be linked to the actors (physiology, voice, mode of play, etc.), to national culture and habits (perception of a given trait by a given audience at a given moment), to art (sensitivity and wishes of the original director) and the character played (type of character, state of mind, appearance, etc.). The second shared hypothesis is the fact that it is possible, even if the explicit analysis of these factors is (at least still) impracticable, to highlight their combined effects on the acoustic representation of speech. These combined effects are called "*character information*".

[5] proposes to build a learning-based voice representation based on the work in [26] dedicated to the "character information", denoted as p-vector. First, a *sequence extractor* transforms a time-variable speech sequence into a fixed length vector representing the general aspects of the vocal excerpt. Second, a specific neural network is trained using these vectors and is optimized for character-related tasks. The p-vectors are extracted from this second neural network [4, 8, 13, 19]. Finally, a Siamese neural network is used in the p-vector space in order to measure the character-oriented distance between two recordings. [5] has shown that it is possible, thanks to this approach, to link two audio files corresponding

to the same character, even if the vocal extracts are spoken by two different actors in two different languages (French and English here). They also showed that the training of the Siamese vocal similarity system at the level of the *p*-vector improves performance compared to its direct application on the outputs of the sequence extractor.

The *sequence extractor* used in the described approach comes from the speaker recognition domain. It requires a very large amount of training data coming from thousands of speakers, and it is optimized according to a speaker identification task. Its use in a character-based voice representation process seems compulsory because the annotated corpora available in the field are small in size. This sequence extractor process could be considered as "pre-training" [10, 20, 24] within the meaning of the transfer learning approach [10, 11, 14, 18, 25].

Although the use of a pre-trained *sequence extractor* clearly helps in building a character-oriented voice representation, it could also create biases in the system. If certain works [22, 23, 27] have already been interested in the information encoded by the embeddings, it seems however legitimate to wonder if this pre-training does not guide the *p*-vectors too strongly towards speaker information. This article is dedicated to this question and aims to verify the two following hypotheses:

- (A) The more a sequence extractor is dedicated to speaker recognition, the more it integrates specific high-level information about the speaker and risks losing information about the character himself. Thus, taking the embedding at a lower level in the sequence extractor neural network could help capture speaker less-specific information capable of better characterizing the character dimension.
- (B) Adapting parts of the sequence extraction model when learning the character-based representation itself could improve that representation and, in particular, its generalization capabilities.

Section 2 presents the character voice representation framework. The corpus and the details of the neural networks are presented in Section 3. Results of our experiments are presented for the first and the second hypothesis in Sections 4 and 5 respectively. For reproducibility reasons, scripts and models are available on GitHub[1]. Finally, we conclude by presenting some takeaways and possible directions for future work in Section 6.

2 Neural-network-based character voice representation

Figure 1 gives an overview of the character voice similarity framework used in this work. First, a *sequence extractor* outputs a fixed-length vector from a time variable speech excerpt. This output is consumed by the character representation module to generate *p*-vector. Then, the *decision* module takes as input a pair

[1] https://github.com/LIAvignon/specom2021-influence-of-speaker-pre-training-on-character-voice-representation

of p-vectors and generates a *score* about the character similarity between them. Finally, this score is compared to a decision threshold in order to obtain a binary decision.

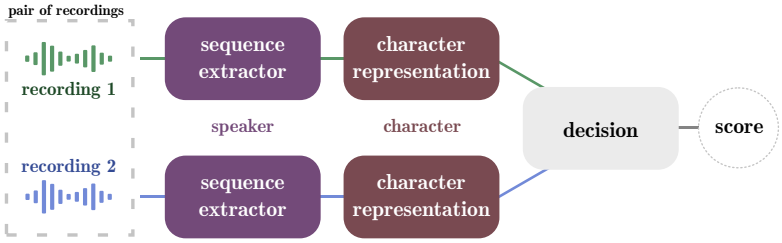

Fig. 1. Character voice similarity framework.

The *character representation* and the *decision* modules are presented in Sections 2.1 and 2.2 respectively. Finally, Section 2.3 focuses on the *sequence extractor* module and its links with the character voice representation system.

2.1 Character-oriented representation

The p-vectors were first introduced in [5]. They are built from a representation of speech signal and are intended to highlight character-related information in a given recording. To build p-vectors, a Multi-Layer-Perceptron (MLP) is trained to classify the character played by the given recording. Once the MLP is trained, the authors propose to use the last layer (before softmax) as an embedding, forming the p-vector of a voice extract. In this article, recordings are classified by the MLP among 16 or 12 characters for the protocols described in Sections 3.2 and 3.3.

