Continual Representation Learning in Written and Spoken Language

Juan Manuel Coria

université PARIS-SACLAY

Continual Representation Learning in Written and Spoken Language

Apprentissage de représentation en continu pour la langue écrite et parlée

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: informatique
Graduate School : Informatique et sciences du numérique, Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche **LISN** (Université Paris-Saclay, CNRS), sous la direction de
Sophie ROSSET, Directrice de Recherche CNRS,
et le co-encadrement de
Hervé BREDIN, Chargé de Recherche CNRS,
Sahar GHANNAY, Maîtresse de Conférences

Thèse soutenue à Paris-Saclay, le 5 avril 2023, par

Juan Manuel CORIA

Composition du jury

Membres du jury avec voix délibérative

Slim Essid Professeur, Télécom Paris	Président
Richard Dufour Professeur des Universités, Université de Nantes, LS2N	Rapporteur & Examinateur
Corinne Fredouille Professeure des Universités, LIA, Université d'Avignon et des pays du Vaucluse	Rapporteur & Examinatrice
Géraldine Damnati Ingénieure-Chercheuse, Orange Labs	Examinatrice
Benoit Favre Professeur des Universités, Aix Marseilles Université, LIS	Examinateur

THESE DE DOCTORAT
NNT : 2023UPASG025

"People think of education as something they can finish."

— Isaac Asimov

Acknowledgements

The path of the PhD is a strenuous one. This has been a challenging experience, but also an incredibly rewarding one, both professionally and personally. Here, I would like to take the time to thank the people that have made this possible.

First of all, I could not have undertaken this journey without the support of my family, and especially my parents Gabriel and Patricia, who have been there for me my entire life and who have always encouraged me to pursue higher education and science. I also want to thank my brother Matías and my sister Florencia, who know how to make their presence felt despite the distance that separates us.

I would like to express my deepest gratitude to my supervisors Sophie, Hervé and Sahar, whose complementary expertise, constant guidance and incredible pedagogy allowed me to carry on with ease and confidence. I consider myself extremely lucky to have had the chance to work with you, and the quality of my work would have never been the same without your outstanding supervision.

I also want to thank the people that have made my life at LISN unforgettable. I want to thank (in no particular order) Mathilde, Nicolas, Paul, Nesrine, Camille, Natalia, Alban, Marco, Hugo, Hicham, François, Lisa, Atilla, Thomas, Paritosh, Shu, Armand, Mathieu, Remi, Hugues, as well as all the people that I have met there and with whom I have shared a beer or even a short discussion. I am also incredibly grateful to Sofiya, who accompanied me through challenging times, and who knew how to calm me when I thought I was going to fail. Thank you all for laughing at my bad jokes and for making this one of the most beautiful experiences of my life. A single page would not be enough to thank all of you.

Finally, I would like to acknowledge the funding of Université Paris-Saclay under PhD contract number 2019-089. My work has also been supported by the French National Research Agency (ANR) via the funding of the LILITH (ANR-17-CHR2-0001-03) and PLUMCOT (ANR-16-CE92-0025) projects, and was granted access to the Saclay-IA computing platform as well as the HPC resources from GENCI-IDRIS under allocations AD011011182, AD011012609R1 and AD011012177R1.

Résumé

Bien que l'apprentissage automatique ait récemment connu des avancées majeures, les modèles actuels sont le plus souvent entraînés une fois pour toutes sur une tâche cible, puis déployés dans l'environnement de production, et leurs paramètres sont rarement (voire jamais) révisés. Cette approche affecte les performances dans le nouvel environnement, car les données et les spécifications de la tâche peuvent évoluer avec le temps et les besoins des utilisateurs. L'apprentissage continu propose une solution en entraînant des modèles au fil du temps, à mesure que de nouvelles données sont disponibles. Cependant, il souffre d'un phénomène appelé "oubli catastrophique", qui dénote une perte de performance significative sur des exemples déjà vus. De nombreuses études ont proposé différentes stratégies pour contrer ce phénomène, mais la plupart des algorithmes reposent sur des données étiquetées rarement disponibles en pratique. En revanche, l'adaptation continue d'un modèle pré-entraîné en production, où les données sont généralement non étiquetées et acquises au fil du temps, est un problème moins étudié.

Dans cette thèse, nous étudions l'adaptation continue pour les applications de traitement de la langue écrite et parlée. Notre objectif principal est de concevoir des systèmes autonomes et auto-apprenants capables d'exploiter les données disponibles sur le terrain, afin de s'adapter aux environnements de production. Pour ce faire, nous proposons d'exploiter des représentations adaptées à la tâche cible. Cela contraste fortement avec les travaux récents sur le pré-entraînement auto-supervisé, dont l'objectif est d'apprendre des représentations à usage général telles que les plongements lexicaux. Nous pensons que les représentations adaptées à la tâche sont plus faciles à interpréter et à manipuler par des algorithmes non supervisés comme le partitionnement (ou "clustering" en anglais), qui sont moins affectés par l'oubli.

Nous étudions d'abord l'apprentissage de ces représentations pour les tâches à ensemble ouvert (pour la généralisation à de nouvelles classes) et les tâches à ensemble fermé (pour la généralisation à de nouvelles instances des mêmes classes d'apprentissage). Nous concluons que cette approche est mieux adaptée aux tâches

à ensemble ouvert, ce qui oriente nos recherches ultérieures dans deux directions.

D'une part, nous cherchons à mieux comprendre le transfert dans des scénarios à ensemble fermé en étudiant l'adaptation supervisé et en continu des plongements lexicaux contextuelles à de nouvelles langues, dans des tâches d'étiquetage de séquences. Bien que l'oubli soit présent, nous découvrons qu'il existe un niveau élevé de transfert en avant : des langues passées vers les langues futures.

D'autre part, nous étudions l'utilisation des représentations du locuteur pour la segmentation et le regroupement en locuteurs en flux, visant à déterminer "qui parle quand" dans une conversation. Ce problème est non supervisé et à ensemble ouvert, car de nouveaux locuteurs peuvent apparaître à tout moment. Dans ce contexte, s'appuyer sur le partitionnement des représentations nous permet de concevoir un système entièrement autonome et auto-apprenant qui ne nécessite que peu ou pas d'intervention humaine experte. Puisque ce système repose sur un modèle pré-entraîné, nous proposons également une méthode d'adaptation au domaine en continu, avec laquelle il peut être progressivement adapté aux nouvelles conversations du domaine cible.

Dans l'ensemble, nous pensons que notre travail, effectué dans une variété de scénarios liés à la langue, constitue un pas important vers l'apprentissage autonome en continu dans la phase de production. En particulier, nous améliorons notre compréhension du transfert en avant et ouvrons la discussion sur l'utilisation de représentations spécifiques à une tâche cible en montrant leur efficacité dans la segmentation et regroupement en locuteurs en flux.

Contents

1 Introduction **19**
- 1.1 Context . . . 19
- 1.2 Motivation . . . 22
- 1.3 Objective and contributions . . . 23
- 1.4 Publications . . . 24

2 Related work **27**
- 2.1 From hand-crafted to neural representations . . . 28
- 2.2 Transfer learning . . . 30
- 2.3 Metric learning . . . 34
- 2.4 Continual learning . . . 36
  - 2.4.1 Forgetting and transfer . . . 38
  - 2.4.2 Memory-based methods . . . 40
  - 2.4.3 Constraint-based methods . . . 41
  - 2.4.4 Modular methods . . . 43
  - 2.4.5 Meta-learning . . . 45
- 2.5 Conclusion . . . 47

3 The quest for task-specific representations **49**
- 3.1 Introduction . . . 50
- 3.2 Metric learning loss functions . . . 52
  - 3.2.1 Classification-based . . . 52
  - 3.2.2 Contrast-based . . . 54
- 3.3 Speaker verification experiments . . . 58
  - 3.3.1 The speaker verification task . . . 58
  - 3.3.2 The VoxCeleb corpora . . . 60
  - 3.3.3 Model architecture . . . 61
  - 3.3.4 Evaluation . . . 63
  - 3.3.5 Training protocol . . . 63
  - 3.3.6 Implementation details . . . 64

3.3.7 Results and discussion . . . 65
3.4 Misogyny categorization experiments . . . 67
3.4.1 The misogyny categorization task . . . 67
3.4.2 The AMI corpus . . . 68
3.4.3 Model architecture . . . 69
3.4.4 Training protocol . . . 69
3.4.5 Implementation details . . . 72
3.4.6 Results and discussion . . . 72
3.5 Conclusion . . . 74

4 Continual word embedding adaptation 75
4.1 Introduction . . . 76
4.2 Sequence labeling . . . 79
4.2.1 The MultiATIS++ corpus . . . 80
4.2.2 The MultiCoNER corpus . . . 81
4.3 Model architecture and training . . . 83
4.4 Measures of transfer . . . 84
4.5 Cross-lingual transfer . . . 85
4.5.1 Joint transfer . . . 85
4.5.2 Continual transfer . . . 88
4.6 Effect of the training sequence . . . 91
4.7 Performance recovery . . . 95
4.8 Conclusion . . . 97

5 Autonomous streaming speaker diarization 99
5.1 Introduction . . . 100
5.2 Related work . . . 102
5.2.1 Multi-stage approaches . . . 102
5.2.2 End-to-end approaches . . . 102
5.2.3 Online speaker diarization . . . 105
5.2.4 Evaluation . . . 105
5.3 Building blocks . . . 106
5.3.1 Speaker segmentation . . . 107
5.3.2 Overlap-aware speaker embedding . . . 108
5.4 Proposed continual learning approach . . . 110
5.4.1 Constrained incremental clustering . . . 110
5.4.2 Detecting new speakers and updating centroids . . . 112
5.4.3 Latency adjustment . . . 114
5.4.4 Experiments . . . 115
5.4.5 Results and discussion . . . 120
5.5 Conclusion . . . 123

6 Continual self-adaptation for speaker segmentation **125**
6.1 Introduction . . . 126
6.1.1 Removing the need for manual annotation . . . 127
6.1.2 Continual and ephemeral learning . . . 128
6.2 Self-supervised adaptation . . . 129
6.2.1 Related work . . . 129
6.2.2 Proposed approach . . . 130
6.2.3 Experimental protocol . . . 133
6.2.4 Results and discussion . . . 138
6.3 Continual learning extension . . . 139
6.3.1 Proposed approach . . . 139
6.3.2 Experimental protocol . . . 141
6.3.3 Results and discussion . . . 142
6.4 Conclusion . . . 146

7 Conclusion **149**
7.1 Summary . . . 149
7.2 Reproducible research . . . 151
7.3 Contributions . . . 154
7.4 Future directions . . . 156
7.4.1 Continual learning with task-specific representations . . . 156
7.4.2 Better understanding of forward transfer . . . 158

List of Figures

1.1 Single-stage, transfer and continual learning schemes 21
1.2 Scientific contributions of this work 23

2.1 Comparison between hand-crafted and neural representations 29
2.2 Parameter shifts in joint and sequential transfer learning 31
2.3 Complexity of transfer and continual learning 33
2.4 The metric learning approach 35
2.5 Metric learning representations in two dimensions 36
2.6 Performance matrix commonly used for continual learning evaluation 38
2.7 Existing strategies to prevent catastrophic forgetting 42
2.8 Block modular architecture 44
2.9 The teacher-student approach of knowledge distillation 45
2.10 Comparison between meta-learning methods 47

3.1 The metric learning approach 51
3.2 The congenerous cosine loss 53
3.3 The additive angular margin loss 54
3.4 Comparison between congenerous cosine loss and center loss 55
3.5 The contrastive loss 55
3.6 Contrastive loss behavior 56
3.7 The triplet loss 57
3.8 Easy and hard embedding pairs in two dimensions 57
3.9 Comparison between speaker identification, speaker verification and speaker diarization 59
3.10 End-to-end speaker verification model 62
3.11 Detection error trade-off curves 63
3.12 Speaker verification results 65
3.13 The misogyny categorization task 67
3.14 Misogyny categorization models 70
3.15 Misogyny categorization results 72

4.1 Transfer styles present in large multilingual language models 77
4.2 A training sequence across 4 languages 79
4.3 The IOB labeling scheme 80
4.4 A labeled MultiATIS++ sentence 81
4.5 MultiCoNER sentences 82
4.6 Sequence labeling model 83
4.7 Performance matrix used to measure backward and forward transfer 84
4.8 A training sequence of 4 languages 87
4.9 Per-language sequence sampling algorithm 89
4.10 Evaluation focus in forward transfer 89
4.11 Evaluation focus in backward transfer 90
4.12 Anatomy of a box plot 92
4.13 Forward transfer at different positions in the training sequence . . . 93
4.14 First-language backward transfer for different sequence lengths . . . 93
4.15 The progressive parameter shift hypothesis 94

5.1 Online speaker diarization 100
5.2 Multi-stage approach to speaker diarization 103
5.3 Speaker segmentation model 104
5.4 The diarization error rate 106
5.5 Proposed online speaker diarization approach 107
5.6 Binarization of local segmentation output 108
5.7 End-to-end speaker embedding model 109
5.8 Overlap-aware weights for speaker embedding extraction 110
5.9 Overlap-aware speaker embedding extraction 111
5.10 Consecutive but inconsistent segmentation model outputs 112
5.11 Speaker recognition in ncremental clustering 113
5.12 Latency and aggregation in local segmentation predictions 115
5.13 Impact of the chosen latency 121
5.14 Evolution of performance as conversations unfold 122
5.15 Speaker number estimation performance 123

6.1 Domain mismatch and domain adaptation 127
6.2 Teacher-student training scheme 129
6.3 Segmentation model defined in Chapter 5 131
6.4 Non-continual self-supervised training scheme 131
6.5 Early stopping based on the AUROC metric 132
6.6 The AUROC metric 133
6.7 Evaluation per domain and hyper-parameter configuration 135
6.8 Evaluation of the best hyper-parameter configuration 136
6.9 The chunk-wise diarization error rate (CDER) 137

6.10 Continual training with pseudo-labels over a sequence of target-domain conversations 140
6.11 Partitioning of a conversation into training and validation sets . . . 141
6.12 Performance across the conversation sequence 144
6.13 Performance matrix rows chosen to investigate continual transfer . . 145
6.14 Performance of selected conversations across the training sequence . 146

List of Tables

3.1 Description of VoxCeleb corpora 60
3.2 Speaker verification convergence times 66
3.3 Best hyper-parameters for speaker verification 66
3.4 Misogyny categories in the AMI corpus 68
3.5 Misogyny categories in the AMI corpus 69
3.6 Hyper-parameter search space in misogyny categorization experiments 71
3.7 Best loss hyper-parameters for misogyny categorization 73

4.1 Description of the MultiATIS++ corpus 81
4.2 Description of the MultiCoNER corpus 82
4.3 Monolingual and multilingual performance on MultiATIS++ 86
4.4 Monolingual and multilingual performance on MultiCoNER 86
4.5 Costs of learning a new language according 87
4.6 Continual performance on MultiATIS++ 91
4.7 Continual performance on MultiCoNER 91
4.8 Fast recovery results on MultiATIS++ 95
4.9 Fast recovery results on MultiCoNER 96

5.1 Description of DIHARD III domains 116
5.2 Description of the DIHARD III corpus 117
5.3 Description of the VoxConverse corpus 117
5.4 Description of the AMI meeting corpus 118
5.5 Performance of our online speaker diarization system 120

6.1 Description of DIHARD III domains 134
6.2 DIHARD III domain partitioning and details 135
6.3 Hyper-parameter search space 138
6.4 Performance of the non-continual self-supervised approach 138
6.5 Performance of the continual pseudo-labeling method 142

Chapter 1

Introduction

"Everything in human life was a test. That was why they all looked so stressed out."
— Matt Haig, *The Humans*

1.1 Context

In recent years, machine learning has witnessed major breakthroughs that have dramatically changed the field. Today, models can accurately identify objects in scenes, transcribe speech, describe images or write stories, but despite these enormous advances, many abilities that we consider marks of human intelligence still seem elusive. In particular, one of the key missing pieces is the ability to learn from and adapt to changing environments.

The lifecycle of a typical machine learning model can be divided in two phases: the *development* phase where the model is trained on previously collected (and generally labeled) data, and the *production* phase where the trained model is put to work on new unseen examples. The event that marks the transition from the development to the production phase is typically referred to as the *deployment* of the model. In the development phase, the typical approach consists of a *single-stage* training, where the model is trained once on a target task and then deployed, rarely (if ever) revisiting or updating its parameters. Unfortunately, the performance of a model after deployment usually worsens with time, as the data distribution, task specifications, and even user needs may gradually change with respect to those present in the training stage. In contrast, *continual learning*, sometimes also called life-long learning, studies model training over time as new data becomes available in a sequence of training stages, which solves the lack of flexibility of

single-stage learning. However, a simple approach where the data for each training stage keeps growing with past examples is normally considered unacceptable, as time and space complexity increases linearly with the (possibly infinite) amount of training stages. As a result, continual training schemes are also typically defined by a limited or forbidden access to previously seen examples, which causes a phenomenon called *catastrophic forgetting* (French, 1999) that is defined as significant performance loss on previously seen examples. It has been hypothesized that the dynamics of gradient-based learning are to blame for this (Hadsell et al., 2020), as the greedy gradient descent algorithm shifts model parameters towards a (hopefully) optimal solution for the current set of training examples, completely erasing the optimal solution for previous training stages. Although a wide variety of strategies to avoid forgetting have been proposed over the years (Kirkpatrick et al., 2017; Li and Hoiem, 2016; Zenke et al., 2017; Javed and White, 2019), most algorithms still rely on labeled data, which is rarely available in practice.

In stark contrast to this, *on-the-job* or *production* data (*i.e.* the examples on which the model is applied during the production phase) is not only characterized by new examples appearing over time, but also by their lack of annotations. To make matters worse, on-the-job data can also be scarce if the production environment is overly specific. While often ignored in continual learning research (Kirkpatrick et al., 2017; Li and Hoiem, 2016; Lopez-Paz and Ranzato, 2017), unsupervised low-resource methods are essential to address both the lack of annotations and data scarcity in the production phase. The field of *active learning* (Settles, 2009) provides a partial solution by giving models the ability to interact with a human in order to obtain labels for production data. However, it introduces the additional problem of overhead: the model must wait for the human to provide answers, which can be costly and time-consuming. Consequently, in order to avoid these pitfalls, model functioning must be free from any expert human intervention (*i.e.* autonomous), but also capable of learning from unlabeled data, as well as from its own interactions with the environment.

Crucially, initial training in the development phase can also play a key role in preparing a model for effective learning in new environments. The field of *transfer learning* (Pan and Yang, 2010) investigates ways of training general-purpose models for easy reuse, which typically consist of two stages. The first *pre-training* stage trains a model from scratch on a large corpus with a self-supervised objective, such as predicting masked words or the next few seconds of speech. The goal of this stage is to obtain a general-purpose model that can be adapted to many target tasks with less training data. Thus, the second *fine-tuning* stage trains the resulting model again on a more specific (usually supervised) target task, after which the model is normally deployed. This paradigm has recently led to the popularity

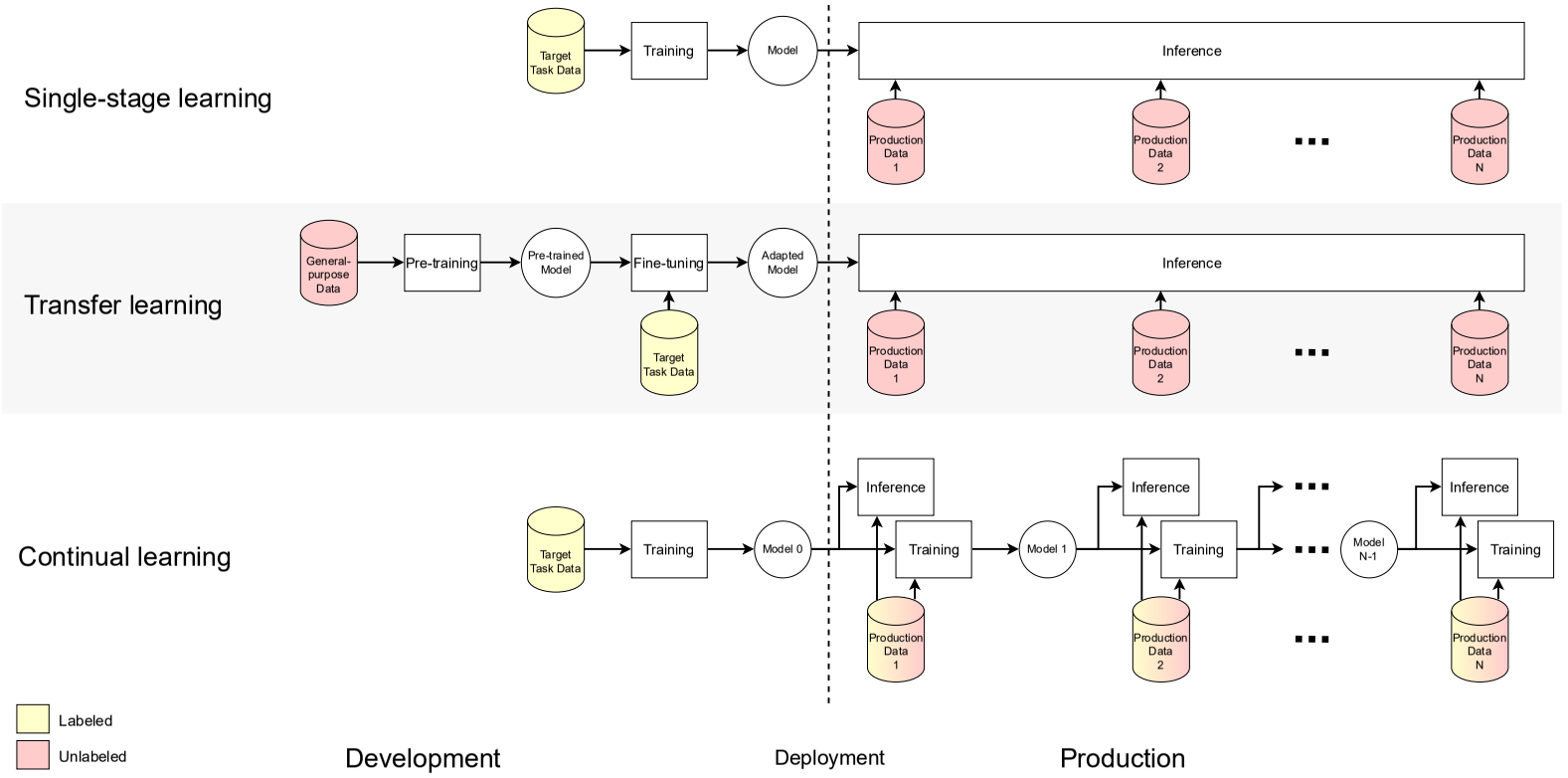


Figure 1.1: Single-stage, transfer and continual learning schemes divided into development and production phases. While single-stage and transfer learning produce a single model that is never revised after deployment, continual learning allows training to go on indefinitely.

of pre-training general-purpose vector representations of words (Peters et al., 2018; Devlin et al., 2019) and even speech (Chung and Glass, 2020; Pascual et al., 2019), also known as *embeddings*. A depiction of both the development and production phases of single-stage, transfer and continual learning is shown in Figure 1.1.

Contrary to self-supervised embeddings, *metric learning* (Kaya and Bilge, 2019) proposes to learn representations that can be compared with a simple distance function, facilitating their interpretation and exploitation in unsupervised distance-based algorithms such as clustering. Instead of being general-purpose, metric learning embeddings are bound to a downstream task, as the distance between representations is directly mapped to a key property of the underlying data (*e.g.* speaker similarity when comparing speech). However, contrary to general-purpose embeddings that heavily rely on the supervised fine-tuning stage, the manipulation of representations based on distances does not rely on gradient-based learning. Hence, it is reasonable to assume that this would be less prone to forgetting, which is precisely why they can emerge as an attractive way to address continual learning in the production phase.

1.2 Motivation

Machine learning applications for written and spoken language are becoming ubiquitous, ranging from dialogue systems that help users navigate the information on a website through natural language, to automatic meeting transcription from live speech. These applications seem a natural candidate for continual learning after model deployment, as a dialogue system may learn from new types of questions, or a real-time meeting transcription system could adapt to the accent of a given speaker, or even the low quality of their microphone. Unfortunately, applications like these are rarely studied in the continual learning literature, and simpler settings from computer vision are usually preferred. As a consequence, it is hard to say if such systems are effective in significantly different language-related applications. As an example, consider two scenarios. The first one being an image recognition model for different animal species, where each training stage introduces new classes. The second one being a model that identifies speakers in a live conversation, where new speakers can appear at any point in time. Given the task definition, it is unclear if a continual learning system for the first case can also be effective for the second, even if both deal with scarce and unlabeled production data.