2.2 Decision

In order to assess our character representation, we use the character similarity task and *decision* module defined in [6]. The task consists in deciding if two recordings spoken by two speakers belong to the same character or not (in this work, as voice dubbing is targeted, the two recordings are in different languages).

The *decision* module is based on the Siamese Neural Network [2, 9, 12]. It is composed of two layers (linear, fully-connected, with the hyperbolic tangent activation function) followed by a last neuron which calculates a score between 0 (the two inputs do not belong to the same class) and 1 (the two inputs belong to the same class). Siamese networks are known for their performance on this kind of tasks but also for their instability. To compensate the variation from one training to another, we train 10 systems and select the system with the highest accuracy on the validation set, during the training phase. At each training, only the initialization matrix changes.

2.3 Sequence extractor

The sequence extractor is based on the speaker recognition x-vector approach [20, 24], described in Figure 2. It can be decomposed into two parts: 1) the *extraction* part, where acoustic features are extracted from the signal, 2) the *classification* part, dedicated to the targeted task (here, speaker recognition). In this figure, *TDNN* stands for Time Delay Neural Network, as referred to in [24]. *SP* corresponds to a Statistical Pooling. The *XV* is based on a linear from which we classically extract x-vectors in the literature. This layer is followed by a Leaky RELU, a batchnorm, and a dropout. LIN is a linear (fully-connected) layer which precedes a softmax.

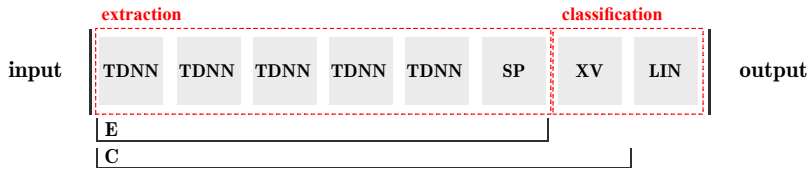

Fig. 2. x-vector extractor architecture.

[23] shown that the layers of such a network encode information with different levels of abstraction depending on the layer. Following hypothesis (A) from Section 1, we assume that the closer we get to the objective function (the output), the closer we are to the information dedicated to the speaker and to the risk of having a speaker bias in the character representation system.

In order to verify our hypothesis, we compare p-vectors issued from two different sequence extractors. One, denoted C, is a classical x-vector architecture when the other, denoted E, stops at an earlier layer, the statistical pooling layer. As synthesized in Figure 3, we use the C or E followed by a dense layer ($DENSE$) and an embedding layer from which we extract the p-vectors (PV). E and C represent the parts pre-trained using a speaker recognition objective and a large dedicated corpus (*VoxCeleb2* [3]). $DENSE$ and PV layers are always trained using a character-oriented objective and a dedicated corpus (*Mass-Effect 3*, see 3.1).

Fig. 3. p-vector extractor architecture. The left part (E or C) corresponds to the sequence extractor while the right part (DENSE and PV) corresponds to the character representation of Figure 1.

3 Experimental Protocol

We present the data in Section 3.1 and how we split them in Sections 3.2 and 3.3. We then detail the x-vector pre-training and the p-vector training in Sections 3.4 and 3.5 respectively. Finally, we describe the evaluation in Section 3.6.

3.1 Corpora

The main corpus is composed of voice recordings coming from the *Mass-Effect 3* role-playing game. Contrary to movies, voices of video games present some particularities explained in [6] (radio effects are present in the original voices of our corpus). They are also easier to collect since they are separated from the ambient sounds even in their final form, contrary to the dubbed film archives. Originally released in English, the game has been translated and revoiced in several other languages. In our experiments, we use the English and French versions of the audio sequences, representing about 7.5 hours of speech in each language. Segments (or recordings) are 3.5 seconds long on average. A character is then defined by a unique French-English couple of two distinct speakers. To avoid speaker identity biases, we consider only a small subset where we are certain that none of the actors play more than one character. A single audio segment corresponds to a unique speaking slot from an actor in a particular language. We then apply a filter that keeps only recordings longer than 1 second. Finally, we only keep the 16 characters for which we have the largest number of recordings, as [6] did.

3.2 Highlighting the speaker-independent character information

The *original* protocol is based on a closed space protocol where we keep the 16 characters, and their 32 corresponding speakers (French and English), for both training and testing phases. We split the corpora into three subsets: training (*train*), validation (*val*), and test (*test*) using a 80/10/10 *rule*. All these subsets are composed of different recordings, but coming from the same 16 characters (and so 32 speakers, including both languages). To build the *train*, *val* and *test* subsets, we randomly select for each character 144, 18, and 18 recordings respectively, while balancing the number of French and English recordings. We then have a total of $2,304$ (*train*), 288 (*val*) and 288 (*test*) recordings. We name S_o the *train*, *val* and *test* sets deriving from this protocol.

The dataset is composed of only characters played by strictly different actors in each language. The trained systems can therefore learn to associate speaker identities without taking into account the character particularities. To verify that the systems are not too much skewed by this training configuration, we propose a *modified* protocol. This protocol aims at neutralizing the character information by modifying associations between actors, while respecting some constraints to avoid bias (like gender). In other words, the actors (original or dubber) are no longer assigned to the same character and are associated with a new actor (dubber or original). By comparing the absolute difference between

scores obtained on the *original* and *modified* protocols, we can then highlight the Speaker-Independent Character Information (SICI) captured by the representation. The presence or absence of such information is a clue to identifying a potential bias in the system. We name S_m the *train*, *val* and *test* sets deriving from this protocol.

3.3 Checking the generalization abilities of the character representation

In order to measure the ability of our systems to generalize, we propose a protocol that breaks down data differently. This time, we only keep 12 characters, and their 24 corresponding speakers, for the training phase and we use the remaining 4 characters for the testing phase. We still break the corpus down into training, validation and test sets. The *train* and *val* sets are composed of different recordings, but coming from the same 12 characters. To build the *train*, *val* and *test* subsets, we randomly select for each character 144, 36 and 180 recordings respectively, while balancing the number of French and English recordings. We then have a total of $1,728$ (*train*), 432 (*val*) and 720 (*test*) recordings. We name the three sets S_n.

3.4 Pre-training of *x*-vector model

The x-vector model is trained on the VoxCeleb 2 corpus. The layer XV from which we extract the x-vectors is accompanied by a LEAKY RELU activation function and a batch normalisation. The last linear layer (*i.e.* LIN in Figure 2) is attached to a logarithmic softmax at the output. The cost function is the cross-entropy loss.

MFCCs are used as input, extracted using Kaldi tools [21] with the following parameters: 30 cepstral coefficients, 25 ms frame length, $20-7,600$ Hz bandwidth.

3.5 Training of *p*-vector neural network

As in [5], the *DENSE* (see Figure 3) is composed of a linear layer with hyperbolic tangent activation function and a dropout of 0.25. The p-vector layer (*i.e.* PV in Figure 3) is composed of the same elements except for the dropout whose value is 0.5. We finally compute a logarithmic softmax at the output of the network. The cost function we have used to train the network is the cross-entropy loss.

3.6 Evaluation

As shown in Section 2.2, we evaluate p-vectors on a character voice similarity task using a character-similarity measure based on a Siamese network. We generate *Target* trials composed of pairs of recordings belonging to the same character and *Non-target* trials made up of recordings belonging to two different characters. To avoid any bias, the number of *targets* and *non-targets* is balanced, as well as the

number of pairs between two actors. For sets S_o and S_m, we generate $165,888$ (*train*), $2,592$ (*val*) and $2,592$ (*test*) *trials*. For set S_n, we generate $124,416$ (*train*), $7,776$ (*val*) and $64,800$ (*test*) *trials*. The threshold is set *a posteriori* at the Equal Error Rate (EER) point. System performance is then expressed in terms of accuracy on the test.

4 Reduce the speaker discriminative information

In order to verify hypothesis (A) from Section 1, we propose to reduce the speaker discriminating power of the sequence extractor by using configuration E (extract the sequence vector at the statistical pooling level) instead of configuration C (full classical x-vector extractor).

First, we evaluate the loss in terms of speaker discrimination when using E versus C (see Figure 2). We measure an EER of 22% (computed on VoxCeleb1) using E, to be compared with an EER of 6% for the configuration C. This result validates the first part of our hypothesis: taking the embedding at a lower level decreases its speaker discriminating power.

We then verify the impact of this decrease on the character representation. For that purpose, we build two p-vector representations, using the outputs of the networks C (*config 1*) and E (*config 2*) respectively. In both cases, we freeze speaker pre-trained layers (C or E) during p-vector training. We then train our two network configurations using the closed space protocol.