After an initial study on task-specific representations, our work branches into two main research directions with their own target tasks. First, our study on written language focuses on sequence labeling, which consists of predicting a class per

word in a given sentence. This task can be helpful in a variety of scenarios, such as detecting entities (*e.g.* names of people or organizations) or even the type of information queried in a dialogue system. Second, we study spoken language through the lens of speaker diarization, which consists of determining "who speaks when" in a recorded conversation. This task is particularly useful to enhance transcription systems and even to organize their output dialogues according to speakers, both of which facilitate the adoption of transcription technologies in everyday life. In particular, we are interested in adapting speaker diarization systems to new conversations, allowing our work to span different continual learning scenarios.

1.3 Objective and contributions

In this work we attempt to better understand and improve continual learning systems via the use of representations that are tailored to and continually improved for a given target task. The recent rise of large pre-trained models allows us to take generic models as a starting point and study methods for progressive adaptation to various environments and conditions. To put it succinctly, we are interested in designing autonomous learning systems able to leverage scarce on-the-job data to adapt to the new environments they are deployed in.

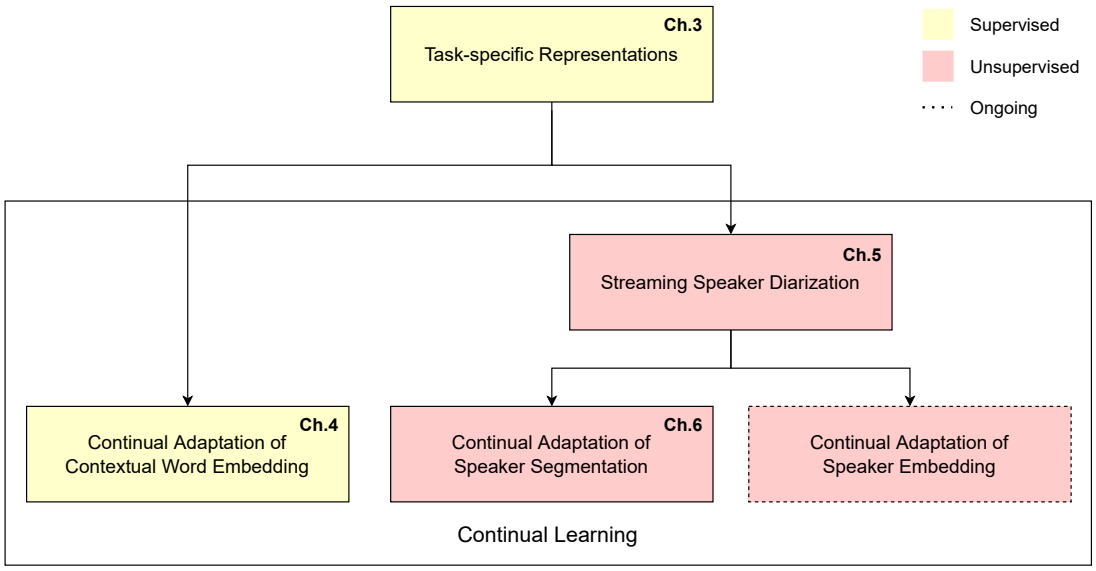


Figure 1.2: Scientific contributions of this work.

In this context, our contributions can be summarized as depicted in Figure 1.2. We first study ways of learning task-specific representations to serve as a knowledge basis for continual adaptation. Next, we explore two very different axes of

continual adaptation: supervised and unsupervised. The supervised axis consists in understanding the continual improvement of recently proposed contextual word embeddings to different languages, while the second, unsupervised axis is more fine-grained, and focuses on autonomous streaming speaker diarization for the adaptation to a live conversation. Finally, given that the latter relies heavily on a pre-trained speaker segmentation model, we investigate how to extend continual adaptation to it as well.

The rest of the manuscript is structured in the following way. In Chapter 2 we discuss the background and previous work on transfer learning and continual learning. In Chapter 3, we present our first contribution on task-specific representations: the systematic comparison of several metric learning loss functions, which spans both written and spoken language tasks. Next, following the results on the written language task from Chapter 3, in Chapter 4 we take a step back from metric learning to present our second contribution, where we investigate the continual adaptation of contextual word embeddings to new languages for the task of supervised sequence labeling. In Chapter 5 we propose a system for streaming speaker diarization that is fully autonomous and self-adaptive by leveraging the representations built in Chapter 3. Later, in Chapter 6 we present our last contribution, which focuses on the continual adaptation of the speaker segmentation model that is the basis of the streaming system proposed in Chapter 5. We investigate how to adapt this model to a new domain where conversations appear sequentially. Finally, we conclude and discuss future research directions in Chapter 7.

1.4 Publications

Some of the work presented in this manuscript has been the subject of the following publications:

- Hervé Bredin, Ruiqing Yin, **Juan M. Coria**, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. pyannote.audio: Neural Building Blocks for Speaker Diarization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Spain, 2020.
- **Juan M. Coria**, Sahar Ghannay, Sophie Rosset, and Hervé Bredin. A Metric Learning Approach to Misogyny Categorization. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, Online, 2020. Association for Computational Linguistics.
- **Juan M. Coria**, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. A Comparison of Metric Learning Loss Functions for End-to-End Speaker Verifica-

tion. In *Statistical Language and Speech Processing*, Online, 2020. Springer International Publishing.

- **Juan M. Coria**, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. Overlap-Aware Low-Latency Online Speaker Diarization Based on End-to-End Local Segmentation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Online, 2021.
- **Juan M. Coria**[1], Mathilde Veron[1], Sahar Ghannay, Guillaume Bernard, Hervé Bredin, Olivier Galibert, and Sophie Rosset. Analyzing BERT Cross-Lingual Transfer Capabilities in Continual Sequence Labeling. In *Proceedings of the First Workshop on Performance and Interpretability Evaluations of Multimodal, Multipurpose, Massive-Scale Models*, Online, 2022. International Conference on Computational Linguistics.
- **Juan M. Coria**, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. Continual Self-Supervised Domain Adaptation for End-to-End Speaker Diarization. In *IEEE Spoken Language Technology Workshop (SLT)*, Qatar, 2022.
- **Juan M. Coria**, Hervé Bredin, Sahar Ghannay, Sophie Rosset, Khaled Zaouk, Ingo Fruend, Bertrand Higy, Amit Kesari, and Yagna Thakkar. Diart: A Python Library for Real-Time Speaker Diarization. Submitted to *The Journal of Open Source Software*, 2022.

[1]Equal contribution, order is alphabetical.

Chapter 2

Related work

"Knowledge is finite. Wonder is infinite."
— Matt Haig, *The Humans*

In this chapter, we dive deeper into the background related to the main research directions in this thesis. In particular, we first discuss the importance of learned representations in machine learning and how they became one of the most popular methods to transfer knowledge in neural networks. We define the concept of transfer learning, and specifically *sequential* transfer, where training is divided into a general-purpose pre-training stage and a fine-tuning stage to a target downstream task, and we explore the advantages and inconveniences of this approach. Finally, we define continual learning and its relationship to transfer learning. We discuss how the evaluation of a continual learning model differs from other approaches, and we categorize and review previous work attempting to solve the catastrophic forgetting problem.

The chapter is structured in the following way. In Section 2.1, we define representations in the context of machine learning, reviewing their journey from hand-crafted features to neural representations. In Section 2.2, we define transfer learning and discuss its most popular variant: sequential transfer learning, in particular through general-purpose representations. Then, in Section 2.3 we introduce the concept of metric learning, an alternative to general-purpose representations capable of learning task-specific representations with the properties of intra-class compactness and inter-class separability. Next, in Section 2.4 we discuss continual learning and its core difficulty: catastrophic forgetting. We discuss ways to measure performance changes across training stages, and we explore previous studies attempting to design systems that learn without forgetting. Finally, we conclude the chapter in Section 2.5.

2.1 From hand-crafted to neural representations

In computer science, numerical representations are ubiquitous. At the lowest level, all information is encoded discretely as ones and zeros, but higher level representations allow us to ignore unnecessary details to manipulate data in more abstract and meaningful ways. In this context, we can define a representation as a mathematical structure describing a specific set of characteristics of a piece of data. For example, an image may be represented as a 3-dimensional tensor of red, green and blue color intensity, a word as a vector of discrete characters (with each character encoded as an integer), or an audio recording as a vector of air pressure levels at different points in time.

Although this generally facilitates the communication between humans and machines, not all representation types are suitable for all applications. In fact, many machine learning algorithms, and in particular gradient descent, rely on continuous numerical representations to optimize the parameters of a model. This can be problematic for data that is normally represented discretely, such as text. But even continuous representations such as audio or images as real vectors may be too high-dimensional to process efficiently and too noisy to extract complex information from. For instance, it could be difficult to detect a face in millions of pixels, or someone's voice in a vector of 29 million audio samples (10 minute recording sampled at 48Khz).

In their definition of representation learning, Bengio et al. (2013) explain this complexity as the combination of multiple factors of variation in the data generation process that may not be fully covered in the training set and hence be a source of errors. For example, a speaker's voice may be recorded with microphones of different qualities, or in rooms with different types of reverberation or noise. Thus, in this example, a model focusing on speaker detection should learn to *disentangle* the noise (*e.g.* reverberation or microphone quality) from the signal (*e.g.* the voice and its identity). To this end, a variety of approaches exist to extract discriminative (*e.g.* between speakers) and invariant (*e.g.* to reverberation) representations, ranging from manual engineering to fully data-driven methods like transfer learning.

In the context of machine learning, we can define a model as a function f with parameters θ that are optimized, or *trained*, to output a prediction y given a numerical representation x, or *features*, of a piece of data, where x and y can take a variety of forms depending on the task that f is trained to solve. In the early days of machine learning, the most common way of training a model consisted in carefully crafting the representation x that was thought most meaningful for the task at hand, as shown in Figure 2.1. Some notable examples of this are

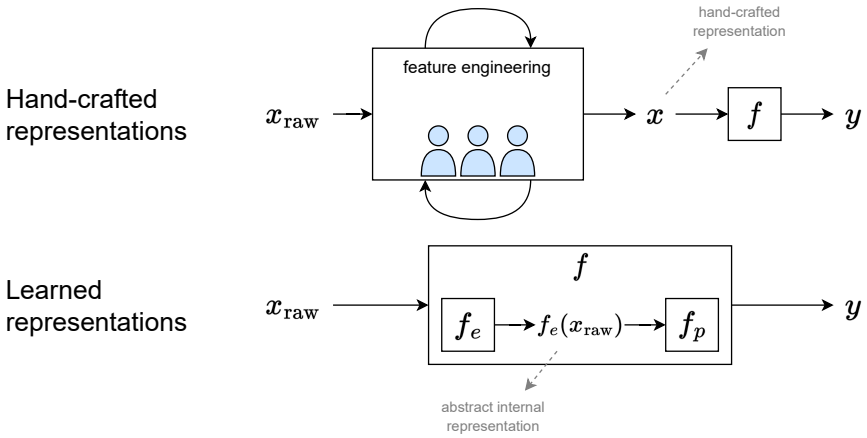


Figure 2.1: Comparison between hand-crafted and neural representations. Feature engineering requires human intervention to design a representation function, which can be costly and time-consuming. Alternatively, a representation function f_e (for *encoding*) can be learned from data, but its black-box nature forces f to rely on an additional function f_p (for *prediction*).

part-of-speech (POS) tags (Florian et al., 2003) or prefixes and suffixes (Chieu and Ng, 2002) in natural language processing. In the domain of audio and speech processing, well-tested hand-crafted features like spectrograms and Mel Frequency Cepstral Coefficients (MFCC) are still used successfully today (Snyder et al., 2018; Yoshioka et al., 2018; Fujita et al., 2019).

However, feature engineering is often time consuming and requires expert knowledge in areas like linguistics or signal processing. As noted by Ravanelli and Bengio (2018), these features are engineered from our limited perception and understanding, providing no guarantee that they are well suited for a given task. To make matters worse, each task may have its own set of optimal features to be discovered from trial and error, making the whole endeavour extremely costly.

In contrast, neural networks leverage *back-propagation* (Rumelhart et al., 1986) and gradient descent to learn internal hidden representations with increasingly complex levels of abstraction directly from the training data. As depicted in Figure 2.1, the most complex representations (those of the last layer), obtained from the *encoding* function f_e, are then decoded by a final jointly trained *prediction* layer f_p to output a prediction for the target task. With the advent of deep learning (LeCun et al., 2015) and the availability of large annotated corpora, delegating feature engineering to the models themselves became a reliable alternative. This allowed inputs x to be replaced by lower-level representations x_{raw} that do not require costly human efforts to obtain. Using neural networks to learn representations from

raw data like images or audio (LeCun et al., 2015) has even outperformed hand-crafted features in a variety of tasks like image classification (Okafor et al., 2016), or speaker identification and verification (Ravanelli and Bengio, 2018). However, contrary to their hand-crafted counterpart, neural representations are generally difficult to interpret. In fact, neural networks are generally considered "black box" algorithms, which means that once trained, it is extremely difficult to explain how they reach a given conclusion. As a result, these representations rely heavily on the f_p function that is trained on top of them to solve a classification or regression task.

2.2 Transfer learning

In recent years, the availability of large corpora has allowed machine learning practitioners to train larger models with increased performance, but the sheer size of these corpora make manual annotation increasingly costly. Hence, it is crucial to understand how to effectively exploit large quantities of unlabeled data.

The field of *transfer learning* studies ways to capture and reuse acquired knowledge from one task to facilitate the learning of a second one (Pan and Yang, 2010; Ruder, 2019). Assuming that the internal parameters of neural networks capture such knowledge and encode it in internal representations, a considerable effort has been made to train general-purpose models that can later be adapted to new more specific tasks (Peters et al., 2018; Pascual et al., 2019; Mikolov et al., 2013; Desplanques et al., 2020).

As noted in previous studies (Pan and Yang, 2010; Ruder, 2019), transfer learning can manifest in a variety of ways depending on the type of knowledge that is transferred (*e.g.* language, domain, task, etc.), the type of supervision, or even the amount of data available for the downstream task (*e.g.* few-shot or zero-shot transfer). However, in the context of this work it is useful to separate transfer learning into *joint* and *sequential*. Consider a set $\mathbf{S}$ of two corpora $\mathcal{D}_1, \mathcal{D}_2 \in \mathbf{S}$ sharing a set of characteristics, such as task or domain of application (*e.g.* both $\mathcal{D}_1$ and $\mathcal{D}_2$ target the named entity recognition task in textual sentences). On the one hand, joint transfer learning consists in training a randomly initialized model m_0 on the concatenation of the entire $\mathbf{S}$ at the same time. Provided that $\mathcal{D}_1$ and $\mathcal{D}_2$ are sufficiently similar, updating the model to correctly predict a given $x_1 \in \mathcal{D}_1$ can potentially shift model parameters closer to an optimal solution for a given $x_2 \in \mathcal{D}_2$ and vice-versa. However, this type of transfer is intensive in computing resources and often requires access to a large $\mathbf{S}$.

On the other hand, sequential transfer learning is defined in two stages. The first

pre-training stage consists in training a model m_0 from scratch on the initial $\mathcal{D}_1$. Although not a requirement, this typically corresponds to a generic task with a large amount of unlabeled data and a self-supervised objective, which is usually derived from the structure of the data itself. For example, in textual sentences it can be the prediction of a masked word (known as masked language modeling, or MLM for short) or predicting whether two sentences are contiguous in a document (next sentence prediction, or NSP for short) (Devlin et al., 2019). The second *fine-tuning* stage continues training for model m_1 (resulting from the first stage) on the remaining corpus $\mathcal{D}_2$, which usually corresponds to a more specific *downstream* task with labeled (and less) data. A typical example of this is the popular BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2019) in natural language processing, where $\mathcal{D}_1$ is the concatenation of English Wikipedia and BooksCorpus (Zhu et al., 2015) and the model is pre-trained using the MLM and NSP objectives. Later, the model is fine-tuned to a variety of supervised tasks separately with less available data, such as sentiment classification or semantic textual similarity, each playing the role of a $\mathcal{D}_2$.

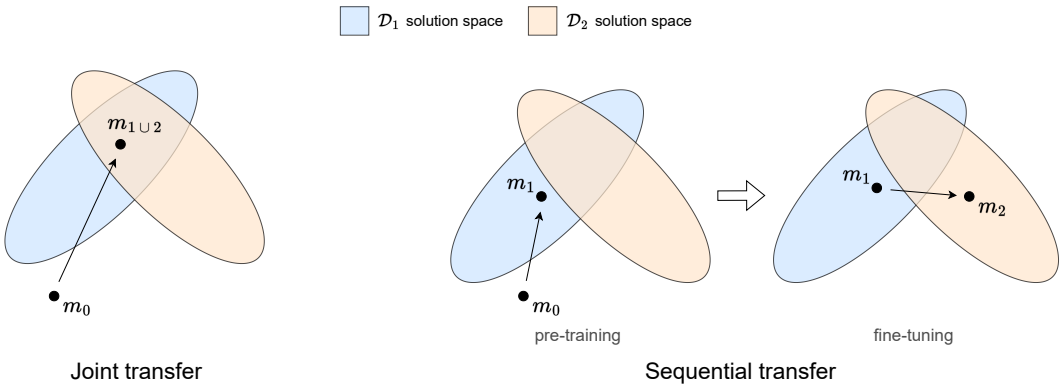


Figure 2.2: Parameter shifts in joint and sequential transfer learning based on the explanation of catastrophic forgetting by Kirkpatrick et al. (2017). Training jointly on $\mathcal{D}_1$ and $\mathcal{D}_2$ at the same time allows gradient-descent to find parameters in a solution space that is common to both corpora. In contrast, sequential transfer aims at shifting parameters closer to a possible solution for $\mathcal{D}_2$, essentially finding a good "initialization" for it, but depending on further training. Notice that the sequential scenario is also prone to forgetting, as training on $\mathcal{D}_2$ moves model parameters past the solution space for $\mathcal{D}_1$, decreasing its associated performance.

To highlight the key differences between joint and sequential transfer learning, consider the example shown in Figure 2.2. Joint transfer attempts to learn model parameters that constitute a solution for both $\mathcal{D}_1$ and $\mathcal{D}_2$ at the same time, while sequential transfer first learns a solution for $\mathcal{D}_1$ that is a good initialization for $\mathcal{D}_2$,

facilitating the work of the subsequent fine-tuning stage that shifts parameters to fit $\mathcal{D}_2$.

Sequential transfer learning has proven to be a reliable technique to improve learning efficiency and performance on $\mathcal{D}_2$ (Peters et al., 2018; Devlin et al., 2019; Pascual et al., 2019), in particular when the unlabeled $\mathcal{D}_1$ is orders of magnitude larger than $\mathcal{D}_2$. Moreover, it fragments the development phase in a way that allows general-purpose *pre-trained* models to be distributed to third parties not having access to the same amount or quality of training data, or even to the same computational resources. The fact that sequential transfer is so effective is one of our main motivations to study continual learning of representations.

While transfer learning restricts $|\mathbf{S}| = 2$, continual learning can be seen as a special form of sequential transfer learning where $\mathbf{S}$ is a sorted list (instead of an unsorted set) with $|\mathbf{S}| \geq 2$. However, as shown in Figure 2.2, the fine-tuning stage is allowed to shift model parameters out of the solution space of $\mathcal{D}_1$ as long as an acceptable solution for $\mathcal{D}_2$ is found. Although not a problem for the scope of transfer learning, this represents a great difficulty in continual learning, which aims at incrementally improving performance on the entire $\mathbf{S}$ instead of only on $\mathcal{D}_2$. This phenomenon is normally referred to as *catastrophic forgetting*, and will be discussed in more detail in Section 2.4. Moreover, as depicted in Figure 2.3, given a general-purpose pre-training stage on $\mathcal{D}_0$, sequential transfer learning restricts fine-tuning to a single downstream $\mathcal{D}_i$, which is then deployed to work on $\mathcal{D}_i$-like data. In contrast, the goal of continual learning is to transfer knowledge to and during the production phase, allowing a single model to work on any type of data seen in the past. As a consequence, the inference time and space complexity needed to work with multiple downstream $\mathcal{D}_i$ in transfer learning is $O(|\mathbf{S}|)$, whereas for continual learning it is $O(1)$.

Given its success to capture knowledge in the underlying training data, sequential transfer is often used to learn efficient and meaningful numerical representations, also known as *embeddings*, that require less data and less training in the fine-tuning stage. In speech processing, training models to produce embeddings containing discriminative speaker qualities has been widely used for speaker verification (Snyder et al., 2018; Desplanques et al., 2020) and speaker diarization (Anguera et al., 2012; Diez et al., 2018). In NLP, word embeddings (Mikolov et al., 2013; Pennington et al., 2014) have been successfully transferred to a great variety of tasks, such as sentiment classification or named entity recognition.

In the early days of embedding learning, model weights m_1 were usually discarded, and the learned word representations were extracted and stored. These were then used to represent text as a sequence of word embeddings that could eventually

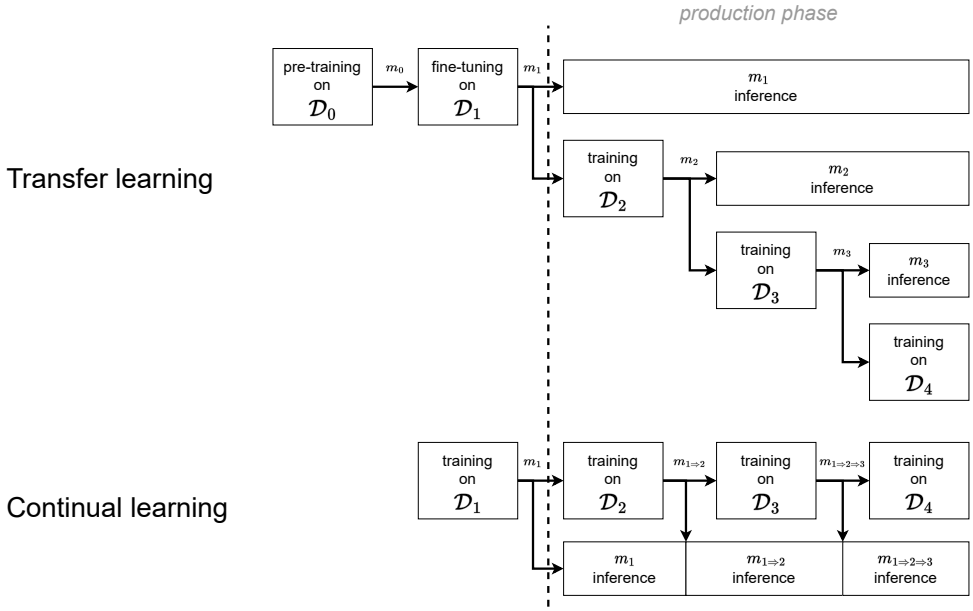


Figure 2.3: Relying on transfer learning to adapt to new production data requires to increase the number of production models and inference time by the number of fine-tuning corpora in order to avoid forgetting. In contrast, continual learning reduces these costs to $O(1)$ by adapting a unique model during the production phase.

be fine-tuned to the downstream task if required, alongside with a task-specific model. Aside from the performance benefits, the use of word embeddings also gained popularity because of their ability to encode high-level information that was hard to encode manually.

However, the benefits of these *static* word embeddings are limited when applied in different contexts than seen during training. For instance, the word "bank" may refer to a banking institution, but it could also be used to denote a "river bank". A single one-size-fits-all embedding for this word would need to somehow encode this ambiguity, provided that it has been seen by the model in the training data, which is subject to be (and often is) incomplete.

This is one of the main motivations behind *contextual* word embeddings (Peters et al., 2018; Devlin et al., 2019), which are able to represent this ambiguity by producing word embeddings that vary according to the sentence in which the word is used. In other words, given a vocabulary $\mathcal{V}$ and the set of all possible sentences $\mathcal{S}$, while a static embedding function $f_{\text{static}} : \mathcal{V} \to \mathbb{R}^d$ maps a single word to a vector of dimension d, a contextual embedding function $f_{\text{contextual}} : (\mathcal{V}, \mathcal{S}) \to \mathbb{R}^d$ maps a word and its context sentence to a vector of dimension d. Contrary to a

finite vocabulary $\mathcal{V}$, since the number of possible sentences in $\mathcal{S}$ is infinite, it is common to reuse and fine-tune model parameters m_1 instead of fixed word vectors.

2.3 Metric learning

Today, the best performing representations are typically obtained by pre-training a general-purpose neural network on a large unlabeled corpus. Although these neural embeddings tend to reduce costs and achieve better performance than their hand-crafted counterpart, one aspect that remains unclear is how to interpret them.