We perform a comparative experiment using three data configurations (*original*, *modified* and *mixed*) and the two sequence extractors already presented (*config 1* and *config 2*). In *original*, p-vectors (character) and the voice similarity system (decision) are trained on the original data S_o (true character-speaker tying). In *modified*, both modules are trained on modified data S_m (artificial tying, but consistent between character and decision levels). In *mixed*, p-vectors are trained on original data S_o while the decision system is using the S_m tying.

We assume that the difference in terms of accuracy between *original* and *modified* is a direct evidence of the amount of character information independent to the speaker embedded in the p-vector representation. We also assume that the accuracy obtained using the *mixed* configuration is a way of measuring the amount of speaker specific information in the p-vector.

Table 1 presents the results of this set of experiments. We observe an absolute difference in accuracy of 3.2 points between *original* and *modified* configurations in the case of *config 1* and 1.7 points in the case of *config 2*. It shows that p-vectors embed speaker-independent character information. Also, the amount of this character information is lower when a sequence extractor with less speaker discriminant power (*config 2*) is used.

The *mixed* protocol shows a high general level and a stronger presence of speaker information for *config 2* than for *config 1*.

	character module	decision module	performance config 1	performance config 2
original	S_o	S_o	92.3	95.7
modified	S_m	S_m	89.1	94.0
mixed	S_o	S_m	79.7	81.8

Table 1. Accuracy of *config 1* (using C) and *2* (using E) on the test with original, modified and mixed protocol

5 Give more power to the character classification

This section sets out to test hypothesis (B) from Section 1. We propose to give more power to the character classification system during the construction of p-vectors by reducing the influence of pre-training from the sequence extractor. For this purpose, we train p-vectors following two types of configurations presented in Table 2. In configuration 3, we do not freeze the E layers to give more power to the p-vector classification network. In configuration 4, we give all the power to the p-vectors by completely removing pre-training.

config	layers	pre-training	freezing
3	E	VoxCeleb 2	$\times$
4	E	$\times$	$\times$

Table 2. Training configuration for config 3 and 4 systems.

Table 3 presents the results of the character similarity task obtained on the *config 3* and *4*. Although the results are also high ($\geq$ 90%), we observe less difference in terms of accuracy between the *original* and the *modified* training configurations. We have 0.5 point and -0.8 point difference for *config 3* and *4* respectively. From these results, and compared to those obtained with *config 1*, we observe an evaporation of character information independent to the speaker when we give more power to the character classification.

	character module	decision module	performance config 3	performance config 4
original	S_o	S_o	95.6	93.6
modified	S_m	S_m	95.1	94.4
mixed	S_o	S_m	81.5	82.1

Table 3. Results obtained in accuracy computed on the test with original, modified and mixed protocol using the *config 3* and *4*.

Table 4 shows the number of modified parameters during training step (*learnable parameters*). It also shows the total number of systems parameters where frozen parameters are taken into account. Since *config 3* and *4* have a higher number of parameters to learn than *config 1* and *2*, we propose to evaluate a smaller system without pre-training having then 664, 144 parameters. We name this system *small*. We obtain accuracies of 92.5 and 89.5 on the *original* and *modified* protocols respectively. This difference of 3 points suggests that reducing the number of parameters brings to light more character information in the p-vector representation.

system	learnable params	total params
config 1	296 528	4 523 492
config 2	1 570 384	4 260 836
config 3	4 260 836	4 260 836
config 4	4 260 836	4 260 836
small	664 144	664 144

Table 4. Number of parameters and learnable parameters for each studied system configuration.

Finally, we seek to verify the generalization abilities of our different system configurations on characters unseen in our training data (S_n protocol). The results of this protocol are shown in Table 5. The systems configured with the *config 1* and *2* respectively obtained a 68.5% and 68.6% accuracy in terms of character similarity. This result strongly suggests that decreasing the amount of speaker discriminant information in the sequence extractor module (*config 2*) does not degrade the generalization abilities of the character representation (*config 1*). As expected, these experiments also tend to demonstrate that the use of a speaker pre-training helps the character representation (p-vectors) to generalize to unseen characters, as the *config 3*, *config 4* and *small* (where speaker pre-training is partially or completely removed) obtained lower accuracies than *config 1* and *2* (with speaker pre-training).

	GP	SI	SICI
config 1	68.5	79.7	3.2
config 2	68.6	81.8	1.7
config 3	61.2	81.5	0.5
config 4	61.1	82.1	−0.8
small	66.9	80.8	3.0

Table 5. Speaker-Independant Character Information (SICI), Speaker Information (SI) and Generalization Power (GP) per system. SICI is the accuracy difference between original and modified protocols, SI is the accuracy obtained using the mixed protocol and GP is the accuracy computed using the generalization protocol (S_n).