Interpreting embeddings has been attempted before. Early studies on static word embeddings (Mikolov et al., 2013; Pennington et al., 2014) have shown that they display a form of additive compositionality. More formally, given words w_1 and w_2, the embedding $f_{\text{static}}(w_1) + f_{\text{static}}(w_2)$ would be close to an embedding whose associated word is a semantically meaningful composition of w_1 and w_2. As a result, this makes it possible to reason about semantic content with simple mathematical operations, a famous example being $king - man + woman \approx queen$.

Additionally, this means that a well-defined mathematical distance metric like the euclidean distance should be loosely correlated with the semantic similarity between two words. For example, the embedding for *man* would be closer to *woman* than to *moon*. This surprising property of static word embeddings is emergent, as the objective function does not specifically enforce it during training, and it has been the subject of several research studies (Gittens et al., 2017; Jatnika et al., 2019).

In contextual word embeddings with Transformer models (Vaswani et al., 2017) there is also evidence supporting a correlation between word embedding distance and semantic similarity at different levels of the architecture (Reif et al., 2019). For example, the representations of "a die" (as a noun) are closer together than those of "to die" (as a verb). However, further studies are required to better understand how this contextual representation space is formed and how it can be exploited. Since Transformer models are based on the concept of attention, it is also common to look at the activations of these layers in order to understand the information encoded in an embedding, but it remains unclear how to manipulate this information more effectively than an appended trained neural layer.

Similarly to static word embeddings, some tasks like face recognition, speaker verification, or speaker diarization, can often benefit from enforcing a correlation between a complex definition of similarity and a simple mathematical distance function in order to manipulate embeddings easily during inference.

In particular, considering a face or a speaker as a class, intra-class compactness (*i.e.* low distance between same-class embeddings) and inter-class separability (*i.e.* high distance between embeddings from different classes) are often desirable properties that improve interpretability by mapping a complex and abstract property, like the identity of a voice in a recording, to a simpler mathematical one like the cosine distance. Additionally, this allows well-known distance-based unsupervised algorithms like clustering to work directly on such abstract concepts.

Following this hypothesis, the goal of *metric learning* is to find a pair (f, d) of representation function f and distance function d with the following ideal property. Given a sample x_a (*e.g.* an utterance) belonging to a class (*e.g.* a speaker), any sample x_p belonging to the same class should be closer to x_a than any sample x_n belonging to a different one:

$$d(f(x_a), f(x_p)) < d(f(x_a), f(x_n)) \tag{2.1}$$

Metric learning techniques aim at simplifying the distance function d all the way down to the most simple distance function (*e.g.* euclidean or cosine distance), delegating all the work to the representation function f (usually a trained neural network) that should ensure intra-class compactness and inter-class separability. Figure 2.4 depicts this concept graphically, where the distance between learned representations can be compared to address a given task.

To better understand how the representation space is constructed, Figure 2.5 shows an example in two dimensions using the angular distance as the choice for d. In this example, the distance θ_{ij} between $f(x_i)$ and $f(x_j)$ is small because samples x_i and x_j both belong to class 1, while the distance between $f(x_l)$ and $f(x_j)$ is large

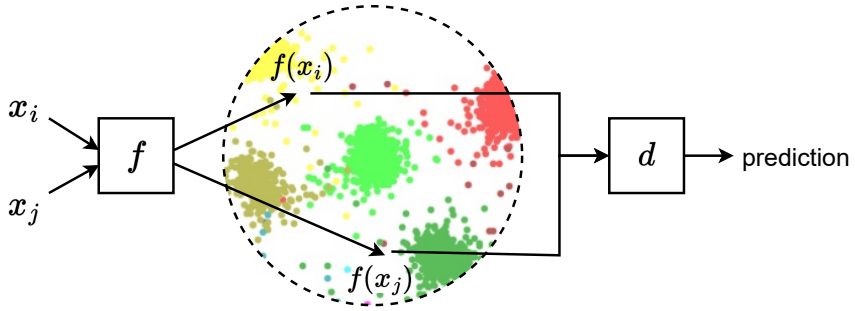


Figure 2.4: Given training samples x_i and x_j, the neural network f produces representations $f(x_i), f(x_j) \in \mathbb{R}^m$ such that the distance $d(f(x_i), f(x_j))$ between them is small if they belong to the same class (compactness) and large otherwise (separability).

because x_l belongs to class 2. Internally, some metric learning techniques rely on centroids c_k that are trained jointly with the representation function f and can be seen as a canonical representation of each class.

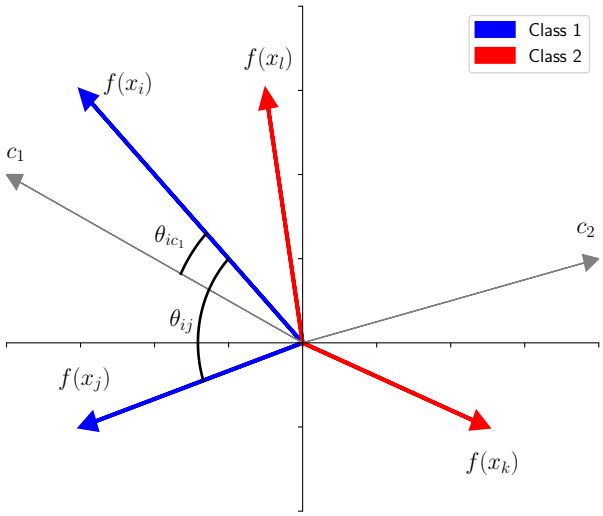


Figure 2.5: Metric learning approaches aim at making representations of similar classes close to each other, while separating those from different ones as much as possible. Some methods rely on jointly trained centroids that act as attractors for representations belonging to the same class.

The properties of intra-class compactness and inter-class separability are typically encouraged during training by modifying the loss function in a specific way. Popular techniques from the literature are reviewed in more detail in Chapter 3, where we perform a systematic comparison between several metric learning loss functions.

2.4 Continual learning

Compared to learning in humans, single-stage learning seems counter-intuitive and even inefficient, as we are constantly learning and revisiting previous knowledge with every interaction with our environment. Single-stage learning assumes that problems are stationary and that training and production examples are independent and identically distributed (*i.i.d.*). Assuming that the distribution of training and production data is identical may be acceptable for a small subset of problems, but this is rarely the case in most practical scenarios. A common approach to deal with this training-production mismatch is to rely on joint transfer by interleaving development and production phases, where each development phase trains a new model with the entire data available up to that moment. As shown in Figure 2.2

(page 31), this solves the problem by shifting model parameters towards a subspace that constitutes an acceptable solution to the task associated with initial and new training data. Unfortunately, this back-and-forth between development and production is increasingly costly. As new examples are collected in production, the cost of development phases keeps raising in terms of computing resources, manual annotation and engineering labor.

Continual learning (Hadsell et al., 2020) denotes the ability of machine learning systems to acquire knowledge and improve their performance over time through a flow of data. This flow is generally presented to the model in windows that are accessible one at a time, and where access to previous windows is limited or even forbidden in order to keep training costs within bounds. Whereas transfer learning is defined by a set of corpora $\mathbf{S}$, where $|\mathbf{S}| = 2$, in continual learning the flow of data can be considered a sequence of corpora $\hat{\mathbf{S}} = (\mathcal{D}_1, \ldots, \mathcal{D}_L)$, where $|\hat{\mathbf{S}}| = L$ and $L \geq 2$. In this context, the windows to which the model has sequential and limited access are represented by each $\mathcal{D}_i$. As an example, each of these windows can be as small as a few seconds of audio or as big as multiple hour-long recordings.

As mentioned before, transfer learning typically separates training in two different and well-defined stages: a pre-training stage on a large unlabeled corpus $\mathcal{D}_1$ and a fine-tuning stage on a smaller downstream corpus $\mathcal{D}_2$, where the associated task of $\mathcal{D}_1$ is usually different and more generic than $\mathcal{D}_2$. In contrast, training stages in continual learning typically share a set of common characteristics and differ only by a main aspect that is the focus of improvement for the continually trained model. To facilitate our discussion, we refer to this main differing aspect as the *adaptation axis*, which can manifest in the form of varying tasks (*e.g.* first dog-cat image recognition and later bird-mammal), domains (*e.g.* named entity recognition on scientific, then financial, later news documents), or an incremental set of classes (*e.g.* speaker identification where new and old speakers can appear at any time), among others.

Apart from the immediate advantages of a continual learning agent, like constant adaptation to a changing problem, it also has the potential of improving learning efficiency and transfer across all $\mathcal{D}_i$ by leveraging past knowledge (Hadsell et al., 2020). However, contrary to transfer learning, which is only interested in the performance of the downstream $\mathcal{D}_2$, continual learning is a form of *incremental learning*, meaning that the model should accumulate (and not replace) the knowledge it acquires across training stages. However, as depicted in Figure 2.2 (page 31), the main issue that arises when training a model on the sequence $\hat{\mathbf{S}}$ is known as *catastrophic forgetting* (French, 1999), where model parameters change drastically to suit the current $\mathcal{D}_i$, losing the knowledge acquired for any previous $\mathcal{D}_{j<i}$. Much of the existing continual learning work attempts to prevent catastrophic forget-

ting by constraining neural networks in different ways. The hope is to guarantee the preservation of previously learned parameters or previously given predictions, either with manual or learned algorithms (Kirkpatrick et al., 2017; Robins, 1995; Javed and White, 2019).

The rest of this section is structured as follows. We first define in Section 2.4.1 common metrics used in continual learning to measure performance beyond task-specific performance metrics like accuracy or F1 score. In subsequent sections, we follow the categorization initially introduced by Hadsell et al. (2020) to explore the techniques that have been proposed in the past to address catastrophic forgetting.

2.4.1 Forgetting and transfer

Given that the focus is on measuring performance changes over time, model evaluation is usually a challenging endeavor in continual learning. A simple way to evaluate incremental learning is to use a task-specific metric (*e.g.* accuracy) on a test set $\mathbf{S}_{\text{test}}$ after training on each $\mathcal{D}_i$, where $\mathbf{S}_{\text{test}}$ contains held-out examples from all $\mathcal{D}_i \in \hat{\mathbf{S}}$. In this context, the goal of the learner is to increase its performance on $\mathbf{S}_{\text{test}}$ with each training stage, effectively accumulating knowledge. However, this method has two significant drawbacks. First, it does not provide a way to measure forgetting and could be misleading if $\mathbf{S}_{\text{test}}$ is not built carefully. For example, performance on $\mathbf{S}_{\text{test}}$ could increase after learning $\mathcal{D}_i$ because there is a large number of held-out examples from $\mathcal{D}_i$ in $\mathbf{S}_{\text{test}}$. Second, although convenient in experimental setups, it assumes that $\mathbf{S}_{\text{test}}$ can be known in advance, which is not often the case in practice.

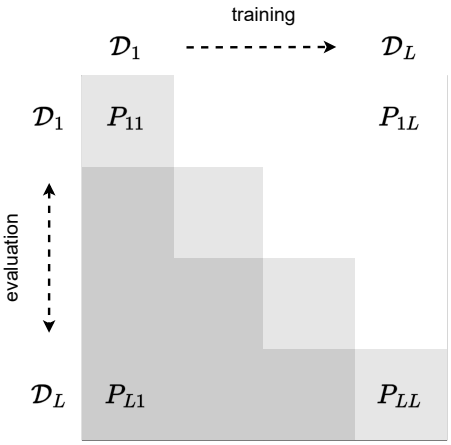


Figure 2.6: Performance matrix commonly used for continual learning evaluation.

A more informative approach that is commonly used (Lopez-Paz and Ranzato, 2017; Pomponi et al., 2020) consists in computing a performance matrix $P \in \mathbb{R}^{L \times L}$ as depicted in Figure 2.6, where P_{ij} is the performance on the $\mathcal{D}_i$ test set after training on the $\mathcal{D}_j$ training set (with higher values being better).

This matrix is independent from the target task and metric, and it shows a broader picture of how performance fluctuates across the training stages. To measure dif-

ferent continual performance metrics, it is sometimes useful to split P into various sections. First, the diagonal $diag(P)$ corresponds to the performance on a given $\mathcal{D}_i$ immediately after training on it, which is generally high as there is no room for forgetting. Second, the upper triangular matrix (excluding $diag(P)$) represents the performance on all past $\mathcal{D}_i$, hence being of interest to measure forgetting. Finally, a third section of P is the lower triangular matrix (also excluding $diag(P)$), showing performance on all future $\mathcal{D}_i$. As with $\mathbf{S}_{\text{test}}$, the latter is rarely accessible in real use cases but it is sometimes useful in controlled experiments to measure transfer to new unseen data without further training, a phenomenon known as *zero-shot* transfer.

However, using an entire matrix to compare models can sometimes be challenging to interpret and lead to ambiguity. As a result, it is also common practice to define metrics that are computed from P. Typically, we measure forgetting on a past $\mathcal{D}_i$ as the difference in performance at two moments: after it is first shown to the model (P_{ii}) and after the training sequence has ended (P_{iT}):

$$forgetting(\mathcal{D}_i) = P_{ii} - P_{iT} \tag{2.2}$$

This is often averaged over all rows to obtain a single value spanning the entire sequence (Lopez-Paz and Ranzato, 2017). Because this metric usually ignores the performance fluctuations that happen between the beginning and the end of the training sequence $\hat{\mathbf{S}}$ (*i.e.* the columns of P between i and T), some studies choose to average all performance differences between columns as well (Pomponi et al., 2020). However, as we explain in Chapter 4, averages may not always provide an accurate vision of forgetting and transfer.

On the other hand, matrix P also allows to measure transfer between future and past $\mathcal{D}_i$. These types of transfer are known as *forward* and *backward* transfer respectively (Hadsell et al., 2020). More concretely, better performance on a given $\mathcal{D}_i$ relative to learning $\mathcal{D}_i$ in isolation is called *forward transfer*, as it represents the transfer from past training stages to new ones. This is defined as:

$$forward(\mathcal{D}_i) = P_{ii} - base_i \tag{2.3}$$

where $base_i$ is the performance on $\mathcal{D}_i$ learned in isolation. Other studies have also defined forward transfer using $P_{i\,i-1}$ (Lopez-Paz and Ranzato, 2017), essentially looking at a kind of zero-shot forward transfer where the model should perform well on $\mathcal{D}_i$ before actually seeing any of its examples.

Finally, backward transfer is defined as the improvement on previously seen examples from $\mathcal{D}_{j<i}$ thanks to the acquisition of new knowledge from $\mathcal{D}_i$, and is equivalent to negative forgetting:

$$backward(\mathcal{D}_i) = P_{iT} - P_{ii} \tag{2.4}$$

As with forgetting, both forward and backward transfer metrics are often averaged across all $\mathcal{D}_i \in \hat{\mathbf{S}}$ (Lopez-Paz and Ranzato, 2017).

2.4.2 Memory-based methods

When thinking about learning without forgetting, it is natural to consider the existence of a long-term memory. One of the first attempts at solving the catastrophic forgetting problem, named *rehearsal* or *replay* (Robins, 1995), consists in dynamically building an external memory buffer with previously seen examples, which are then shown to the model again, or "replayed", when training on new examples. As shown in Figure 2.7 (page 42), replay does not restrict parameter changes during training, and instead saves past information in the buffer to include in future training stages. The motivation behind this idea is that by replaying examples from old training stages, model parameters are forced to stay in a subspace that represents a solution for both new and previous data.

Although the simplicity of this approach is attractive, using a memory buffer introduces a set of additional difficulties, such as determining the buffer size. A small buffer may leave out useful examples that should not be forgotten, but a buffer that is too large may consume more computing resources. Regardless of size, it is also impossible for a memory buffer to scale to infinity without forgetting, as a buffer of size B cannot store examples from $B + 1$ training stages. Moreover, one must devise a strategy to decide which examples to keep and which examples to remove. Finally, it is worth noting that this technique relies on the availability of previously seen examples, which may not be the case if data cannot be stored due to privacy-related or legal concerns.

Several studies have attempted to address the shortcomings of replay while following the same principle. In generative replay (Shin et al., 2017; Rao et al., 2019; Sun et al., 2020), a generative model is trained to produce synthetic examples to fill the buffer instead of relying on past data. Although this alleviates much of the issues mentioned before, like needing access to previous data, generative models are often difficult to train and may be more resource hungry than other approaches.

Latent replay (Pellegrini et al., 2020) attempts to find a middle ground by storing and replaying activations of middle layers, effectively splitting the model in two parts, where the lower part of the model (*e.g.* a feature extraction layer) is trained with a slower learning rate than the upper part. This reduces storage and computational costs significantly, but keeps the difficulties associated with the management of an external buffer. Additionally, stored latent representations are subject to a slow "aging" effect as new training stages keep morphing the representation space. This means that recomputing the buffer is necessary from time to time to prevent divergence.

A method based on embedding regularization (Pomponi et al., 2020) follows a similar approach, adding a penalization term to the loss function so that specific middle-layer representations are consistent with those from examples stored in the memory buffer. Another approach called Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017) uses the buffer to constrain gradients to point in the same direction as previous $\mathcal{D}_{j<i}$ during training on the new $\mathcal{D}_i$.

2.4.3 Constraint-based methods

This family of approaches is rooted in the stability-plasticity dilemma (Grossberg, 1987; Abraham and Robins, 2005). The term stability denotes the fact that model parameters remain unchanged, hence keeping previously acquired knowledge intact. Conversely, the term plasticity refers to the capability of drastic change in these parameters, allowing learning to occur. According to this view, stability and plasticity are opposing properties of a model that need to be carefully balanced in order to retain knowledge while allowing learning.

In this context, we can categorize gradient descent as an algorithm with full plasticity but no stability, since it allows parameters to change without restriction. In neural networks and deep learning, this premise has been the basis for multiple works attempting to keep catastrophic forgetting under control by avoiding drastic changes to weights (*i.e.* neural network parameters) deemed important to previous training stages. Various strategies have been proposed in order to determine which particular parameters should be protected from modifications. Given learned weights $\mathbf{w}^{(t-1)}$ after training on $t-1$ tasks, and Ω_i the importance of a given parameter $\mathbf{w}_i$, these methods learn a new task t using a regularized loss function of the form:

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}_t(\mathbf{w}) + \lambda \sum_i \Omega_i \left(\mathbf{w}_i - \mathbf{w}_i^{(t-1)} \right)^2 \tag{2.5}$$

where $\mathcal{L}_t$ is the loss of task t and λ is a hyper-parameter balancing stability and

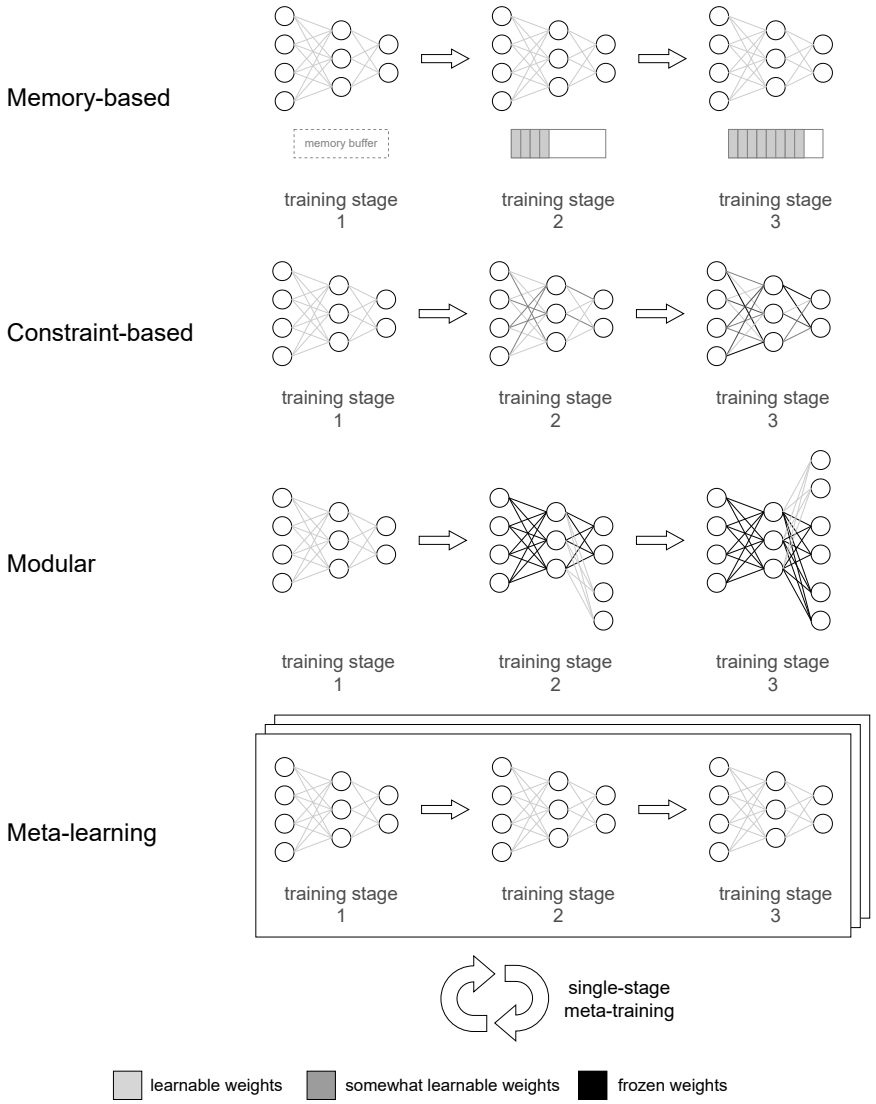


Figure 2.7: Existing strategies to prevent catastrophic forgetting. Memory-based techniques rely on an external buffer that is progressively filled and mixed with new training data. Since a buffer introduces additional challenges (like devising a replacement strategy when full), constraint-based methods progressively freeze crucial parameters in order to keep model predictions from changing. However, this may limit future learning and backward transfer. Instead, modular approaches introduce new parameters at each training stage, raising spatial complexity from $O(1)$ to $O(|\hat{\mathbf{S}}|)$. Finally, meta-learning approaches propose a data-driven approach where the model is trained to learn without forgetting. Since meta-training is essentially single-stage, this approach is vulnerable to changes in implicit assumptions during training.

plasticity. A depiction of such an algorithm is presented in Figure 2.7, where we can see model weights being progressively frozen after each training stage. A famous example of constraint-based algorithms is Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), which uses an Ω proportional to the diagonal of the Fisher information matrix. Arguing that this computation may be too expensive, other studies have tried to improve EWC by estimating weight importance from gradients in an online fashion, like Synaptic Intelligence (SI) (Zenke et al., 2017) and Memory Aware Synapses (MAS) (Aljundi et al., 2018), while others have proposed to determine the importance at the level of neurons instead of parameters (Paik et al., 2020).

One of the main difficulties of these approaches is that, since there is a fixed number of model parameters, increasingly constraining changes may result in a completely frozen model that is incapable of learning (*i.e.* with full stability and no plasticity). As noted by Lopez-Paz and Ranzato (2017), contrary to memory-based methods, an additional disadvantage of enforcing stability in this way is that backward transfer (*i.e.* performance improvements on $\mathcal{D}_{j<i}$) cannot happen because the modification of previously used parameters is highly limited.

2.4.4 Modular methods

Modular approaches to continual learning consist in dynamically altering the architecture of a model to account for new training stages without interfering with existing parameters. In this context, previously learned parameters are considered an acceptable solution for past training stages and their modification is minimized or even forbidden.

A naive application of this principle consists in using a frozen pre-trained model as a common main architecture, appending new layers on top of it at each new training stage, as shown in Figure 2.7. Since new layers are trained from scratch and then frozen, training time is constant with respect to the number of training stages, but the required space grows linearly and can be problematic in the long term or in low-resource conditions, similarly to our example on the complexity of transfer learning in Figure 2.3. To make matters worse, freezing parameters also hinders backward and forward transfer across training stages.

In a similar manner, Terekhov et al. (2015) propose to attach a new set of parameters to each main architecture layer (or to a few specific ones) instead of adding layers on top. These attached parameters, called "blocks", are also created in each training stage, trained from scratch, and then frozen in order to preserve their behavior. A simplified example of this concept is shown in Figure 2.8, where each block layer receives the output from all previous layers and sends its own output

to the next block layer. Contrary to the naive method introduced above, this type of architecture does allow for forward transfer, although backward transfer is still highly limited.

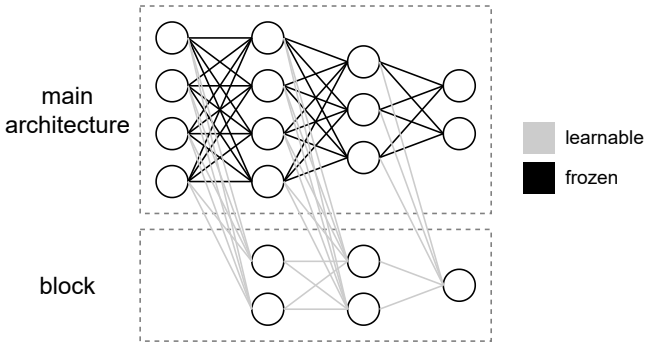


Figure 2.8: Block modular architecture as proposed by Terekhov et al. (2015). Each block layer receives the output from all previous layers, but its own output is only connected to the following block layer.

Claiming that these modular architectures may add an excessive number of parameters that require large amounts of training data, an alternative method called Learning without Forgetting (LwF) (Li and Hoiem, 2016) proposes to ensure that predictions for old tasks remain consistent by relying on a knowledge distillation (Hinton et al., 2015) penalty in the loss function. As shown in Figure 2.9, knowledge distillation can be defined as a teacher-student approach in which a student model m_s is trained to match the outputs $\hat{y}$ of a teacher model m_t, as if they were ground-truth labels. Similarly to the naive method, LwF uses appended layers on top of the main architecture. However, contrary to the naive approach, model parameters are never frozen, leaving room for both backward and forward transfer.

Instead of replaying old examples, LwF takes a snapshot m_{i-1} of the model before training stage i and records its output $\hat{y}^{(j<i)}$ of the appended layers for training stages j using the data of training stage i. Then, the model m_i (initially a copy of m_{i-1}) is trained on the current training stage with its corresponding loss function, but adding a loss penalty whenever its output $\hat{y}_s^{(j<i)}$ of the layers j appended in the past differs from $\hat{y}^{(j<i)}$.

As mentioned before, one of the main problems of modular architectures for continual learning is their linearly growing cost with respect to the amount of training stages. Some approaches (Draelos et al., 2017; Yoon et al., 2018; Rao et al., 2019) attempt to solve this issue by relying on a dynamic expansion approach, in which new parameters are added only when they are needed. The decision to expand

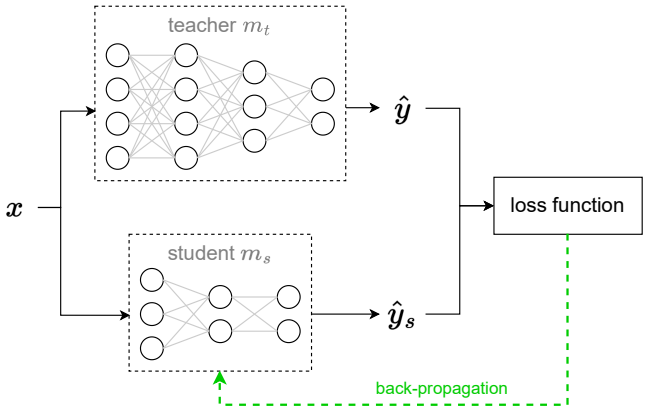


Figure 2.9: The teacher-student approach of knowledge distillation.

the model can be based on several metrics, such as the reconstruction error in autoencoder models (Draelos et al., 2017) or the value of a classification loss function (Yoon et al., 2018). Although this reduces the amount of parameter additions, the cost still remains linear and potentially prohibitive for large numbers of training stages.

2.4.5 Meta-learning

Similarly to how hand-crafted representations gave way to neural representations that are automatically learned from data, recent studies suggest that it is possible to train models to learn continually and discover how to avoid forgetting.

This data-driven approach is based on previous work on meta-learning, or *learning to learn* (Vanschoren, 2019). The goal of non-continual meta-learning algorithms like "model-agnostic meta-learning" (MAML) (Finn et al., 2017) is to automatically discover inductive biases that lead to a model quickly learning a new task from a handful of examples (*i.e.* few-shot learning). In other words, it attempts to find an initialization of model parameters such that learning a new task from those parameters only requires a few examples and training iterations. Put simply, meta-learning algorithms for few-shot learning can be defined as two nested training loops. The first *inner* loop trains the current model initialization on a few set of examples from a selected target corpus, while the second *outer* loop repeats this process for a large number of corpora, optimizing the initial model parameters at each iteration.

In the context of continual learning, a straightforward alternative is to modify the inner loop to train on a sequence of corpora $\mathcal{D}_i$ (as shown in Figure 2.7), and modify the loss function in the outer loop to encourage few errors on the

entire sequence, effectively learning a (hopefully) optimal initialization of model parameters to learn without forgetting. However, this can quickly prove a difficult endeavour because of the amount of computational resources needed.

To solve this, recent work (Javed and White, 2019) has proposed an *online* approach in which each $\mathcal{D}_i$ is reduced to a single example. However, they found that learning a model initialization was ineffective in reducing the catastrophic forgetting problem. Instead, they propose an online-aware meta-learning algorithm (OML) to learn a representation function f that favors low forgetting in a subsequent prediction function g. More formally, given a sequence of inputs $[x_1, \ldots, x_L]$ and ground-truth labels $[y_1, \ldots, y_L]$, the inner loop consists in training g (from scratch) on each pair $(f(x_i), y_i)$ in order using a task-specific loss function, while the outer loop updates the parameters of f with a loss function that penalizes errors in the entire sequence. Interestingly, they also found that the representations learned in f where sparse, which has been thought to reduce forgetting in neural networks for a long time (French, 1991). Indeed, sparse internal representations eventually lead to updates in fewer parameters, which (if partitioned correctly) could reduce the interference between them.

An alternative approach (Beaulieu et al., 2020) was recently proposed to improve OML by leveraging its representation sparsity. They suggest that it may be more effective to learn a modulation function f (a neural network) to modify the internal representations of a continually trained prediction network g. More formally, the inner loop is modified to train g on each pair (x_i, y_i), where $g(x_i) = g_p(f(x_i) \cdot g_e(x_i))$. In this context, we can think of g_e as a feature extraction or *encoding* layer, and of g_p as a *prediction* layer. The approach, called "a neuro-modulated meta-learning algorithm" (ANML), is motivated by the fact that f may learn to "block" certain gradient paths during back-propagation (thanks to the sparsity of $f(x_i)$), encouraging less parameter interference and hence less forgetting. A diagram comparing all of the mentioned approaches is shown in Figure 2.10 for convenience.

Although meta-learning approaches seem a promising direction in continual learning research, they still have two major disadvantages. First, given their computational cost, they are currently limited to simple scenarios in which sequences are short and training stages are small (*e.g.* only one example per training stage). Second, since they learn to avoid forgetting from data, they may be prone to overfitting to specific sequences or types of data. More generally, as with all data-driven approaches, they are subject to biases and assumptions in the training data that may not hold later in the production phase.

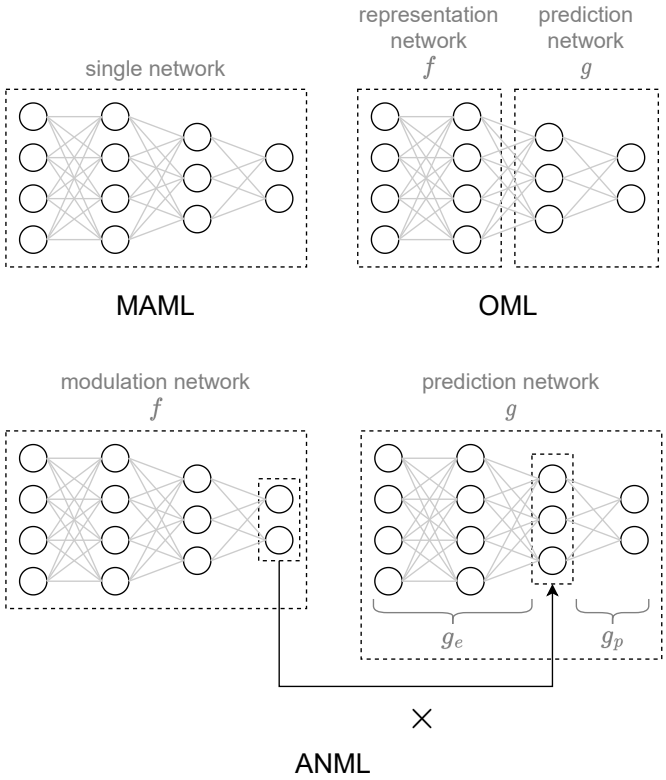


Figure 2.10: Architectural differences between MAML, OML and ANML. While MAML learns the initialization of the entire model for few-shot learning, OML learns a representation function for continual learning in the prediction network. In contrast, ANML learns a modulation network to modify the internal representations of the prediction network during continual training.

2.5 Conclusion

Throughout this chapter, we have introduced, defined and discussed a variety of concepts from three research areas that are key to this thesis: neural representations, transfer learning and continual learning.

In the rest of the manuscript, we heavily rely on many of these concepts to investigate ways in which models for spoken and written language can progressively learn from new data, even well after their initial development phase. In other words, we put a special focus on the production phase of these machine learning models, where data is usually unlabeled, and available progressively as time passes. In particular, transfer and continual learning are two subjects that appear regularly to help introduce more complex ideas.

On the one hand, in Chapter 3 and Chapter 5 we study and exploit task-specific neural representations, especially for speaker embeddings, which are crucial in the context of speaker recognition applications. On the other hand, the studies presented in Chapter 4, Chapter 5 and Chapter 6 dive deeper into continual learning for language-related applications, while borrowing key concepts from transfer learning as they are described in this chapter.

The background work specific to the various tasks and applications we address throughout the manuscript is presented and discussed in the corresponding chapters. Previous work on speaker verification and misogyny categorization is presented in Chapter 3. Sentence labeling, and in particular slot filling and named entity recognition are discussed in Chapter 4, while speaker diarization is discussed in Chapter 5 and Chapter 6.

Chapter 3

The quest for task-specific representations

> *"If we didn't have concept cells we wouldn't have representations of concepts, irrespective of specific details, we wouldn't be able to think."*
>
> — Rodrigo Quian Quiroga

We begin our studies by attempting to learn representations that are tailored to a specific task. In particular, we focus on the speaker verification and misogyny categorization tasks to make a systematic comparison of metric learning loss functions, whose goal is to encourage models to learn such representations.

The chapter is structured as follows. First, we introduce the problem we address in Section 3.1. In Section 3.2, we define the loss functions that we compare in the study. Next, along sections 3.3 and 3.4, we describe the tasks that we address along with their respective corpora, and we present our experiments and results. Finally, we discuss our conclusions with respect to the results obtained on each task.

The work presented in this chapter has been the subject of the two following scientific publications:

- **Juan M. Coria**, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. A Comparison of Metric Learning Loss Functions for End-to-End Speaker Verification. In *Statistical Language and Speech Processing*, Online, 2020. Springer International Publishing.

- **Juan M. Coria**, Sahar Ghannay, Sophie Rosset, and Hervé Bredin. A Metric Learning Approach to Misogyny Categorization. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, Online, 2020. Association for Computational Linguistics.

3.1 Introduction

Today, one of the most popular methods for learning representations of data is transfer learning with large neural network models, whose training scheme is divided in two stages: the pre-training stage and the fine-tuning stage. The initial pre-training stage consists in training a model from scratch using a large corpus. Since annotations are unavailable due to the size of the training set, this is generally achieved by devising general-purpose and self-supervised tasks that require a high level understanding of the underlying data. For example, the masked language modeling (MLM) task (Devlin et al., 2019) consists in predicting a word from the input sentence that has been masked. In the field of speech processing, the autoregressive predictive coding (APC) task (Chung and Glass, 2020) consists in predicting future frames of the audio input. Once the model is optimized for the chosen general-purpose task, any task-specific layers are discarded, resulting in a model that can produce complex and high-level representations, also known as embeddings. However, since the interpretability of pre-trained self-supervised embeddings is limited, further training is needed to solve a particular task. Consequently, the final fine-tuning stage consists in training the pre-trained representation model on a more specific downstream task, for which a small but labeled corpus is typically available.

One of the difficulties of using these embeddings to solve the continual learning problem is that they are subject to catastrophic forgetting (French, 1999), as they heavily rely on the fine-tuning stage. In contrast, metric learning investigates ways to learn neural representations that can be compared with a simple and well-defined distance function such as the euclidean distance. Although these representations can be refined if needed, they do not rely on further learning, which makes them an interesting candidate to achieve continual learning with limited forgetting.

Despite its name, the goal of metric learning is to map a pre-defined distance function d to an intrinsic property of the underlying data, such as semantic or speaker similarity. The goal is to obtain a mapping f from an input example x_i to a representation space $\mathbb{R}^m$ with the following property: given an *anchor* sample x_a belonging to a class, any *positive* sample x_p belonging to the same class should be closer to x_a than any *negative* sample x_n belonging to a different one:

$$d(f(x_a), f(x_p)) < d(f(x_a), f(x_n)) \tag{3.1}$$

Going one step further, metric learning aims at achieving intra-class compactness and inter-class separability, which means that class clusters should be as compact and distant from each other as possible. This concept is depicted in Figure 3.1, where d is defined as the euclidean distance.

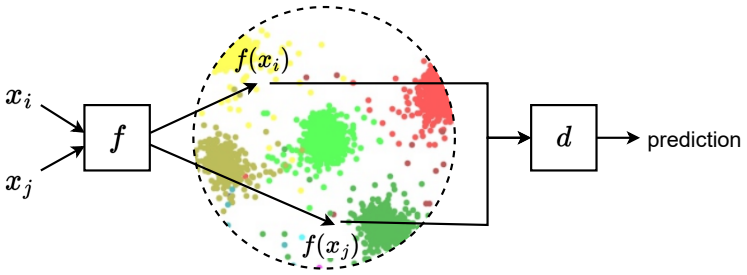


Figure 3.1: Given training samples x_i and x_j, the neural network f produces representations $f(x_i), f(x_j) \in \mathbb{R}^m$ such that the distance $d(f(x_i), f(x_j))$ between them is small if they belong to the same class (compactness) and large otherwise (separability).

However, since mapping f needs to exploit properties of the training data that are relevant to the target task, embeddings learned with metric learning techniques are generally tailored to a specific task instead of being general-purpose. In order to learn the representation function f from data, the typical approach is to train a neural network with a modified loss function that penalizes the model whenever the intra-class compactness and inter-class separability requirements are not satisfied.

A number of loss functions have been proposed to train such representation functions. While these approaches were mostly introduced for computer vision (Hadsell et al., 2006) and facial recognition in particular (Schroff et al., 2015; Liu et al., 2017; Wen et al., 2016; Deng et al., 2019), they have been rapidly adopted in other language-related domains, like speaker verification (Bredin, 2017b; Chung et al., 2018), language identification (Gelly and Gauvain, 2017) and sentence embedding (Reimers and Gurevych, 2019).

In this chapter, we attempt to learn task-specific representations using metric learning techniques. We make a systematic comparison of several metric learning loss functions from the literature focusing on two multi-class classification tasks: misogyny categorization from tweets in written language, and speaker verification from short audio extracts in spoken language. By choosing these two tasks, we evaluate the resulting embeddings on both unseen examples of a given class set

(closed-set misogyny categorization) and unseen classes (open-set speaker verification).

Our results shine light on our understanding of metric learning as well as on its applicability to the continual learning problem. In particular, we show (like others did before us for face recognition (Srivastava et al., 2020)) that in speaker verification the additive angular margin loss is better with respect to all considered criteria. More generally, margin-based loss functions lead to representations that can be compared directly without heavy back-end computations. Furthermore, we set new state-of-the-art performance for the misogyny categorization task, and we provide empirical evidence that metric learning may not constitute an advantage over a simple cross entropy loss on closed-set classification.

3.2 Metric learning loss functions

This section defines the loss functions considered in the study and divides them in two families: the ones relying on classification with cross entropy loss, and the ones that rely on a pre-defined notion of similarity between examples.

3.2.1 Classification-based

The first family of loss functions is derived from the cross entropy loss $\mathcal{L}_{\text{CE}}$, initially introduced for multi-class classification:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\frac{\exp(\sigma_{iy_i})}{\sum_{k=1}^{K}\exp(\sigma_{ik})}\right] \tag{3.2}$$

where N is the number of training examples (here, audio segments or textual sentences x_i), K the number of classes (here, speakers or misogyny categories) in the training set, y_i the class of training sample x_i, and σ_i is the output of a linear classification layer with weights $C \in \mathbb{R}^{m\times K}$ and bias $b \in \mathbb{R}^K$:

$$\sigma_i = f(x_i)\cdot C^T + b \tag{3.3}$$

We can facilitate the comparison with other metric learning loss functions thanks to the geometric interpretation of the euclidean dot product, which states that given vectors $x, z \in \mathbb{R}^m$ and the angle θ_{xz} between them, the following equality holds:

$$x \cdot z = \|x\| \, \|z\| \cos \theta_{xz} \tag{3.4}$$

Hence, we rewrite Equation 3.3 as follows:

$$\forall k \;\; \sigma_{ik} = \|f(x_i)\| \cdot \|c_k\| \cdot \cos \theta_{ic_k} + b_k \tag{3.5}$$

where θ_{ic_k} is the angular distance between the representation $f(x_i)$ of training sample x_i, and c_k the k$^{\text{th}}$ row of matrix C. Furthermore, by removing the bias parameter, we can interpret the minimization of the cross entropy loss as the maximization of the cosine similarity between the representation and the row of the weight matrix C associated to its class:

$$\forall k \;\; \sigma_{ik} = \|f(x_i)\| \cdot \|c_k\| \cdot \cos \theta_{ic_k} \tag{3.6}$$

In this context, the kth row of matrix C can then be seen as a canonical representation, or centroid, of the kth class. If we choose d to be the cosine distance $d(f(x_i), f(x_j)) = 1 - \cos(\theta_{ij})$, maximizing the cosine similarity between $f(x_i)$ and c_{y_i} while minimizing its distance to other c_k seems an adequate training objective to achieve intra-class compactness and inter-class separability. This is the idea behind the congenerous cosine loss (Liu et al., 2017), which forces the model to rely only on the cosine between between $f(x_i)$ and c_k:

$$\forall k \;\; \sigma_{ik} = \alpha \cdot \cos \theta_{ic_k} \tag{3.7}$$

where the scaling hyper-parameter α further penalizes errors and favors correct predictions. As depicted in Figure 3.2, this simplification enforces the desired properties of metric learning with the help of class centroids that are jointly trained to attract the embeddings from their associated class.

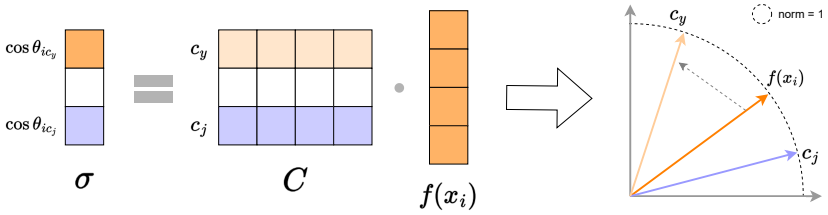


Figure 3.2: The congenerous cosine loss is a geometrical reinterpretation of classification with cross entropy.

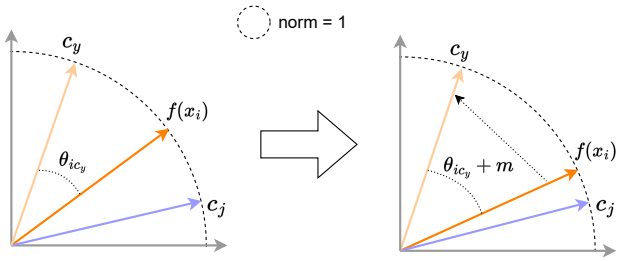


Figure 3.3: In additive angular margin loss, the angular distance between $f(x_i)$ and its centroid c_y is penalized to achieve stronger intra-class compactness.

The additive angular margin loss (Deng et al., 2019) goes one step further by introducing a margin to penalize the angular distance between $f(x_i)$ and c_{y_i}:

$$\forall k \quad \sigma_{ik} = \begin{cases} \alpha \cdot \cos(\theta_{ic_k} + m) & \text{if } y_i = k \\ \alpha \cdot \cos \theta_{ic_k} & \text{otherwise} \end{cases} \tag{3.8}$$

where m is the margin. As shown in Figure 3.3, this loss encourages embeddings to be closer to their centroids more explicitly by artificially augmenting their distance by the margin.

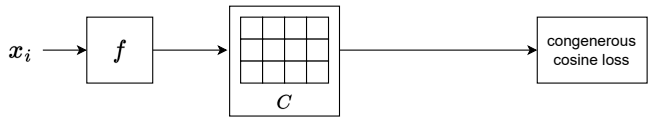
Finally, the center loss (Wen et al., 2016) takes a different approach by adding a term to the cross entropy loss to penalize the distance between a representation $f(x_i)$ and an external centroid $\gamma_{y_i} \in \mathbb{R}^m$ of its class y_i that is jointly trained but not part of the neural network:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \frac{\lambda}{2} \sum_{i=1}^{N} 1 - \cos \theta^2_{i\gamma_{y_i}} \tag{3.9}$$

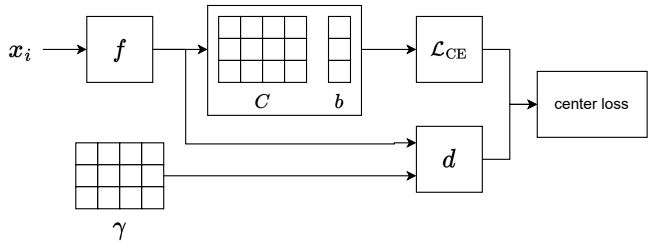
As depicted in Figure 3.4, while the congenerous cosine and additive angular margin loss are restricted to rely on the cosine distance (because of their derivation from cross entropy), center loss is capable of optimizing embeddings for any differentiable distance function, as the centroid matrix is external. This property grants center loss a superior flexibility.

3.2.2 Contrast-based

While classification-based loss functions assume that the class of each training sample is known, this second family of loss functions relies solely on *same/different*



(a) Embedding model using congenerous cosine loss.



(b) Embedding model using center loss.

Figure 3.4: Comparison of congenerous cosine loss (a) and center loss (b). In center loss, the centroid matrix γ is external to the classification layer and C rows cannot be interpreted as centroids.

binary annotations: given a pair of training samples (x_i, x_j), the pair is said to be positive when $y_i = y_j$ and negative otherwise.

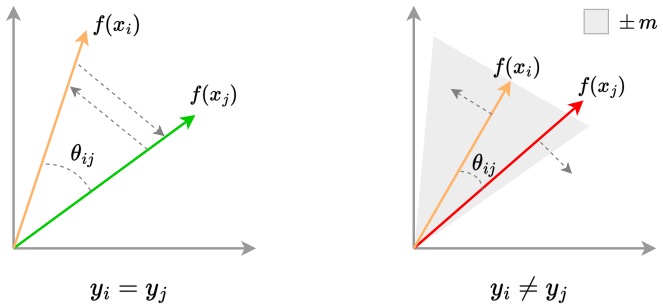


Figure 3.5: In contrastive loss, embeddings considered similar ($y_i = y_j$) are encouraged to be closer together, while those considered different ($y_i \neq y_j$) are encouraged to be distant from each other.

As shown in Figure 3.5, the contrastive loss (Hadsell et al., 2006) aims at making representations of positive pairs $\mathcal{P}$ closer to each other, while pushing negative pairs $\mathcal{N}$ farther away than a positive margin $m \in \mathbb{R}^+$:

$$\mathcal{L} = \sum_{(x_i,x_j)\in\mathcal{P}} d(f(x_i), f(x_j))^2 + \sum_{(x_i,x_j)\in\mathcal{N}} \max(m - d(f(x_i), f(x_j)), 0)^2 \tag{3.10}$$

where d is the cosine distance between $f(x_i)$ and $f(x_j)$. The effect of this formula can be visualized in Figure 3.6, where the loss encourages the distance of positive pairs to be low and the distance of negative pairs to be at least equal to m.

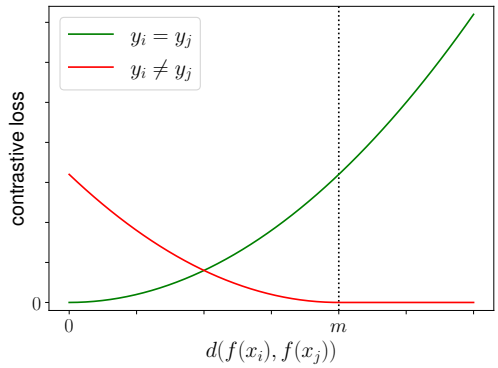


Figure 3.6: Behavior of the contrastive loss for both positive (green) and negative (red) pairs. Notice that the loss value is the lowest when the distance between positives is low, and when the distance between negatives is higher or equal to the margin m.

The triplet loss (Schroff et al., 2015) is defined in a similar way, but relies on triplets $(x_a, x_p, x_n) \in \mathcal{T}$, such that $y_a = y_p$ and $y_a \neq y_n$:

$$\mathcal{L} = \sum_{(x_a,x_p,x_n)\in\mathcal{T}} \max(\cos\theta_{an} - \cos\theta_{ap} + m, 0) \tag{3.11}$$

This loss function aims at making the representation of positive sample x_p closer to the anchor sample x_a than the representation of any other negative sample x_n by a positive margin $m \in \mathbb{R}^+$. Figure 3.7 shows a simplified example of this behavior in two dimensions.