6 Conclusion

This article investigated the influence of speaker pre-training on character voice representation. A first experiment showed that choosing a less speaker discriminative representation for the sequence extractor weakens the presence of speaker-independent character information. A second experiment showed that partially or completely removing speaker pre-training does not make it possible to encode more character information or even to better preserve its part independent to the speaker. We can conclude that the pre-training of the sequence extractor does not guide the p-vectors too strongly towards speaker information and brings useful knowledge to the construction of a character representation. Based on our experimental results, the representation of the characters appear as essentially inseparable from the notion of speaker. We now wish to direct our future work towards the discovery of an "alphabet" of character attributes clustering short voice segments into the p-vector space.

7 Acknowledgements

This project is supported by the French National Research Agency (ANR) "TheVoice" grant (ANR-17-CE23-0025).

References

1. Bonastre, J.F.: Representation learning for underdefined tasks. In: Nyström, I., Hernández Heredia, Y., Milián Núñez, V. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. pp. 42–47. Springer International Publishing, Cham (2019)
2. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 539–546 vol. 1 (2005)
3. Chung, J.S., Nagrani, A., Zisserman, A.: Voxceleb2: Deep speaker recognition. In: Interspeech (2018)
4. Chung, Y.A., Glass, J.: Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech (2018)
5. Gresse, A., Quillot, M., Dufour, R., Bonastre, J.F.: Learning Voice Representation Using Knowledge Distillation For Automatic Voice Casting. In: Interspeech (2020)
6. Gresse, A., Quillot, M., Dufour, R., Labatut, V., Bonastre, J.F.: Similarity metric based on siamese neural networks for voice casting. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2019)
7. Gresse, A., Rouvier, M., Dufour, R., Labatut, V., Bonastre, J.F.: Acoustic pairing of original and dubbed voices in the context of video game localization. In: Interspeech (2017)
8. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742 (2006)

9. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742 (2006)
10. Hendrycks, D., Lee, K., Mazeika, M.: Using pre-training can improve model robustness and uncertainty. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. vol. 97, pp. 2712–2721. PMLR (2019)
11. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015)
12. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop. vol. 2. Lille (2015)
13. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. Advances in neural information processing systems **27**, 2177–2185 (2014)
14. Lopez-Paz, D., Bottou, L., Schölkopf, B., Vapnik, V.: Unifying distillation and privileged information (2016)
15. Malik, A., Nguyen, H.: Exploring automated voice casting for content localization using deep learning. SMPTE Motion Imaging Journal **130**(3), 12–18 (2021)
16. Obin, N., Roebel, A.: Similarity search of acted voices for automatic voice casting. IEEE/ACM Transactions on Audio, Speech, and Language Processing **24**, 1642–1651 (2016)
17. Obin, N., Roebel, A., Bachman, G.: On automatic voice casting for expressive speech: Speaker recognition vs. speech classification. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2014)
18. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)
19. Pascual, S., Ravanelli, M., Serrà, J., Bonafonte, A., Bengio, Y.: Learning problem-agnostic speech representations from multiple self-supervised tasks (2019)
20. Peddinti, V., Povey, D., Khudanpur, S.: A time delay neural network architecture for efficient modeling of long temporal contexts. In: Interspeech (2015)
21. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K.: The kaldi speech recognition toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society (2011), iEEE Catalog No.: CFP11SRW-USB
22. Raj, D., Snyder, D., Povey, D., Khudanpur, S.: Probing the information encoded in x-vectors. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). pp. 726–733. IEEE (2019)
23. Rouvier, M., Favre, B.: Investigation of speaker embeddings for cross-show speaker diarization. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5585–5589. IEEE (2016)
24. Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S.: X-vectors: Robust dnn embeddings for speaker recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5329–5333 (2018)
25. Vapnik, V., Izmailov, R.: Learning using privileged information: Similarity control and knowledge transfer. J. Mach. Learn. Res. **16**(1), 2023–2049 (2015)
26. Variani, E., Lei, X., McDermott, E., Moreno, I.L., Gonzalez-Dominguez, J.: Deep neural networks for small footprint text-dependent speaker verification. In: ICASSP. pp. 4052–4056 (2014)
27. Wang, S., Qian, Y., Yu, K.: What does the speaker embedding encode? In: Interspeech. pp. 1497–1501 (2017)