Because positive pairs $\mathcal{P}$, negative pairs $\mathcal{N}$ and triplets $\mathcal{T}$ need to be sampled from the training set, they bring an additional computational cost that may slow down the training process. Morever, many tuples may satisfy intra-class compactness and inter-class separability early during training. Since these tuples constitute a weak training signal, they may cause convergence issues (Schroff et al., 2015;

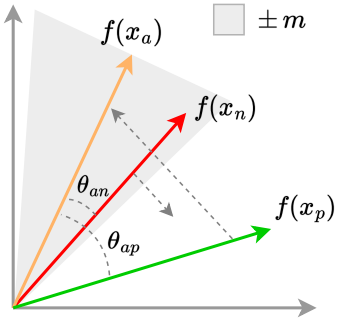


Figure 3.7: Given an anchor $f(x_a)$, a positive example $f(x_p)$ and a negative example $f(x_n)$, the triplet loss brings the anchor and the positive together and pushes the negative away in a single optimization step.

Hermans et al., 2017), especially if they constitute the majority of training examples. This problem is usually addressed with careful filtering in a process known as *mining*, which makes the whole process even more costly without any guarantee of training stability. Figure 3.8 shows several examples of easy and hard embedding pairs in two dimensions. The goal of mining algorithms is to select hard and meaningful examples to train the representation model more effectively.

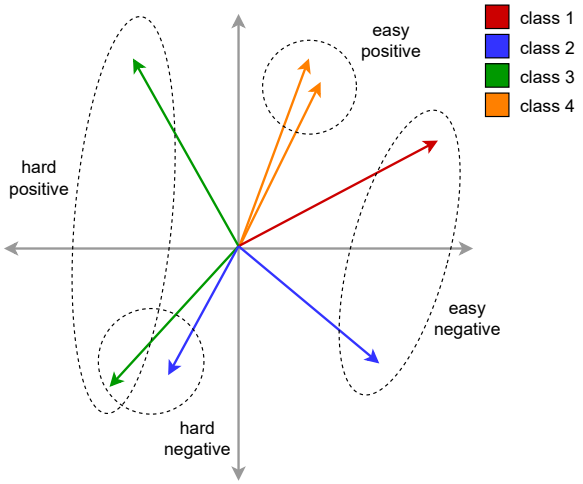


Figure 3.8: Easy and hard embedding pairs in two dimensions. Easy pairs are either negatives that are sufficiently distant or positives that are sufficiently close, while hard pairs are either close-together negatives or distant positives.

To circumvent these issues, we use a slightly modified version of the triplet loss in our experiments:

$$\mathcal{L}_T = \sum_{(x_a, x_p, x_n) \in \mathcal{T}} \text{sigmoid}(\alpha \cdot (\cos\theta_{an} - \cos\theta_{ap})) \tag{3.12}$$

where α plays the same role as in Equations 3.7 and 3.8. This idea was first introduced by Gelly and Gauvain (2017), and can be interpreted as an approximation of the area under the ROC curve (Mingote et al., 2020). We hypothesize that the use of sigmoid may force all triplets to provide a normalized training signal, making large errors saturate to 1. Getting rid of the positive truncation also ensures that positive pairs keep getting closer and negatives pairs farther apart, reducing the utility of the margin.

Note that while classification-based loss functions can only be used in a fully supervised setup, contrast-based loss functions that only rely on *same/different* labels can also be used in self-supervised scenarios (Pascual et al., 2019; Ravanelli et al., 2020) by exploiting prior knowledge about the data. A simple example of this concept could be the extraction of positive pairs from audiobook data, which is likely to contain single-speaker reading sessions.

3.3 Speaker verification experiments

In this section, we define the task of speaker verification. We introduce the corpus and model architecture on which we rely, and finally we present our experiments and results.

3.3.1 The speaker verification task

Given an utterance x and a claimed identity a, speaker verification aims at deciding whether to accept or reject the identity claim. It is a supervised binary classification task usually addressed by comparing the test utterance x to the enrollment utterance x_a pronounced by the speaker a whose identity is claimed. Speaker identification is the task of determining which speaker (from a predefined set of speakers $a \in S$) has uttered the sequence x. It is a supervised multi-class classification task addressed by looking for the enrollment utterance x_a the most similar to the test utterance x. After a preliminary speech segmentation step, speaker diarization aims at grouping speech turns according to the identity of the speaker. A diagram summarizing the difference between these three tasks can be seen in Figure 3.9.

Given that the definition of speaker verification is not tied to a set of pre-defined speaker identities (a property called *open-set*), it is imperative for trained models

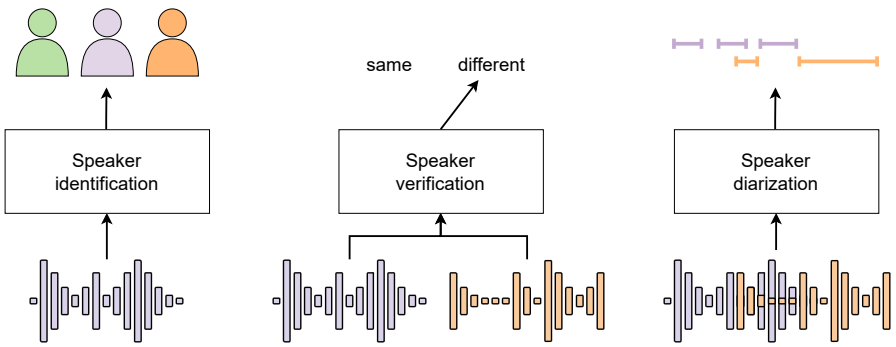


Figure 3.9: Comparison between speaker identification, speaker verification and speaker diarization.

to generalize to speakers never seen during training. This is typically addressed by training a mapping capable of projecting utterances to a representation space where embeddings from different identities are easily separable, assuming that the model can disentangle speaker identities from other acoustic information. Once the mapping is learned, a score representing the likelihood of two utterances belonging to the same speaker is estimated and a prediction is made based on a fixed threshold.

The *x-vector* approach (Snyder et al., 2018) is no exception to this framework. Its goal is to learn a representation function $f(x) = n(h(x))$, which is the composition of a hand-crafted feature extraction step h (e.g. filter banks or MFCC) and a neural network n trained for closed-set speaker recognition on a large collection of utterances from a large number of speakers. However, contrary to metric learning, its comparison function d is the composition of several post-processing steps.

First, a linear discriminant analysis (LDA) transform l and probabilistic LDA (PLDA) scoring p (Ioffe, 2006) are trained to increase the discriminant power of the embeddings, effectively boosting inter-class separability.

Second, a normalization function s for the predicted scores is estimated based on the mean and standard deviation of the scores with respect to an additional set of data, which is commonly referred to as a *cohort*. Score normalization is a fairly common practice that aims to reduce the effect of domain mismatch between training and test utterances, for example when they are recorded using different microphones (Rosenberg et al., 1992). In the x-vector approach, the authors choose to use the adaptive s-norm score normalization strategy (Matejka et al., 2017), which has been shown effective in a variety of scenarios. Instead of using a fixed cohort, this strategy is considered adaptive because it selects the top k highest (most positive) scores relative to each utterance in the test pair to compute the

mean and standard deviation.

Finally, the combination of these post-processing steps l, p and s leads to a distance function defined as:

$$d(f(x_i), f(x_j)) = s(p(l(f(x_i)), l(f(x_j)))) \tag{3.13}$$

Although most metric learning losses were designed for face verification and re-identification (Schroff et al., 2015; Liu et al., 2017; Deng et al., 2019), speaker verification suffers from much of the same difficulties. In the same way that face representations should be invariant to pose, lighting, or facial expression, utterances from a single speaker might differ in noise, phonetic content, or mood, among others. Achieving intra-class compactness and inter-class separability is in fact highly desirable in order for speaker embeddings to be robust to these variability factors. Moreover, an added benefit of metric learning is its ability to remove the need for costly post-processing steps like the LDA and PLDA (*i.e.* the back-end of the speaker embedding model). Recent work in speaker verification (Garcia-Romero et al., 2019) has even shown that a simple cosine distance scoring with a metric learning loss can perform equally or better than a PLDA scoring on top of the same architecture.

3.3.2 The VoxCeleb corpora

Our experiments on speaker verification are conducted using the VoxCeleb1 (Nagrani et al., 2017) and VoxCeleb2 (Chung et al., 2018) corpora, which contain recordings of single-speaker English utterances that were automatically extracted from Internet videos.

	VoxCeleb1	**VoxCeleb2**
Number of speakers	1251	6112
Number of male/female speakers	690/561	3761/2351
Duration (in hours)	352	2442
Number of utterances	153,516	1,128,246
Average number of utterances per speaker	116	185
Average utterance duration (seconds)	8.2	7.8

Table 3.1: VoxCeleb corpora as described by Chung et al. (2018).

Although refined for VoxCeleb2, the pipeline for the extraction of videos and utterances follows the same principle in both corpora. After selecting a list of n candidate speakers, the top k YouTube videos for each speaker are downloaded, where

the search term used is the speaker name with the word "interview" appended (to maximize the probability of finding videos with speech). Finally, a face tracking and recognition system identifies the faces in each frame of the video, followed by an active speaker detection step that determines when the target person is speaking based on the audio and mouth motion. Although the data is collected in a fully automated way, the authors make sure to reduce the number of false positives by setting conservative thresholds for the speaker detection and face recognition systems, raising their precision to 1 at the expense of recall.

The resulting corpora can be considered fairly balanced in terms of gender, ethnicity, accent, profession and age. Moreover, utterances are extracted from videos spanning a variety of acoustic environments, ranging from professional studios to outdoor stadiums. Additional information about the composition of these corpora is shown in Table 3.1.

3.3.3 Model architecture

An additional contribution of our study on speaker verification is a step towards the definition of a truly (front) end to (back) end neural speaker verification approach. On the back end of the original x-vector approach, every one of LDA transform l, PLDA scoring p and adaptive s-norm score normalization s needs its own (ideally disjoint) set of training data, making the approach quite complex and data-hungry. This is why we define our architecture with only the cosine distance for scoring.

Some end-to-end architectures in the literature avoid PLDA as well (Zhang and Koishida, 2017; Li et al., 2018; Wan et al., 2018) but still rely on hand-crafted features. Therefore, on the front-end, we combine SincNet (Ravanelli and Bengio, 2018) trainable feature extraction with the x-vector network architecture to build a fully trained representation function f that processes the waveform directly and does not rely on hand-crafted features: $f(x) = n(h(x))$ becomes $f(x) = n(x)$. SincNet features have proven to outperform handcrafted features for some tasks (Pascual et al., 2019; Ravanelli et al., 2020) and have been used in conjunction with an angular margin loss in (Chagas Nunes et al., 2019) for speaker recognition on the simple telephone corpus TIMIT (Garofolo et al., 1993), which contains 3s-long (on average) speech recordings of 630 different speakers.

To summarize, the network architecture used in this set of experiments combines SincNet trainable feature extraction (Ravanelli and Bengio, 2018) with the standard x-vector architecture (Snyder et al., 2018) to build a fully end-to-end speaker verification system. Both SincNet and x-vector use the configuration proposed in their respective papers (except for the SincConv layer of SincNet that uses a stride of 5 for efficiency).

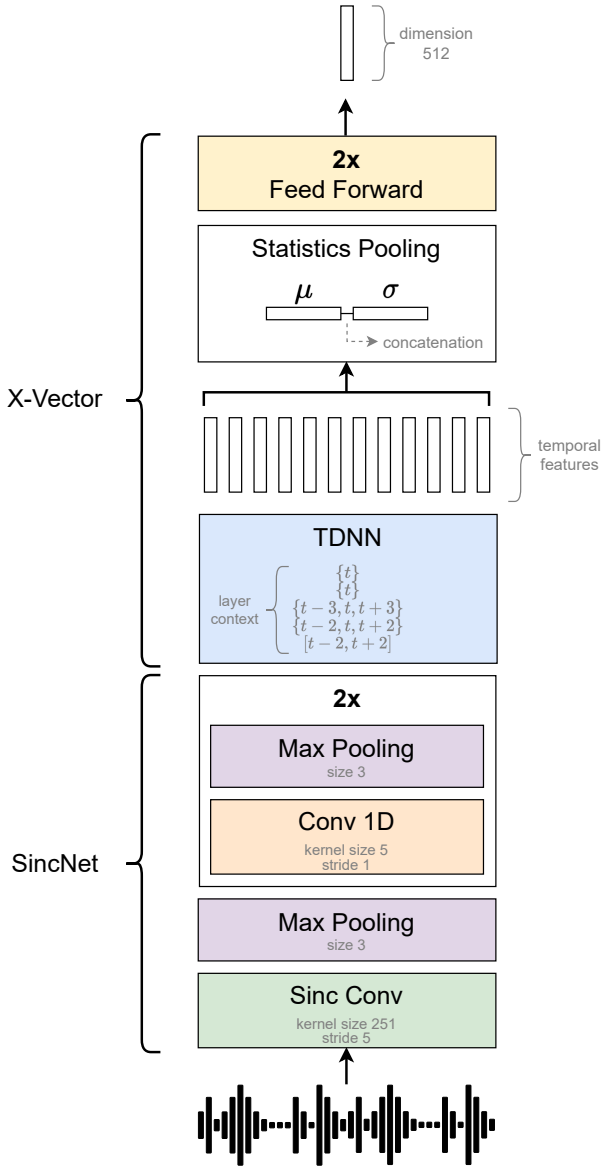


Figure 3.10: End-to-end model architecture combining SincNet trainable features with the standard TDNN x-vector architecture.

As depicted in Figure 3.10, the network takes the waveform as input and returns 512-dimensional speaker embeddings. In practice, we use a 3s-long sliding window with a 100ms step to extract a sequence of speaker embeddings that are then averaged to obtain just one speaker embedding per file. These average speaker embeddings are then simply compared with the cosine distance.

3.3.4 Evaluation

We evaluate our speaker verification model with the equal error rate (EER), defined as the value at which the false positive rate (*i.e.* different speakers classified as the same) equals the false negative rate (*i.e.* same speakers classified as different). The EER can be explained more simply with the help of the detection error trade-off (DET) curve (Martin et al., 1997), which relates false positives and false negatives at multiple decision thresholds in logarithmic scale, as depicted in Figure 3.11.

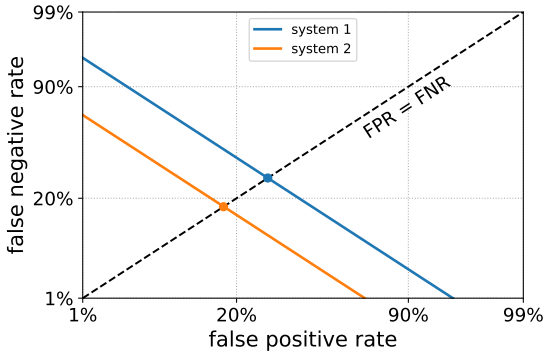


Figure 3.11: Detection error trade-off curves for two verification systems. The equal error rate is shown as a dot over the system's curve. Note that each point forming these curves corresponds to a different threshold value. FPR and FNR denote false positive and false negative rates, respectively.

As a result, the EER can be interpreted as the best compromise between false positives and false negatives when both are equally important for the target application. Since a lower EER is better, system 2 in Figure 3.11 would be considered better than system 1.

3.3.5 Training protocol

The whole VoxCeleb 2 development set (5994 speakers) serves as our training set. The VoxCeleb 1 development set (1211 speakers) is split into two parts: 41 speakers (whose name starts with U, V, or W) serve as our development set (1000 trials per speaker), the remaining 1170 speakers are used as training data for adaptive s-norm

score normalization. Final evaluation is performed on the official VoxCeleb 1 test set, containing a total of 40 speakers and 37720 trials (equally distributed between *same/different* speakers).

The optimal hyper-parameters for each loss function are selected with a grid search by training the model with each configuration for 20 hours and evaluating it on the development set. The configuration with the best performance is selected and used for further training for a total of 200 hours. Once training is completed, we choose the model with the best performance on the development set. Finally, we apply the resulting model on the VoxCeleb 1 official test set (40 speakers) and report the equal error rate and corresponding 95% confidence interval computed with the FEERCI package (Haasnoot et al., 2018).

Contrary to other normalization strategies, the adaptive s-norm score normalization selects a subset of its training data to compute the mean and variance. In fact, only the top k highest scores (in our case lower distances) are selected for each speaker. This is why we report the equal error rate after adaptive s-norm score normalization with a cohort size k tuned on the development set.

3.3.6 Implementation details

All models were trained on a GPU (NVIDIA Tesla V100) with stochastic gradient descent using a fixed learning rate selected during the initial hyper-parameter grid search: we tried 10^{-3}, 10^{-2}, and 10^{-1}. Mini-batches were built by stacking 3s audio chunks extracted randomly from the training set, making sure each speaker was equally represented.

Following lessons learned by others (Mclaren et al., 2018), on-the-fly augmentation was used by dynamically adding random background noise from the MUSAN (Music, Speech and Noise) corpus (Snyder et al., 2015) with a random signal-to-noise ratio between 10 dB and 20 dB. This corpus contains 60 hours of speech (discarded in our experiments), 42 hours of music and 6 hours of background noise (*e.g.* dial tones, car diling, footsteps, rain, etc.).

For classification-based loss functions, the batch size was fixed to 128 (from 128 different speakers). For contrast-based loss functions that expect pairs (or triplets) of training samples, a fixed number of audio chunks from a fixed number of different speakers were stacked to build mini-batches, before forming all possible pairs (or triplets) for each batch. Both numbers were added to the set of hyper-parameters for these loss functions: we tried 20 and 40 for the number of speakers per batch, 2 and 3 for the number of audio chunks per speaker.

3.3.7 Results and discussion

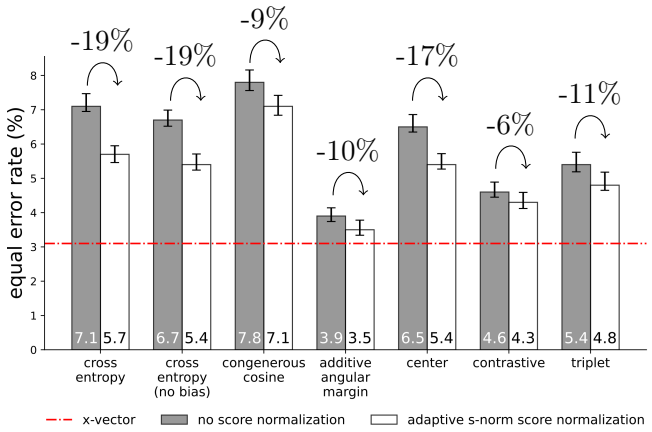


Figure 3.12: Speaker verification equal error rate on the official test set of VoxCeleb 1 (lower is better), with and without adaptive s-norm score normalization. Relative improvement brought by score normalization is reported with curved arrows at the top. 95% confidence intervals are depicted as vertical error lines. Performance of the x-vector baseline (Snyder et al., 2018) is shown for reference.

Performance. Figure 3.12 summarizes the performance of the proposed end-to-end speaker verification architecture when trained with each loss function. We report the equal error rate on the test set of VoxCeleb 1. The provided 95% confidence intervals show that additive angular margin loss significantly outperforms all other loss functions. When combined with adaptive s-norm score normalization, it is even competitive with respect to the x-vector baseline performance reported by Snyder et al. (2018), that relies on hand-crafted features (and for which we could not compute confidence intervals without access to the system's output).

Speaker embedding robustness. A closer look at the relative improvement brought by the score normalization step shows that additive angular margin loss and contrastive loss are the only ones for which the difference is not statistically significant. This suggests that the use of a margin leads to representations that are both better (in terms of performance) and more robust to domain mismatch. In particular, the latter is an extremely relevant property that could facilitate continual adaptation to new domains without the need for further fine-tuning.

Training time. Despite training each variant for 200 hours each on the speaker verification task, some of them were still improving on the development set when the time limit was reached: the congenerous cosine loss and the contrast-based (contrastive and triplet) losses. A closer look at convergence time with respect to the amount of data seen by each model is presented in Table 3.2. We observe that

Loss function	**Epochs**	**Audio chunks** (in millions)	**Time**
Cross entropy	208	30	60h
Center	240	35	70h
Additive angular margin	560	81	160h
Triplet	680	98	>200h
Congenerous cosine	709	102	>200h
Contrastive	846	122	>200h

Table 3.2: Convergence time for each loss on speaker verification in terms of examples seen and training time. The contrastive loss, congenerous cosine loss and triplet loss need more training time and examples to converge.

the difference in training time is also correlated to the amount of audio chunks seen, which shows that in general this slower convergence is due to the need for more training examples rather than implementation differences. While the relative slowness of the contrastive loss and triplet loss can be explained by the lack of previous tuple mining, we are still unsure as to why congenerous cosine is so slow.

Loss function	**Hyper-parameter**	**Value**
Cross entropy	learning rate	10^{-1}
Congenerous cosine	learning rate	10^{-1}
	α	10
Additive angular margin	learning rate	10^{-2}
	α	10
	m	0.05
Center	learning rate	10^{-1}
	λ	1
Contrastive	learning rate	10^{-1}
	m	0.2
	other	3 chunks $\times$ 20 spk
Triplet	learning rate	10^{-2}
	α	10^{*}
	other	3 chunks $\times$ 40 spk

Table 3.3: Best hyper-parameter configurations per loss for speaker verification. Hyper-parameters m, α and λ are loss-specific (see Section 3.2). Values marked with * were not tuned.

Optimal hyper-parameters. Table 3.3 shows the best hyper-parameters found for each of the considered loss functions. In particular, we observe that additive angular margin loss works best with a lower margin than contrastive loss. This

is consistent with the margin's role, serving as an upper bound for the distance between an embedding and its centroid, while the margin in contrastive loss serves as a lower bound for the distance between two negatives.

3.4 Misogyny categorization experiments

In this section, we define the task of misogyny categorization. We introduce the corpus and model architecture used in our experiments, and we present and discuss the results we obtain.

3.4.1 The misogyny categorization task

The term misogyny is defined as hatred towards women. Hate speech of this nature is unfortunately common in social Internet interactions, and current language models are generally unable to accurately detect and classify it.

One of the goals of the IberEval 2018 (Fersini et al., 2018b) and Evalita 2018 (Fersini et al., 2018a) evaluation campaigns was to address misogyny on tweets. Included tasks were identification (binary classification: misogynous or not), categorization over five different misogyny types (multi-class sentence classification), and target identification (binary classification: to an individual or a group).

As shown in Figure 3.13, our study focuses on classifying misogyny using an additional class for the absence of misogyny, as we consider it suitable to study textual sentence representations. This corresponds to the misogyny categorization part of subtask B of the Evalita 2018 campaign.

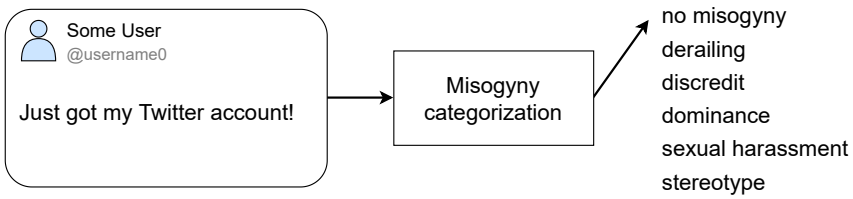


Figure 3.13: The misogyny categorization task (Evalita 2018 subtask B (Fersini et al., 2018a)).

However, no participant has proposed a model trained with a metric learning objective. The best system (Ahluwalia et al., 2018) uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with word embeddings of size 100 for the identification task, and ensemble methods with feature engineering for category and

target classification. They achieve a macro F1 score of 36.1 on the misogyny categorization part of sub-task B, which is the one we address as well. A different architecture (Caselli et al., 2018) uses a multi-layer character bidirectional LSTM for categorization, obtaining a macro F1 score of 14.1.

Our motivation for the use of metric learning is that it might help to reduce the natural intra-class variability within misogyny categories, making representations robust to variation factors like writing style, irony, insults, etc.

3.4.2 The AMI corpus

Category	Description	Example
derailing	"to justify women abuse, rejecting male responsibility"	"if rape is real why aren't more people reporting it? just another feminist lie"
discredit	"slurring over women with no other larger intention"	"this b*** is a s***"
dominance	"to assert the superiority of men over women to highlight gender inequality"	"#didyouknow the male brain is 3.4 times larger than the female brain? #maledominance"
sexual harassment	"sexual advances, harassment of a sexual nature, etc."	"come on box I show you my c*** darling"
stereotype	"a widely held but fixed and oversimplified image or idea of a woman"	"these people are hysterical. it's like a commercial for why men should never marry [...]"

Table 3.4: Misogyny categories as described by the corpus authors (Fersini et al., 2018a) along with examples found in the training set.

The automatic misogyny identification (AMI) task and corpus were proposed in the context of the IberEval 2018 (Fersini et al., 2018b) and Evalita 2018 (Fersini et al., 2018a) evaluation campaigns. The corpus consists of an ensemble of tweets with three different types of annotations: misogyny (binary), misogyny category (multiclass) and target (active or passive).

In our experiments, we focus exclusively on misogyny categorization, which contains a total of 5 categories, plus an additional category for non misogynous tweets. A description of misogyny categories according to the definitions given by Fersini et al. (2018a) can be found in Table 3.4.

In order to collect misogynous tweets, the authors rely on three approaches. First, a search of relevant keywords often used to express misogyny (*e.g.* b***h, w**re, etc.). Second, monitoring the accounts of potential victims of misogyny, like public feminist figures. Third, using the history of accounts that have publicly declared hate against women in their profiles. Finally, collected tweets are manually labeled and submitted to a gold standard for quality control. The corpus is available in

English, Spanish and Italian, but we choose to work only on English for simplicity, which contains a total of 5000 tweets.

3.4.3 Model architecture

We experiment with two different encoder architectures. The first one is a single-layer bidirectional long short-term memory (LSTM) model (Hochreiter and Schmidhuber, 1997) with output size 768 and word embeddings of size 300 obtained from a word2vec continuous bag of words (CBOW) model (Mikolov et al., 2013) trained on 2-billion-word Wikipedia dumps. The second one is the standard monolingual uncased bidirectional encoder representations from Transformers (BERT) model (Devlin et al., 2019) from the huggingface library (Wolf et al., 2019) pre-trained on English Wikipedia and BooksCorpus (Zhu et al., 2015).

To obtain a sentence embedding from the model, we perform a max pooling over the hidden states of the last layer, leaving us with sentence embeddings of size 768 on both models. When optimizing classification-based loss functions, a linear classification layer is jointly trained with the sentence encoder. Both model architectures can be visualized in Figure 3.14.

3.4.4 Training protocol

As the corpus does not provide a development set, one was constructed from the training set following the same class distribution. The final training set is composed of 3200 tweets, and the development and test sets of 800 and 1000 tweets respectively. The distribution of classes is described in detail in Table 3.5.

class	**training**	**development**	**test**
derailing	74	18	11
discredit	811	203	141
dominance	118	30	124
sexual harassment	282	70	44
stereotype	143	36	140
non misogynous	1,772	443	540
total	3,200	800	1,000

Table 3.5: Number of sentences per class for each subset of the AMI corpus. Note that classes are greatly imbalanced.

Furthermore, since different losses rely on different hyper-parameters, we perform a hyper-parameter search including learning rates, margins m, scalings α, and λ

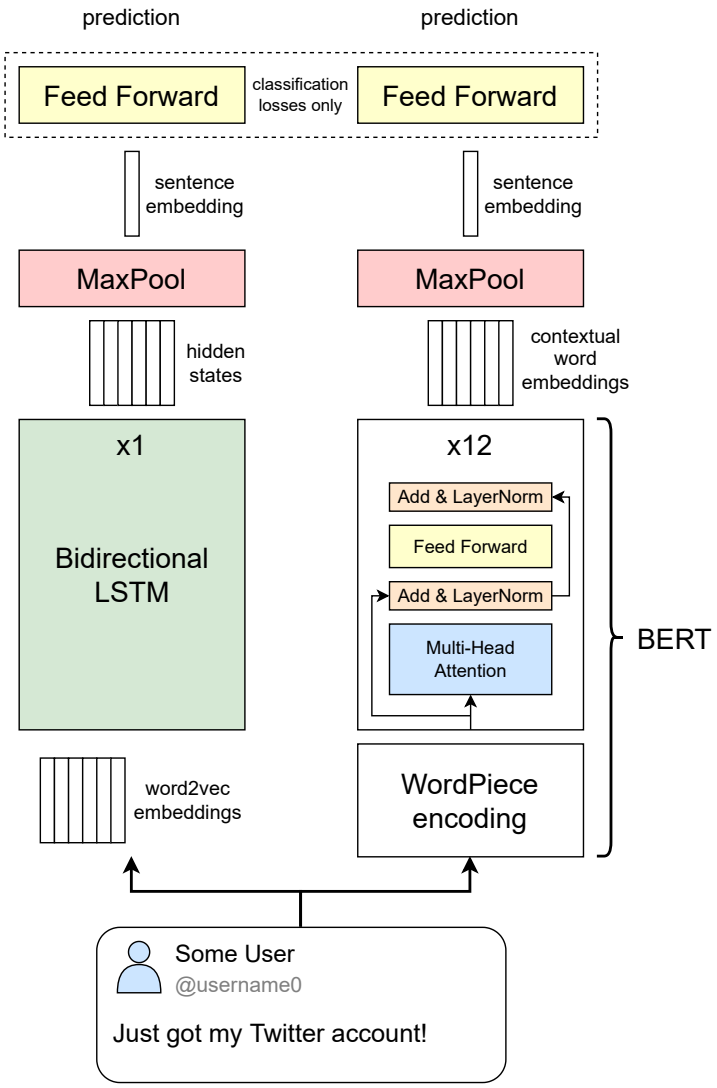


Figure 3.14: Bidirectional LSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2019) used in our misogyny categorization experiments.

(introduced in Section 3.2). The values we have experimented with are shown in Table 3.6. Each configuration is trained on the training set for 60 epochs and validated using a KNN classifier on the development set. As we deal with a rather small corpus, the best configuration for each loss and each architecture is then trained and validated from scratch 10 times to reduce the effect of randomness. Reported results are the mean macro F1 score and standard deviation on the test set over these 10 runs.

Hyper-parameter	**Values**
learning rate	$\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}^{\bullet}$ $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}^{\circ}$
m	$\{0.02, 0.05, 0.25, 0.5, 0.75\}$
α	$\{0.01, 0.1, 1, 10, 100, 1000\}$
λ	$\{0.01, 0.1, 1, 10, 100, 1000\}$

Table 3.6: Values tested during initial hyper-parameter search, totaling 486 configurations. Each one of m, α and λ are loss hyper-parameters (see Section 3.2). Values with $\bullet$ are LSTM only and values with $\circ$ are BERT only.

In all experiments we use the cosine distance to compare embeddings, as congenerous cosine loss and additive angular margin loss can only be optimized in this way. We evaluate the models with the macro F1 score of a KNN classifier with $K = 10$ fit with all sentence embeddings from the training set. However, the a priori probability of a random embedding being assigned to a given class may be affected by the high class imbalance. For example, it may be more likely to find a random embedding closer to a *non-misogynous* embedding than to a *discredit* one (see Table 3.5).

To circumvent this issue, we compute the prediction $\hat{y}$ for a given embedding as:

$$\hat{y} = \underset{c}{\operatorname{argmax}} \frac{v_c}{N_c^{\text{train}}} \tag{3.14}$$

where v_c is the number of votes for class c, and N_c^{train} the number of samples from c in the training set. We believe this simple classifier to be a better measure for representation quality, as it relates to the separability and compactness properties that we expect from a metric learning model.

3.4.5 Implementation details

In both models, sentences are pre-tokenized using the `TweetTokenizer` from the NLTK toolkit (Bird et al., 2009) in order to correctly deal with Twitter-specific tokens like hashtags, mentions, and even emojis. During this process we also remove handles and URLs.

When training BERT, we do a second pass of tokenization with BERT's pre-trained tokenizer based on WordPiece (Devlin et al., 2019), which relies on subwords instead of full words. For example, the tokenization of "this chapter" using WordPiece could be [this, chap, ##ter] instead of [this, chapter], where "##" denotes the continuation within a word.

Finally, we use a batch size of 32 sentences and RMSprop as optimizer, reducing the learning rate by half every 5 epochs if no improvement is detected.

3.4.6 Results and discussion

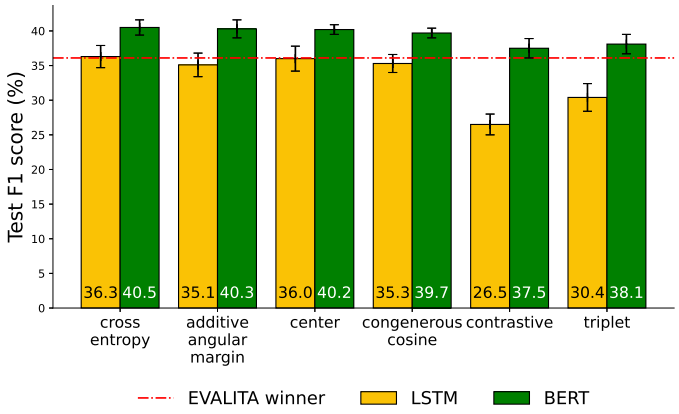


Figure 3.15: Misogyny categorization test F1 scores for each architecture and loss function (higher is better). Scores are calculated as the mean of 10 runs and standard deviation is shown as error bars. The baseline of the Evalita 2018 winner (Ahluwalia et al., 2018) is shown for reference.

Performance. Our main results are summarized in Figure 3.15. With a fixed architecture, it is clear that all loss functions perform equally on this task, with the exception of LSTM with contrastive and triplet loss. As the LSTM encoder is rather shallow (4.4M parameters) in comparison to BERT (110M parameters), it is possible that contrast-based losses need larger models to perform competitively. In contrast, our fine-tuned BERT outperforms the Evalita winner baseline (Ahluwalia et al., 2018) with a macro F1 score of 40.5, setting new state-of-the-art for misogyny

categorization, with the added benefit of providing embeddings comparable with a simple cosine distance.

Open-set vs closed-set. The fact that almost all losses perform equally well on misogyny categorization shows that, contrary to both our initial hypothesis and the results obtained on speaker verification, metric learning models perform no better than cross entropy in this case. This is also in stark contrast to other findings on face verification (Srivastava et al., 2020). One possible explanation is that the AMI dataset may not contain enough examples or classes for these models to exploit. However, another factor might be responsible for this behavior. One of the key characteristics of AMI with respect to speaker or face verification is the closed-set nature of the problem. An open-set task is evaluated with unseen *classes*, while a closed-set task is evaluated with unseen *instances* of the training classes. We hypothesize that open-set verification tasks are more suitable for metric learning than closed-set tasks, meaning that the power of metric learning might in fact lie in generalizing to unseen classes rather than unseen class instances. The fact that verification tasks more closely resemble the training objective than exact class prediction could provide an explanation for this.

Optimal hyper-parameters. As a final note, the results of hyper-parameter optimization (shown in Table 3.7) suggest that congenerous cosine loss and center loss hyper-parameters could be more sensitive to architecture changes than other losses, as they are the only ones whose best configurations differ across architectures within the same task. Perhaps not surprisingly, we also observe that, as in speaker verification, additive angular margin loss works better with lower margins.

Loss function	**Hyper-parameter**	**AMI LSTM**	**AMI BERT**
Cross entropy	learning rate	10^{-3}	10^{-5}
Congenerous cosine	learning rate	10^{-3}	10^{-5}
	α	10	100
Additive angular margin	learning rate	10^{-3}	10^{-5}
	α	100	100
	m	0.05	0.05
Center	learning rate	10^{-4}	10^{-5}
	λ	1000	0.1
Contrastive	learning rate	10^{-4}	10^{-6}
	m	0.25	0.25
Triplet	learning rate	10^{-4}	10^{-6}
	α	1000	1000

Table 3.7: Best hyper-parameter configurations per loss for misogyny categorization models. Hyper-parameters m, α and λ are loss-specific (see Section 3.2).

3.5 Conclusion

In this chapter, we performed a systematic comparison of several metric learning loss functions representing different strategies towards achieving intra-class compactness and inter-class separability in neural representations.

Overall, no matter the comparison criterion (performance, robustness, or training time), the additive angular margin loss is always better than the other loss functions that were considered in speaker verification experiments. If we had to find one drawback, this would be the fact that it can only be used in a fully supervised learning scenario (contrary to its closest competitor: the contrastive loss). On the other hand, contrary to what we thought, none of the considered losses can outperform the regular cross entropy on the task of misogyny categorization.

After extensive experiments on both speaker and sentence embedding, we reach the following conclusions. First, metric learning approaches may be better suited to open-set classification tasks, where the model must generalize to unseen classes (like new speakers or faces), instead of generalizing to new examples from the same classes. This motivates the subject of the next chapter, where we study the continual adaptation of contextual word embeddings to new languages in sequence labeling tasks.

Second, on open-set speaker verification, margin losses like additive angular margin loss encourage models to learn representations with better intra-class compactness and inter-class separability, as evidenced by the achieved performance based solely on the cosine distance. This seems to confirm our initial hypothesis that exploiting task-specific representations can constitute an advantage in some continual learning applications, which motivates our decision to leverage metric learning speaker embeddings for streaming speaker diarization in Chapter 5.

Our results on speaker verification also suggest that additive angular margin embeddings may be more robust to domain mismatch, an extremely relevant property when considering continual learning during the production phase.

Chapter 4

Continual word embedding adaptation

"What matters is not how much we remember, but how we remember."

— Rodrigo Quian Quiroga, *The Forgetting Machine*

In the previous chapter we sought to obtain sentence representations tailored to the task of misogyny categorization with the properties of intra-class compactness and inter-class separability. Although our motivation was to leverage sentence embeddings for continual adaptation, we learned that such representations may not be particularly suited to closed-set tasks like misogyny categorization.

In this chapter we take a step back from our initial hypothesis to study continual adaptation for the closed-set task of sequence labeling, where the goal is to predict a class for each word in a sentence. Instead of explicitly enforcing geometric properties, we perform a wide range of experiments to better understand the transfer capabilities of contextual word embeddings as they are progressively adapted to new languages.

Continual adaptation is a highly desirable property of language models because they may need to be updated with new user requirements or previously unavailable information. For example, a model trained to answer questions based on news articles may be regularly updated with the latest available information. In particular, we consider the progressive adaptation to new languages to be an extremely relevant problem applicable to a multitude of natural language processing tasks. For instance, the same question-answering model may be progressively deployed

in multiple countries as it gains popularity.

Finally, our study is also motivated by the fact that the relationship between contextual word embeddings from large Transformer models (Vaswani et al., 2017) and catastrophic forgetting has not received as much attention as other simpler problems in the continual learning literature.

The chapter is structured as follows. Through sections 4.2 to 4.4, we describe the target tasks and their corpora, as well as the model architecture we work with and the metrics we rely on to measure transfer and forgetting. Our main research question concerning the existence of cross-lingual transfer is addressed in Section 4.5, and in Section 4.6 we perform extensive experiments to understand how such transfer is affected by the training sequence. Finally, in Section 4.7 we investigate whether lost performance (due to forgetting) can be recovered and at what cost.

This work is the result of a collaboration with Mathilde Veron, another PhD student at the LISN laboratory, whose work focuses on life-long learning in the context of task-oriented dialogue systems.

This chapter has been the subject of the following scientific publication:

Juan M. Coria[1], Mathilde Veron[1], Sahar Ghannay, Guillaume Bernard, Hervé Bredin, Olivier Galibert, and Sophie Rosset. Analyzing BERT Cross-Lingual Transfer Capabilities in Continual Sequence Labeling. In *Proceedings of the First Workshop on Performance and Interpretability Evaluations of Multimodal, Multi-purpose, Massive-Scale Models*, Online, 2022. International Conference on Computational Linguistics.

4.1 Introduction

In the previous chapter, we experimented with a popular pre-trained Transformer-based (Vaswani et al., 2017) language model called BERT (Devlin et al., 2019) for sentence embedding. These model architectures have proven to perform extremely well on several natural language processing (NLP) tasks, often achieving state-of-the-art performance (Raffel et al., 2020; Brown et al., 2020). They are pre-trained with a self-supervised objective on large text corpora and rely on knowledge transfer for fine-tuning to a downstream task. Multilingual versions of these models have also been trained, and they have demonstrated high cross-lingual transfer as well (K et al., 2020; Wang et al., 2020; Conneau et al., 2020; Xue et al., 2021a).

[1]Equal contribution, order is alphabetical.

Given NLP tasks t_a expressed in language a and t_b expressed in language b, cross-lingual transfer can be defined as the performance improvement on t_b thanks to the knowledge acquired while learning t_a. Similar to our previous categorization of transfer learning (see Section 2.2), this can be defined as joint or sequential. Although large pre-trained models like BERT rely on sequential transfer in the task axis (from pre-training to fine-tuning), multilingual versions typically rely on a mix of joint and sequential transfer in the language axis, as shown in Figure 4.1. However, although joint training maximizes cross-lingual transfer, it assumes that data in all languages is available, which may not be the case in practice.

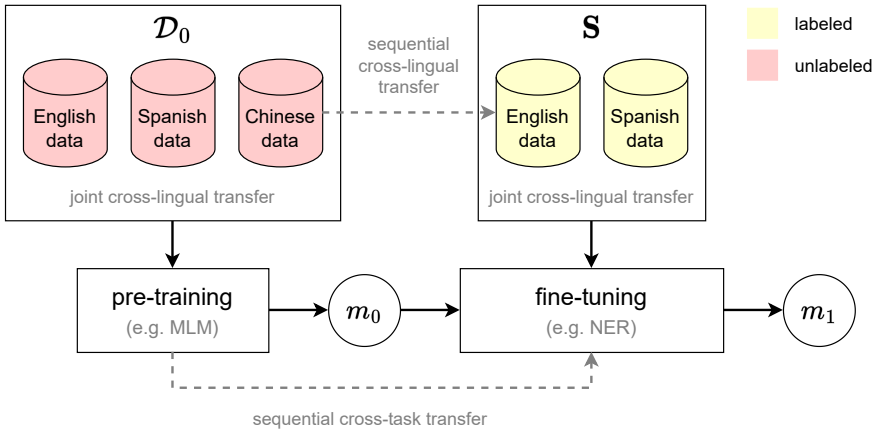


Figure 4.1: Mix of transfer styles present in large multilingual language models. While pre-training and fine-tuning tasks are generally different, multilingual training is often a mix of joint and sequential transfer. In this example, fine-tuning on $\mathbf{S}$ jointly after pre-training on $\mathcal{D}_0$ can be seen as joint transfer between English and Spanish, but also as sequential transfer from Chinese (in $\mathcal{D}_0$) to English and Spanish (in $\mathbf{S}$). We use m_i to denote the model resulting from training stage i.

Given the raising interest in these models to transfer knowledge between languages, it is of great importance to better understand the phenomenon of cross-lingual transfer as well as its limits. In the context of our study, we rely on a pre-training stage on a large $\mathcal{D}_0$ consisting of a mix of multiple languages, but study continual adaptation in the fine-tuning stage using $\mathbf{S}$: the sorted sequence of monolingual corpora $\mathcal{D}_{1 \leq i \leq L}$ from L different languages. To do this, we analyze the cross-lingual capabilities of multilingual BERT on two sequence labeling tasks, where each word of a sentence must be associated to a specific label.

Sequence labeling regroups various NLP tasks like named entity recognition (NER), part-of-speech (POS) tagging, text chunking and slot-filling. We focus our study on two of these tasks using two multilingual corpora: MultiATIS++ for slot-filling (Xu

et al., 2020) and MultiCoNER for NER (Malmasi et al., 2022a,b). Experimenting on different corpora allows us to identify which observations may generalize to other tasks and data, and which ones may be task-specific.

While most cross-lingual transfer studies about slot-filling or NER focus either on joint training or sequential training with source and target languages $\mathcal{D}_1$ and $\mathcal{D}_2$ (Xu et al., 2020; Schuster et al., 2019; Arkhipov et al., 2019; Mueller et al., 2020; Wang et al., 2020) (*e.g.* English → Spanish), our main contribution is a study with special focus on *continual* cross-lingual transfer, where the target task remains the same and the adaptation axis is defined over the sequence of languages $\mathbf{S}$ (*e.g.* Spanish → English → Italian → ...).

We believe this experimental setup to be interesting not only as a novel way of studying cross-lingual transfer but also because it is better suited to practical scenarios. As a matter of fact, adaptation to new data over time is a highly desirable feature of most NLP models: oftentimes, collecting data and annotating it is expensive, which makes training data scarce or incomplete in the development phase. Additionally, requirements might also evolve with time based on the needs of users. This means that the model needs to adapt sequentially as training data becomes available in production. An example of this could be a dialogue system that is gradually deployed in different countries. Unfortunately, naive solutions to adapt a pre-trained model are costly, as they require either re-training from scratch or maintaining many distinct models.

As discussed in Chapter 2, training a model on a sequence $\mathbf{S}$ of corpora $\mathcal{D}_i$ in multiple training stages is at the heart of continual learning (Hadsell et al., 2020), where the goal is to improve performance on both past and new data. In this chapter, we refer to $\mathbf{S}$ and the order of $\mathcal{D}_i$ as a *training sequence* for simplicity. An example of this is depicted in Figure 4.2.

Single-stage training schemes assume that training examples (in this case annotated sentences) are independent and identically distributed (*i.i.d.*), which does not usually hold when data becomes available sequentially. Moreover, access to previous data is not allowed[2], as this represents a linear use of resources with respect to the length of $\mathbf{S}$, which can in theory be infinite. In this context, measuring transfer is generally divided in two: forward and backward (Hadsell et al., 2020; Lopez-Paz and Ranzato, 2017; Arora et al., 2019), defined in our case as improvement on future and already acquired languages respectively. Note that forgetting can also be defined as negative backward transfer, as it represents the loss of previously acquired knowledge. While previous studies on continual learning tend to

[2]As we have discussed in Section 2.4, access to previous data is sometimes allowed if limited, for example in memory-based techniques (see Section 2.4.2)

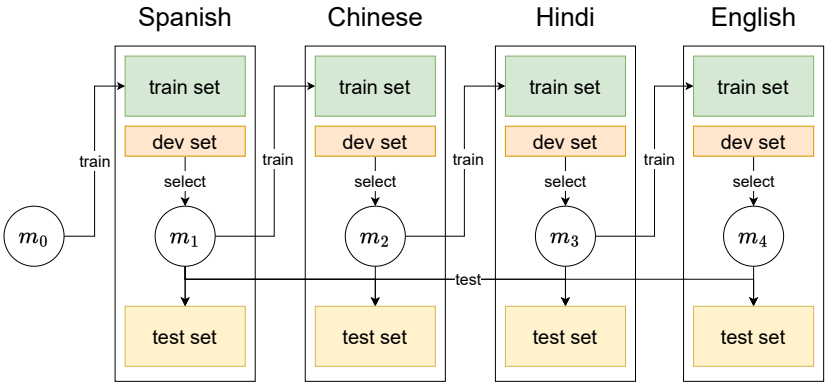


Figure 4.2: A training sequence across 4 languages. For each language in the given order, we train the model on its training set, select the best epoch on the development set and then test on all test sets independently. We use m_i to denote the model resulting from training on language at position i. Model m_0 denotes the initial pre-trained model.

focus on the domain axis for the slot-filling task (Lee, 2017; Madotto et al., 2021), or on the class axis for the NER task (Monaikul et al., 2021; Xia et al., 2022), we concentrate on the adaptation axis over languages.

Similar work also investigates cross-lingual transfer of multilingual BERT fine-tuned on sequence labeling tasks, namely NER and POS-tagging (Liu et al., 2021). They focus on preserving masked language modeling performance and cross-lingual ability after fine-tuning on one of the two tasks on English only. In stark contrast, our work focuses on fine-tuning on a single task over a sequence of many languages (*i.e.* it is *language-incremental* and not *task-incremental*) and addresses use cases where $|\mathbf{S}| > 2$.

4.2 Sequence labeling

The goal of sequence labeling tasks is to predict the correct label for each word in a sentence, which makes it appropriate to identify concepts or entities. In our study, the set of labels to predict is the same across languages so that the task remains unchanged over the continual learning process. In other words, we restrict our study to closed-set sequence labeling problems.

Sequences are typically labeled using the IOB format (Ramshaw and Marcus, 1995), where labels consist of a prefix (B, I or O) and an optional type that categorizes the identified concept. While O indicates that the word is not part of a concept (O for *outside*), B (for *beginning*) and I (for *inside*) indicate that it

is the beginning or continuation of a concept, thus allowing the identification of multi-word concepts (*e.g.* "New York"). An example of this labeling scheme is shown in Figure 4.3.

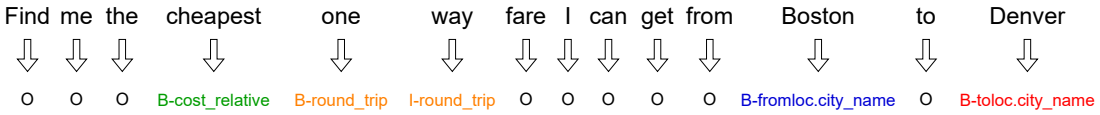


Figure 4.3: Slot-filling IOB (Ramshaw and Marcus, 1995) labels for an utterance of MultiATIS++ (Xu et al., 2020) in English. Label "O" (from *outside*) denotes that no concept is mentioned, "B" (from *beginning*) denotes the first word of a concept and "I" (from *inside*) the continuation of a concept. Different slot types are shown in different colors.

Given the possibility of class imbalance, sequence labeling tasks are usually evaluated using the slot micro F1 score (Tjong Kim Sang and Buchholz, 2000). Contrary to the macro averaging strategy that computes the individual F1 scores of each class and then averages them, the micro F1 score computes the average across all predictions independently of the class.

In the following sections 4.2.1 and 4.2.2 we present the corpora we use in our experiments: MultiATIS++ (Xu et al., 2020) for slot-filling in natural language understanding for task-oriented dialogue systems, and MultiCoNER (Malmasi et al., 2022a) for complex and ambiguous named entity recognition. In the rest of the chapter and for both corpora we denote the *train*, *dev* and *test* sets of a given language i with a subscript (e.g. $train_i$).

4.2.1 The MultiATIS++ corpus

The MultiATIS++ multilingual corpus derives from the Air Travel Information System (ATIS) corpus (Hemphill et al., 1990), consisting of user utterances asking for flight information. This corpus is built for the slot-filling task, which is related to task-oriented dialogue systems. It enables the system to identify the important concepts mentioned by the user that are needed to successfully continue the dialogue. These concepts are related to the system's domain (in this case flight information queries) and to the tasks that the system should perform (*e.g.* listing flights from Boston to Atlanta).

MultiATIS++ is the manual translation of the original English (EN) ATIS sentences into 6 different languages: Spanish (ES), Portuguese (PT), German (DE), French (FR), Chinese (ZH) and Japanese (JA). It also includes two additional languages: Hindi (HI) and Turkish (TR), that were added as part of MultiATIS

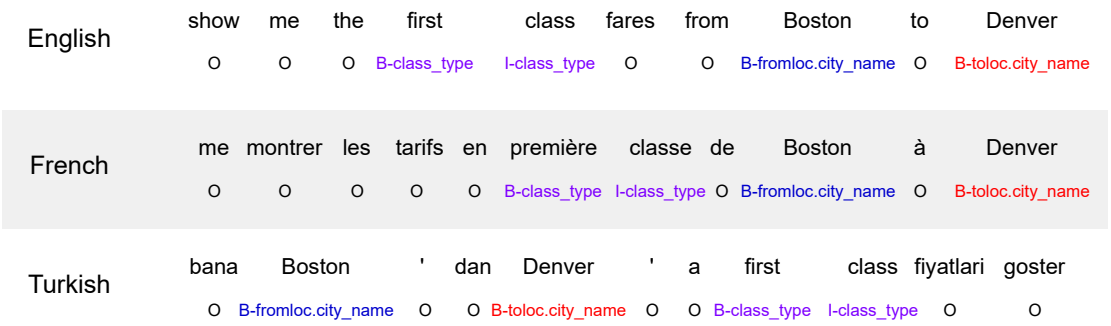


Figure 4.4: A labeled sentence extracted from MultiATIS++ (Xu et al., 2020) alongside its French and Turkish translations.

in (Upadhyay et al., 2018). An example of a labeled English sentence translated into different languages is shown in Figure 4.4.

Contrary to the translations added in MultiATIS++, the number of utterances of Hindi and Turkish translations are not as many as for the other languages. More details on the composition of this corpus are shown in Table 4.1.

Language	**Utterances**			**Labels**
	train	*dev*	*test*	
German (DE)	4,488	490	893	84
English (EN)	4,488	490	893	84
Spanish (ES)	4,488	490	893	84
French (FR)	4,488	490	893	84
Japanese (JA)	4,488	490	893	84
Portuguese (PT)	4,488	490	893	84
Chinese (ZH)	4,488	490	893	84
Hindi (HI)	1,440	160	893	75
Turkish (TR)	578	60	715	71

Table 4.1: Number of sentences per subset and number of unique labels (not counting B and I prefixes as different classes) in MultiATIS++ (Xu et al., 2020).

4.2.2 The MultiCoNER corpus

The MultiCoNER corpus was proposed as part of the SemEval 2022 Task 11 (Malmasi et al., 2022a,b) and focuses on the named entity recognition task. While it is usually a generic task consisting in identifying entities like people, organizations or dates in written text, MultiCoNER focuses on ambiguous and complex entities

English	it	was	republished	by	MIT	Press	in	1971	and	is	still	in	print
	O	O	O	O	B-CORP	I-CORP	O	O	O	O	O	O	O
Spanish	Europa	es	una	película	dirigida	por	Lars	Von	Trier				
	O	O	O	O	O	O	B-PER	I-PER	I-PER				
Chinese	裸	麥	麵	包	,	用	黑	麦	面	粉	制	成	
	B-PROD	I-PROD	I-PROD	I-PROD	O	O	O	O	O	O	O	O	

Figure 4.5: Labeled sentences extracted from MultiCoNER (Malmasi et al., 2022a) in English, Spanish and Chinese. "CORP" denotes an organization, "PER" denotes a person, and "PROD" denotes a product. Sentences with multiple types of entities are also found in the corpus.

in short and low-context settings. The entities defined in this corpus are "person", "location", "group", "organization", "product" and "creative work". Some examples of annotated sentences in the corpus are shown in Figure 4.5.

Language	**Utterances**		
	train	*dev*	*test*
Bengali (BN)	15,300	800	133,119
German (DE)	15,300	800	217,824
English (EN)	15,300	800	217,818
Spanish (ES)	15,300	800	217,887
Farsi (FA)	15,300	800	165,702
Hindi (HI)	15,300	800	141,565
Korean (KO)	15,300	800	178,249
Dutch (NL)	15,300	800	217,337
Russian (RU)	15,300	800	217,501
Turkish (TR)	15,300	800	136,935
Chinese (ZH)	15,300	800	151,661

Table 4.2: Number of sentences per subset in MultiCoNER (Malmasi et al., 2022a).

MultiCoNER also aims at stimulating research on multilingual models, as it contains annotations in 11 languages. Due to time and computing limitations, we restrict experiments on MultiCoNER to contain 9 languages as well, while targeting a maximum language overlap with MultiATIS++. As a result, the languages used in our experiments are Bengali (BN), German (DE), English (EN), Span-

ish (ES), Hindi (HI), Korean (KO), Dutch (NL), Turkish (TR) and Chinese (ZH). Further details about the composition of MultiCoNER are shown in Table 4.2.

4.3 Model architecture and training

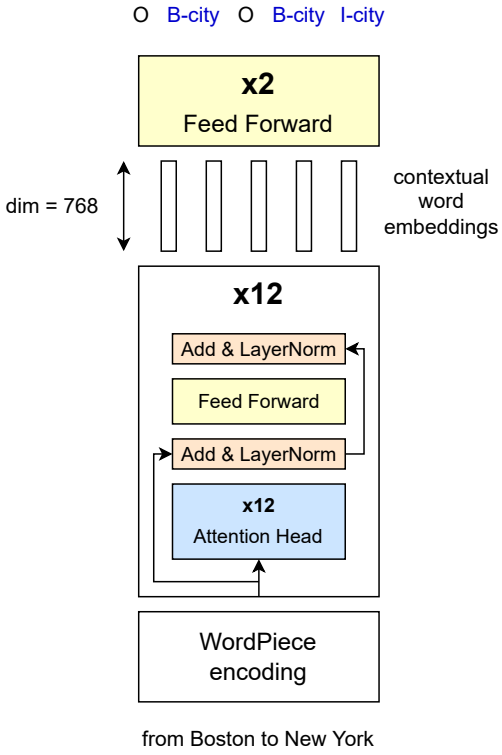


Figure 4.6: BERT (Devlin et al., 2019) for sequence labeling.

We use the multilingual BERT (Devlin et al., 2019) base model, consisting of 12 multi-head attention layers with 12 heads and hidden size of 768 (177M parameters). This model was trained on large Wikipedia dumps from 104 different languages using masked language modeling and next sentence prediction objectives.

In order to train the model for sequence labeling, we append a two-layer feed-forward classifier with hidden size 768 and ReLU (rectified linear unit) activation (Nair and Hinton, 2010). The input of the classifier are the last-layer hidden states of each word (*i.e.* the contextual word embeddings) after applying dropout with $p = 0.1$. A diagram of the full architecture is shown in Figure 4.6.

Following (Xu et al., 2020), we train on MultiATIS++ using the Adam opti-

mizer (Kingma and Ba, 2015) with a learning rate of 10^{-5} and a batch size of 32 utterances for 50 epochs (unless stated otherwise), selecting the checkpoint with the highest slot F1 on the corresponding *dev* set. We train the model on MultiCoNER the same way, except for the learning rate (optimized on *dev* and set to 5×10^{-5}) and the number of epochs, which is set to 15. We evaluate the model on all $test_i$ sets for every language i using the slot micro F1 score calculated with the `seqeval` library (Nakayama, 2018).

4.4 Measures of transfer

Cross-lingual transfer is defined as the performance improvement of a model on a particular language based on knowledge of other languages. This can take several forms depending on the training structure. In an *i.i.d.* context, where all data is available from the start, we think of transfer in terms of joint training. If training on language i and j jointly (multilingual) yields better performance on j than training only on j (monolingual), then there is transfer from i to j.

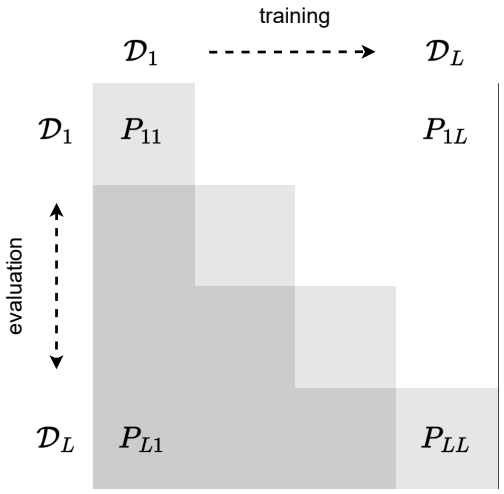


Figure 4.7: Performance matrix used to measure backward and forward transfer.

However, continual learning adds a different dimension. When training on a language sequence we can identify two types of transfer: forward and backward (Hadsell et al., 2020; Lopez-Paz and Ranzato, 2017). Forward transfer denotes the performance and learning efficiency improvement on a given language thanks to previously acquired knowledge of other languages. Conversely, backward transfer denotes the performance improvement on a previously acquired language when learning a new one. More formally, and similarly to Lopez-Paz and Ranzato (2017), given a sequence $\mathbf{S} = (\mathcal{D}_1, \ldots, \mathcal{D}_L)$ of L monolingual corpora $\mathcal{D}_i$ from L different languages, we define the performance matrix $P \in \mathbb{R}^{L \times L}$, where P_{ij} is the performance of language i after learning language j.

In this context, backward transfer of language i is defined as:

$$\mathrm{BT}_i = P_{iL} - P_{ii} \tag{4.1}$$

Note that negative backward transfer is equivalent to forgetting, as it denotes performance loss on previous languages. Since P_{11} is equivalent to monolingual performance mono_1 (obtained after training exclusively on $\mathcal{D}_1$), we facilitate our discussion by defining the backward transfer of the first language after learning language j:

$$\mathrm{BT}_{1j} = P_{1j} - \mathrm{mono}_1 \tag{4.2}$$

Conversely, we define forward transfer as:

$$\mathrm{FT}_i^{\mathrm{mono}} = P_{ii} - \mathrm{mono}_i \tag{4.3}$$

where mono_i denotes monolingual performance on language i (obtained after training exclusively on $\mathcal{D}_i$). By comparing performance to a different reference like multilingual, we can measure how close forward transfer is to the joint transfer topline:

$$\mathrm{FT}_i^{\mathrm{multi}} = P_{ii} - \mathrm{multi}_i \tag{4.4}$$

where multi_i denotes the multilingual performance on language i (obtained after training on the entire **S** jointly). These definitions will be particularly useful when examining the effect of the training sequence in Section 4.6.

4.5 Cross-lingual transfer

Before diving into the continual learning setting, we first measure transfer when training the model on all languages at once (*i.e. joint* transfer). Then, having this frame of reference, we investigate transfer when training the model on the **S** language sequence (*i.e. continual* transfer).

4.5.1 Joint transfer

In order to measure transfer in unstructured *i.i.d.* training, we train the model on all languages together (multilingual) and compare the performance we obtain with monolingual training. Note that multilingual training corresponds to concatenating all $train_i$ for training and all dev_i for validation. We report the mean

and standard deviation of *test* slot F1 per language across 5 runs to reduce the effect of randomness.

Training	DE	EN	ES	FR	PT	ZH	JA	HI	TR
Monolingual	94.4 (0.2)	95.6 (0.1)	88.9 (0.4)	93.2 (0.1)	90.3 (0.6)	93.3 (0.4)	93.1 (0.4)	82.4 (0.5)	71.3 (0.9)
Multilingual	95.0 (0.2)	96.0 (0.2)	90.4 (0.4)	94.0 (0.3)	91.4 (0.2)	93.6 (0.2)	93.0 (0.1)	87.2 (0.3)	85.2 (0.6)
Joint transfer	+0.6	+0.4	+1.5	+0.8	+1.1	+0.3	-0.1	+4.8	+13.9

Table 4.3: Slot F1 performance on MultiATIS++ on $test_i$ sets for monolingual and multilingual experiments. Reported values are the average of 5 runs with standard deviation shown in parenthesis. Joint transfer denotes the difference between multilingual and monolingual performance.

Results on MultiATIS++ are reported in Table 4.3. We observe that multilingual is always stronger than monolingual, which confirms the existence of joint cross-lingual transfer. The only exceptions to this are Chinese and Japanese, whose joint transfer values (0.3% and -0.1% respectively) are not significant when compared to the standard deviation. European languages (German, English, Spanish, French and Portuguese) show modest but visible gains from transfer, whereas Asian languages (Chinese and Japanese) do not seem to benefit from it. However, transfer for the two low-resource languages (Hindi and Turkish) is outstanding, with an absolute 4.8% and 13.9% improvement.

As noted by Do et al. (2020), MultiATIS++ translations keep the same (unrealistic) slot values for particular labels (e.g. American *departure city* and *destination city* in Turkish utterances). We suspect this may be the reason why transfer is particularly high in this corpus. The fact that the corpus contains less training data for Hindi and Turkish than for the other languages might also explain why joint transfer is much higher for these two languages. This is an interesting result for continual adaptation in production, as it shows that languages with little data have a high potential for performance gains from transfer.

Training	BN	DE	EN	ES	HI	KO	NL	TR	ZH
Monolingual	41.6 (3.2)	64.1 (0.8)	61.3 (0.6)	59.0 (0.8)	43.1 (1.2)	56.7 (0.7)	61.4 (0.9)	45.7 (0.7)	57.6 (0.8)
Multilingual	44.9 (1.6)	66.9 (0.4)	64.4 (0.7)	63.8 (0.4)	46.4 (1.2)	59.4 (0.8)	66.5 (0.5)	50.6 (1.0)	58.2 (1.0)
Joint transfer	+3.3	+2.8	+3.1	+4.8	+3.3	+2.7	+5.1	+4.9	+0.6

Table 4.4: Slot F1 performance on MultiCoNER on $test_i$ sets for monolingual and multilingual experiments. Reported values are the average of 5 runs with standard deviation shown in parenthesis. Joint transfer denotes the difference between multilingual and monolingual performance.

Table 4.4 shows our experimental results on MultiCoNER. Monolingual performance is much lower than in MultiATIS++ even if the set of labels to predict is

also smaller, which suggests that MultiCoNER may indeed be more difficult than MultiATIS++. Although the corpus is not parallel (*i.e.* a sentence in a language might not be found in another), we observe significant joint cross-lingual transfer (except for Chinese whose 0.6% improvement is negligible). This is interesting considering that only a maximum of 8% of entity mentions appearing in the test set of a given language are common to those appearing in the train set of other languages.

Corpus	**Training**	**Model Cost**		**Data Cost**
		Time	Space	Space
MultiATIS++	Monolingual	≤224K	1.6B	≤4K
	Multilingual	1.7M	178M	33K
	Continual	≤224K	178M	≤4K
MultiCoNER	Monolingual	765K	1.6B	15K
	Multilingual	6.9M	178M	138K
	Continual	765K	178M	15K

Table 4.5: Costs of learning a new language according to training scheme and corpus. Model time cost is measured in iterations, while model space cost is the model size measured in number of parameters. Data space cost is the number of training sentences stored in memory at the same time.

However, multilingual training assumes that all languages are available at once. As mentioned before, this is not always true in practice, since utterances may be scarce and annotations expensive. Moreover, given N the maximum number of utterances per language and L the number of languages, training on a new language has time cost $O(LN)$, as the whole model needs to be trained from scratch. A naive solution is to use multiple monolingual models, raising however the space cost to $O(LN)$. Reducing both costs to $O(N)$ motivates our decision to structure training as a sequence. To put this costs in the context of our experiments, Table 4.5 shows the time and space costs of adding an L^{th} language with each training scheme and

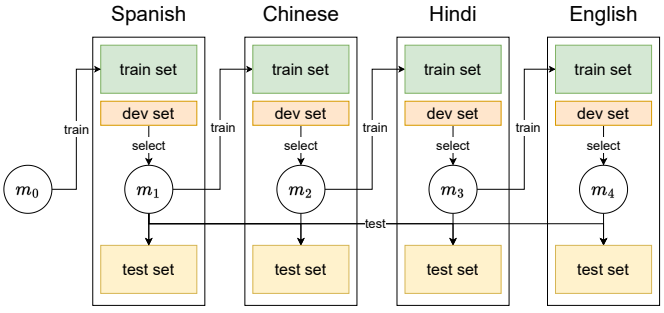


Figure 4.8: A training sequence of 4 languages.

corpus. For example, notice that a single model (composed of 178M parameters) is stored in the continual and joint multilingual training schemes, while $L = 9$ models (1.6B parameters) need to be stored in the monolingual case (one for each language).

4.5.2 Continual transfer

Given a training sequence (a list of languages in a given order), our continual learning experiments consist in training the model on $train_i$ (and validating on dev_i) for each language i in the given order, as depicted in Figure 4.8. Although having all language data at once is not required and the language addition cost is the lowest, this approach is prone to forgetting previously learned languages.

Sequence sampling

In the experiments of this section, we report for both forward and backward transfer the average performance per language. The experiments consist of 3 sequences per language and per transfer type repeated 5 times to reduce the effect of randomness, making a total of 54 sequences and 270 experiments. These 3 sequences per language are selected through a semi-random iterative procedure described in Figure 4.9, which relies on the Kendall rank correlation coefficient τ (Abdi, 2007) to ensure a high level of variability.

After randomly sampling a sequence to constitute the initial base set, each iteration consists of three steps: 1) we sample N random sequences with the same first language (for backward transfer experiments) or last language (for forward transfer experiments), 2) we compute the average Kendall rank correlation coefficient τ between each sequence and the base set, and 3) we rank the sequences according to τ and append the one with the lowest τ to the base set. This process is repeated until the base set contains the desired amount of sequences (3 sequences in our case).

Forward and backward cross-lingual transfer

We first investigate whether forward transfer exists in continual training by looking at the average P_{LL} performance (*e.g.* m_4 evaluated on English in Figure 4.10) against monolingual and multilingual. Notice that we look at the performance of the last language, as this allows us to measure whether the model leverages past knowledge to learn a new language. This also has the advantage of isolating the effect of forward transfer from that of backward transfer. For a fair comparison, when sampling sequences we also make sure that each language appears at the *end* of the sequence the same number of times.

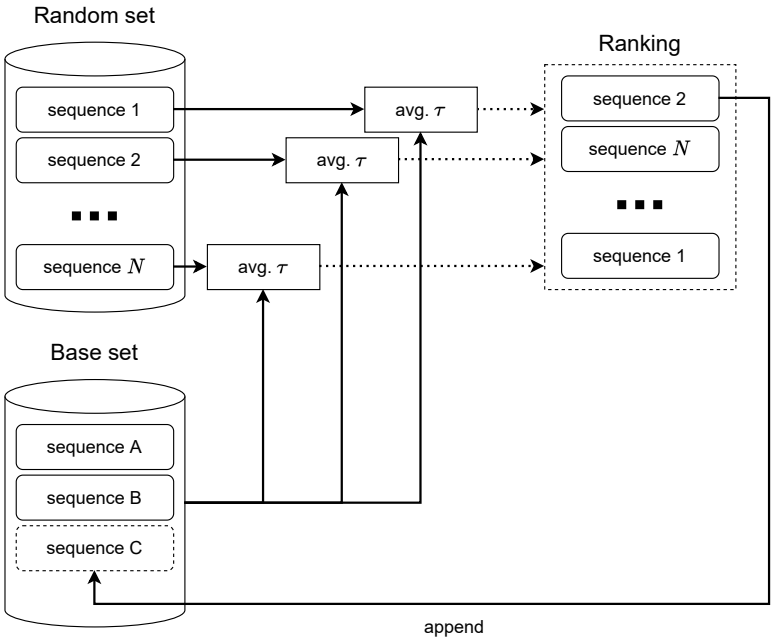


Figure 4.9: Sequence sampling algorithm based on the Kendall rank correlation coefficient τ computed between sequences of language indices. After choosing a randomly sampled sequence for the initial base set, we sample N random sequences with the same first/last language (depending on the target transfer metric), which are then ranked according to their average τ with respect to the base set. The sequence with the lowest correlation is then appended to the base set. This process is repeated until the base set is filled with 3 sequences.

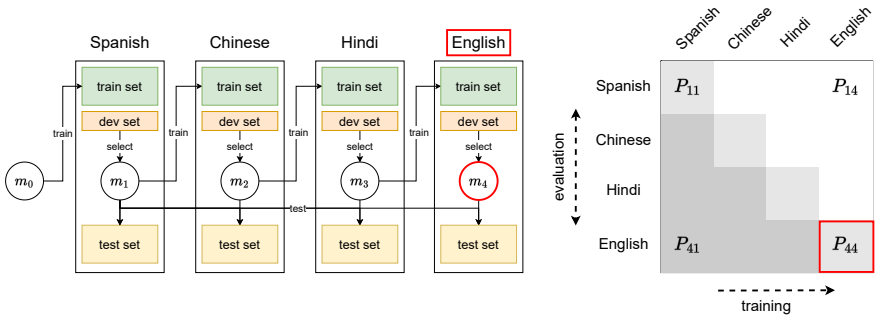


Figure 4.10: To measure forward transfer, we look at the performance P_{LL}. In this example, this is equivalent to obtaining P_{44} by evaluating model m_4 on English. Involved elements are marked in red.

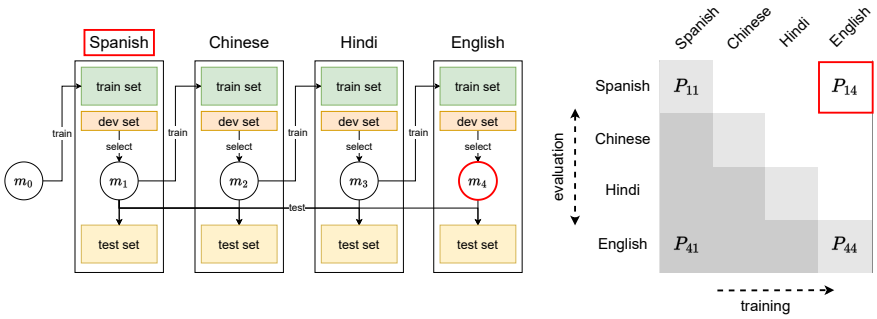


Figure 4.11: To measure backward transfer, we look at the performance P_{1L}. In this example, this is equivalent to obtaining P_{14} by evaluating model m_4 on Spanish. Involved elements are marked in red.

Similarly, we look at backward transfer by comparing the average P_{1L} performance (*e.g.* m_4 evaluated on Spanish in Figure 4.11) against monolingual, making sure that each language appears at the *beginning* of the sequence the same number of times. This way we can determine whether the initial performance P_{11} (equal to monolingual) improves with the introduction of new languages to the model. We also look at the performance of the first language in order to isolate the effect of backward transfer from that of forward transfer.

Notice that whether we focus on the first or the last language, we always look at the performance at the end of the training sequence so that the comparison to multilingual is fair.

Results

Our results on MultiATIS++ are reported in Table 4.6. We observe that continual training benefits from cross-lingual forward transfer, as FT_{1L}^{mono} is always positive and higher than the standard deviation (except for Japanese). In fact, P_{LL} is generally closer to multilingual than to monolingual performance. However, although transfer is present in language L, the performance of the first language P_{1L} suffers from the opposite effect, even falling under monolingual performance. Our results show that, contrary to what we expected from the identical slot values of MultiATIS++ (e.g. American *departure city* and *destination city* in Turkish utterances), the naturally occurring cross-lingual transfer completely vanishes in previous languages.

Similar observations can be made from continual experiments on MultiCoNER, whose results are presented in Table 4.7. Although forward transfer is high in general, it is also lower than the standard deviation for Bengali, Hindi and Ko-

Training	DE	EN	ES	FR	PT	ZH	JA	HI	TR
Monolingual	94.4 (0.2)	95.6 (0.1)	88.9 (0.4)	93.2 (0.1)	90.3 (0.6)	93.3 (0.4)	93.1 (0.4)	82.4 (0.5)	71.3 (0.9)
Multilingual	95.0 (0.2)	96.0 (0.2)	90.4 (0.4)	94.0 (0.3)	91.4 (0.2)	93.6 (0.2)	93.0 (0.1)	87.2 (0.3)	85.2 (0.6)
Continual (P_{LL})	94.9 (0.2)	95.9 (0.1)	89.9 (0.5)	93.9 (0.3)	91.3 (0.3)	93.9 (0.3)	93.1 (0.3)	85.6 (0.7)	84.0 (0.6)
FT^{mono}_{1L}	+0.5	+0.3	+1.0	+0.7	+1.0	+0.6	+0.0	+3.2	+12.7
Continual (P_{1L})	94.0 (0.7)	95.5 (0.2)	89.2 (0.5)	91.4 (1.7)	88.4 (4.9)	92.0 (1.0)	91.7 (0.7)	80.5 (1.8)	68.1 (3.5)
BT_{1L}	-0.4	-0.1	+0.3	-1.8	-1.9	-1.3	-1.4	-1.9	-3.2

Table 4.6: Slot F1 performance on MultiATIS++ on $test_i$ sets for monolingual, multilingual and continual experiments. The latter are calculated as the average of the first (P_{1L}) or last (P_{LL}) language (indicated by the column) at the end of the sequence. See Equations 4.2 and 4.3 for the definition of BT_{1L} and FT^{mono}_{1L}. Reported values are the average of 5 runs with standard deviation shown in parenthesis.

Training	BN	DE	EN	ES	HI	KO	NL	TR	ZH
Monolingual	41.6 (3.2)	64.1 (0.8)	61.3 (0.6)	59.0 (0.8)	43.1 (1.2)	56.7 (0.7)	61.4 (0.9)	45.7 (0.7)	57.6 (0.8)
Multilingual	44.9 (1.6)	66.9 (0.4)	64.4 (0.7)	63.8 (0.4)	46.4 (1.2)	59.4 (0.8)	66.5 (0.5)	50.6 (1.0)	58.2 (1.0)
Continual (P_{LL})	43.4 (1.8)	66.0 (0.6)	63.0 (0.6)	62.1 (0.9)	44.2 (1.0)	57.0 (0.7)	64.6 (0.6)	50.1 (0.8)	56.2 (1.3)
FT^{mono}_{1L}	+1.8	+1.9	+1.7	+3.1	+1.1	+0.3	+3.2	+4.4	-1.4
Continual (P_{1L})	31.7 (4.5)	50.9 (1.5)	52.5 (2.6)	51.1 (2.3)	32.2 (2.4)	43.2 (2.4)	55.4 (3.4)	37.4 (1.9)	40.0 (2.8)
BT_{1L}	-9.9	-13.2	-8.8	-7.9	-10.9	-13.6	-6.0	-8.3	-17.6

Table 4.7: Slot F1 performance on MultiCoNER on $test_i$ sets for monolingual, multilingual and continual experiments. The latter are calculated as the average of the first (P_{1L}) or last (P_{LL}) language (indicated by the column) at the end of the sequence. See Equations 4.2 and 4.3 for the definition of BT_{1L} and FT^{mono}_{1L}. Reported values are the average of 5 runs with standard deviation shown in parenthesis.

rean, and even negative for Chinese. As in MultiATIS++, the negative backward transfer values also confirm the presence of forgetting in the first language learned.

Overall, we can see that continual training benefits from forward transfer, although still not performing as well as the multilingual topline. At the same time, forgetting is clearly present in the first language.

4.6 Effect of the training sequence

In order to better understand the effect of the training sequence on cross-lingual transfer, we first look at measures of forward transfer at each position relative to monolingual and multilingual. Secondly, we study the impact of the training sequence length on backward transfer measured on the first language. This analysis is conducted only on MultiATIS++ due to time and computational constraints.

The results of this section are presented in the form of box plots. As shown in Figure 4.12, the boundaries of each box constitute the 25^{th} and 75^{th} percentiles, denoted as Q1 and Q3 respectively. We also show the median as an orange line and the mean as a green dot. The boundaries of the whiskers are computed in a standard manner: $\text{Q1} - 1.5\,\text{IQR}$ and $\text{Q3} + 1.5\,\text{IQR}$, where $\text{IQR} = \text{Q3} - \text{Q1}$ is the interquartile range. Note that the whiskers can be asymmetrical because the boundaries need to be represented by a data point. In our figures, we choose the closest data point that is within the ideal boundaries described above (which is common practice). Finally, any values outside of the whiskers are considered outliers.

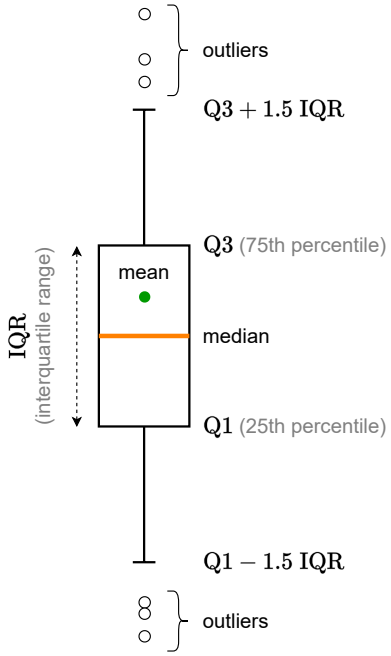
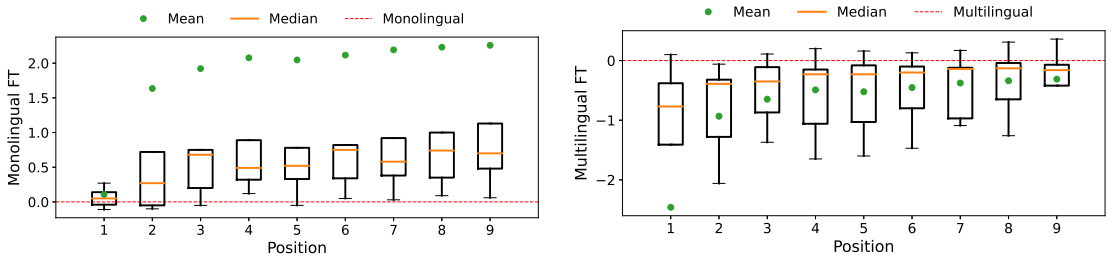


Figure 4.12: Anatomy of a box plot as presented in this section.

Effect of language position on forward transfer

When considering forward transfer, Figure 4.13a shows that apart from the first position (equal to monolingual), the model consistently benefits from transfer at any point in the sequence, as performance is higher than monolingual. Interestingly, due to some outlier languages (generally Hindi and Turkish), we observe that the means are poor estimates of the distribution when measuring $\text{FT}_i^{\text{mono}}$. This is an indicator that commonly used continual transfer metrics might over- or underestimate real performance when transfer is not uniformly distributed among languages. Indeed, these metrics usually consist of averages across the adaptation



(a) $\text{FT}_i^{\text{mono}} = P_{ii} - \text{mono}_i$ (higher is better) (b) $\text{FT}_i^{\text{multi}} = P_{ii} - \text{multi}_i$ (higher is better)

Figure 4.13: Distributions of forward transfer on $test_i$ relative to (a) monolingual and (b) multilingual for different positions i in the sequence. We average over 54 sequences and 5 runs. Note that forward transfer is 0 when performance is equal to (a) monolingual and (b) multilingual. Outliers not shown for readability.

axis (Lopez-Paz and Ranzato, 2017). In Figure 4.13b, we also observe that performance gets closer to multilingual as the sequence advances, although it rarely outperforms it.

Effect of sequence length on backward transfer

As per backward transfer, Figure 4.14 shows that performance of the first language is in general worse than monolingual for any given sequence length. In particular, we observe that performance loss is not strictly monotonic, which means that measuring forgetting between the beginning and the end of the sequence may not be sufficient to explain how the model forgets. Note that a sequence of $L = 7$ would have shown less forgetting than a sequence of $L = 5$ in this particular example.

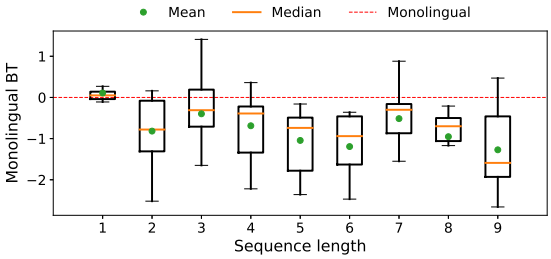


Figure 4.14: Distributions of first-language backward transfer $\text{BT}_{1j} = P_{1j} - \text{mono}_1$ (higher is better) on $test_1$ for different sequence lengths j. We average across 54 sequences and 5 runs. Note that $\text{BT}_{1j} = 0$ if performance is equal to monolingual. Outliers not shown for readability.

Furthermore, as hinted by the results from P_{1L} and P_{LL}, we observe that backward

transfer deteriorates as forward transfer improves with the length of the sequence. Since negative backward transfer (*i.e.* forgetting) tends to be linked to a loss of previously acquired knowledge, it is surprising that new language performance keeps increasing while the performance of known languages decreases.

The "progressively-improving initialization" hypothesis

Our results seem to indicate that the preserved knowledge that facilitates the acquisition of a new language in multilingual BERT for slot-filling is not the same knowledge that preserves previous language performance. This might be explained by a progressive shift of model parameters towards a better multilingual initialization for the ATIS task that might however fail to constitute a sufficiently good solution for previous languages.

This hypothesis is depicted with an example in two dimensions in Figure 4.15, and it motivates the experiments of the next section.

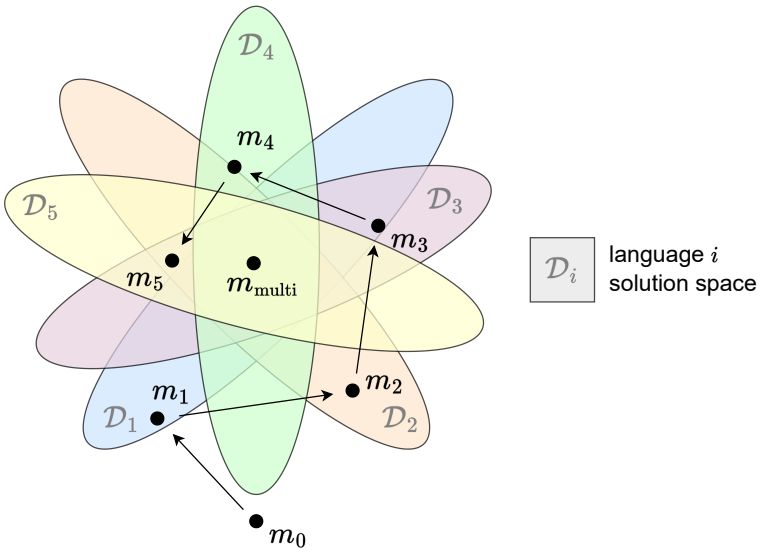


Figure 4.15: Hypothesis on the progressive parameter shift towards a better multilingual initialization depicted in two dimensions. Model parameters m_i are obtained after training on data $\mathcal{D}_i$ for language i, while m_{multi} is obtained after joint multilingual training. Colored regions represent solution spaces for each $\mathcal{D}_i$, where parameters closer to the center of region i obtain better performance on language i. As the training sequence progresses, parameters move to the new solution space, reducing their distance to m_{multi} after each training stage. However, models m_i never reach the optimal multilingual solution m_{multi}.

4.7 Performance recovery

Given that forward transfer does not seem to be affected by forgetting, in this section we investigate whether performance lost as a result of forgetting can be recovered quickly after continual training. In other words, and using Figure 4.15 as a visual example, we attempt to determine how quickly model m_L (the model at the end of training sequence) can reach monolingual optimum m_1, and even the multilingual optimum m_{multi}.

The ability to recover is especially interesting for MultiCoNER, where forgetting is considerably higher, but we still conduct experiments on both corpora. To investigate if this is possible, we set out to confirm whether the model is in fact shifting towards a better multilingual initialization. To do this, we compare the multilingual performance of the initial m_0 (consisting of pre-trained BERT and a randomly initialized classifier) against m_L. In particular, we train both models on all languages jointly (multilingual) and on each language individually (monolingual) for different numbers of epochs and evaluate on each language. Notice that m_L is taken from our continual P_{1L} experiments. The results are presented in tables 4.8 and 4.9.

Training	Model	Epochs	DE	EN	ES	FR	PT	ZH	JA	HI	TR
Multilingual	m_0 (rnd clf)	1	82.7 (1.2)	83.6 (0.7)	78.2 (0.3)	80.7 (0.7)	79.4 (0.5)	83.5 (0.7)	82.7 (1.0)	79.6 (0.7)	69.8 (1.5)
		5	94.7 (0.2)	95.3 (0.2)	89.9 (0.2)	93.2 (0.2)	90.7 (0.2)	94.0 (0.2)	93.2 (0.5)	85.9 (0.3)	83.6 (0.7)
		50	95.0 (0.2)	96.0 (0.2)	90.4 (0.4)	94.0 (0.3)	91.4 (0.2)	93.6 (0.2)	93.0 (0.1)	87.2 (0.3)	85.2 (0.6)
	m_L	1	94.8 (0.3)	95.9 (0.2)	89.7 (0.6)	93.8 (0.3)	91.2 (0.4)	93.6 (0.5)	93.3 (0.3)	85.7 (0.9)	82.8 (1.3)
		5	94.9 (0.2)	95.9 (0.2)	90.0 (0.5)	93.9 (0.3)	91.3 (0.4)	93.7 (0.4)	93.3 (0.3)	86.0 (0.8)	83.4 (1.0)
	m_L (rnd clf)	1	93.1 (0.5)	93.7 (0.5)	87.9 (0.5)	91.1 (0.5)	88.5 (0.6)	92.6 (0.5)	92.3 (0.6)	83.4 (0.8)	80.8 (1.3)
		5	94.8 (0.2)	95.8 (0.2)	89.9 (0.5)	93.6 (0.3)	91.1 (0.4)	93.7 (0.4)	93.3 (0.3)	86.3 (0.6)	84.1 (0.8)
Monolingual	m_0 (rnd clf)	50	94.4 (0.2)	95.6 (0.1)	88.9 (0.4)	93.2 (0.1)	90.3 (0.6)	93.3 (0.4)	93.1 (0.4)	82.4 (0.5)	71.3 (0.9)
	m_L	1	95.1 (0.2)	95.8 (0.2)	90.2 (0.4)	93.6 (0.4)	91.2 (0.4)	93.5 (0.5)	93.4 (0.2)	86.3 (0.6)	79.1 (1.5)
		5	95.0 (0.2)	95.8 (0.2)	90.0 (0.4)	94.0 (0.2)	91.3 (0.2)	93.8 (0.4)	93.4 (0.2)	86.7 (0.4)	81.6 (0.8)
		10	95.1 (0.2)	95.8 (0.2)	90.0 (0.5)	93.9 (0.3)	91.3 (0.4)	93.8 (0.4)	93.4 (0.2)	86.7 (0.4)	82.2 (0.9)

Table 4.8: Slot F1 performance on $test_i$ sets for MultiATIS++ fast recovery experiments. Model m_L monolingual performance is averaged over 3 sequences (the P_{1L} experiment ones starting with the language in question), while m_L multilingual is averaged over all 27 sequences from P_{1L} experiments. Both m_0 and m_L experiments are averaged over 5 runs (standard deviation in parenthesis).

On the multilingual qualities of m_L

The comparison between m_0 and m_L shows two interesting results. On the one hand, we observe that even one epoch of multilingual training for m_L achieves better performance than the monolingual baseline (m_0 monolingual trained on the maximum number of epochs) and is even close to the multilingual topline (m_0

Training	Model	Epochs	BN	DE	EN	ES	HI	KO	NL	TR	ZH
Multilingual	m_0 (rnd clf)	1	36.2 (1.4)	63.1 (0.8)	61.6 (0.6)	60.5 (0.6)	40.5 (1.4)	56.9 (0.4)	63.5 (0.7)	45.5 (0.6)	53.1 (2.4)
		5	43.0 (1.1)	66.6 (1.0)	63.9 (0.2)	63.7 (0.6)	45.4 (1.5)	58.9 (0.7)	66.3 (0.7)	49.7 (1.4)	57.7 (1.5)
		15	44.9 (1.6)	66.9 (0.4)	64.4 (0.7)	63.8 (0.4)	46.4 (1.2)	59.4 (0.8)	66.5 (0.5)	50.6 (1.0)	58.2 (1.0)
	m_L	1	42.7 (1.7)	65.8 (0.7)	63.6 (0.7)	63.0 (0.8)	44.8 (1.4)	58.8 (1.0)	65.9 (0.8)	49.8 (1.0)	56.7 (1.3)
		5	43.8 (1.4)	66.4 (0.6)	64.1 (0.5)	63.5 (0.6)	45.4 (1.1)	59.2 (0.8)	66.4 (0.5)	50.6 (0.9)	57.6 (1.2)
	m_L (rnd clf)	1	42.6 (1.8)	65.5 (0.7)	63.3 (0.6)	62.7 (0.8)	44.7 (1.3)	58.7 (0.8)	65.7 (0.7)	49.6 (1.2)	56.6 (1.4)
		5	43.7 (1.4)	66.3 (0.6)	63.9 (0.6)	63.4 (0.7)	45.2 (1.1)	59.1 (0.8)	66.2 (0.6)	50.4 (1.0)	57.6 (1.1)
Monolingual	m_0 (rnd clf)	15	41.6 (3.2)	64.1 (0.8)	61.3 (0.6)	59.0 (0.8)	43.1 (1.2)	56.7 (0.7)	61.4 (0.9)	45.7 (0.7)	57.6 (0.8)
	m_L	1	41.8 (2.4)	65.5 (0.7)	63.7 (0.8)	61.6 (0.5)	44.2 (1.1)	57.6 (0.4)	64.6 (0.7)	49.5 (1.0)	56.0 (0.9)
		5	43.6 (1.8)	66.5 (0.5)	64.0 (0.6)	62.4 (0.6)	45.4 (0.7)	57.9 (0.5)	65.0 (0.8)	50.7 (0.7)	58.3 (0.9)

Table 4.9: Slot F1 performance on $test_i$ sets for MultiCoNER fast recovery experiments. Model m_L monolingual performance is averaged over 3 sequences (the P_{1L} experiment ones starting with the language in question), while m_L multilingual is averaged over all 27 sequences from P_{1L} experiments. Both m_0 and m_L experiments are averaged over 5 runs (standard deviation in parenthesis).

multilingual trained on the maximum number of epochs)[3]. This means that m_L is capable of achieving good multilingual performance with very little training, hence counteracting the effect of forgetting. On the other hand, we also observe that m_L multilingual performance is greatly superior to m_0 multilingual after a single training epoch. This is partly expected, as the classifier is initialized randomly in m_0, but it shows that the model is capable of retaining knowledge from previous languages, although it is not clear whether that knowledge is preserved in the classifier or in BERT.

On the localization of retained knowledge

We dive deeper into this question by training m_L with a randomly initialized classifier (see "m_L (rnd clf)" in tables 4.8 and 4.9) so that no classifier has any previous knowledge. Surprisingly, we observe that performance is still greatly superior to m_0 multilingual after a single epoch. However, performance is not as high as m_L multilingual (although slightly in MultiCoNER), which keeps its continually trained classifier. This indicates that most of the knowledge retained from previous languages is localized in BERT, and that the knowledge stored in the classifier may be corpus-dependent instead. Because of this, our results show that the widely adopted strategy of freezing parameters to protect previous knowledge (see constraint-based continual learning methods in Section 2.4.3) may in fact be counterproductive, as it is likely to hinder forward transfer, and hence recovery capabilities.

[3]Except for Chinese on MultiCoNER, which is not surprising, as joint transfer is negligible.

From a progressively-improving initialization to fast recovery

Overall, these results lead us to think that for the sequence labeling task, continual training over a language sequence does indeed shift model parameters to a better multilingual initialization. As a result, we explore the possibility to leverage this phenomenon in order to quickly recover lost language specificities (due to forgetting) for both corpora. To do this, we train m_L on the first language of the sequence a second time (*i.e.* as if it were an $(L+1)^{\text{th}}$ language) and evaluate on the first language only (see "m_L monolingual" rows in tables 4.8 and 4.9).

When comparing m_L monolingual to m_0 monolingual (equal to first language performance P_{11}), we see that the performance of the first language can be recovered and even improved with as little as a single training epoch[4]. These results are outstanding for MultiCoNER considering the high forgetting that we previously observed. On MultiATIS++, m_L monolingual even achieves 50-epoch m_0 multilingual performance in most cases after only one epoch, with the remaining languages still showing a large improvement. In particular, Hindi and Turkish improve an absolute 3.9% and 7.8% from m_0 monolingual respectively.

Note that for MultiATIS++ increasing the number of recovery epochs for the first language does not bring considerable improvements. The only exception to this observation is Turkish, which might be explained by the small size of its training set. However, performance still improves after 5 epochs in MultiCoNER, getting closer to the multilingual topline. Surprisingly, m_L monolingual is even on par with the multilingual topline for Turkish and Chinese.

Although the cost of adding a language remains $O(N)$, the ability to recover all languages raises costs to $O(LN)$, making it expensive to use in practice. The design of a strategy taking full advantage of these recovery capabilities to limit forgetting with lower costs is left for future work.

4.8 Conclusion

In this chapter, we have presented an analysis of cross-lingual transfer in continual learning for sequence labeling using multilingual BERT (Devlin et al., 2019) as well as the MultiATIS++ (Xu et al., 2020) and MultiCoNER (Malmasi et al., 2022a) corpora for slot-filling and named entity recognition respectively.

In practical low resource scenarios where data and annotations are scarce (*e.g.* a dialogue system progressively deployed in multiple countries), it may be difficult or even impossible to implement either a monolingual or multilingual fine-tuning

[4]Except for Chinese on MultiCoNER, which is not surprising, as joint transfer is negligible.

approach. This is mainly due to the high time and space complexity, and to the fact that not all language data might be available at once. In a continual learning setting where languages are learned in sequence, these costs are the lowest, but the phenomenon of forgetting appears, decreasing previous-language performance.

After extensive experiments conducted on a large array of sequences with languages arranged in different orders, we reach the following conclusions.

Relationship between transfer and forgetting. The joint cross-lingual transfer typically observed in pre-trained Transformer-based models like multilingual BERT is almost entirely retained in the form of forward transfer, but the model still fails to retain its performance on previously learned languages. This high forward transfer may be linked to a progressive shift of model parameters towards a better multilingual initialization, that still fails to reach optimal multilingual performance on all languages at the same time.

Model parameters and acquired knowledge. Most knowledge from past languages seems to be stored in the contextual word embedding model BERT and not in the task-specific classifier that we appended to it. This suggests that morphing the representation space may allow for more effective forward transfer than learning to exploit fixed pre-trained general-purpose embeddings. Moreover, it shows that relying on partial or total parameter freezing (like constraint-based and some modular approaches to continual learning) may in fact be counterproductive and hinder forward transfer.

Performance recovery. The high occurring forward transfer allows the model to quickly recover lost performance (due to forgetting) in an astoundingly short number of epochs, and even achieve joint multilingual performance. We believe future approaches to address catastrophic forgetting could take advantage of this phenomenon. However, it remains unclear if it is possible to reduce the associated time and space costs to turn this into a viable alternative.

Finally, we also find that current continual learning metrics may need to be adapted in order to better estimate the distribution of transfer across the adaptation axis.

Chapter 5

Autonomous streaming speaker diarization

> *"Quintessence of learning: being able to adapt to unpredictable conditions as quickly as possible."*
>
> — Stanislas Dehaene, *How We Learn*

Despite the valuable insights from the previous chapter, one of the key missing pieces in our study was the ability to learn from unlabeled data, which is far more commonly found in the production phase. In this chapter, we take a step in this direction by leveraging the metric learning representations obtained in Chapter 3 to adapt a modular streaming speaker diarization system to a live conversation in real time.

The chapter is structured as follows. In Section 5.1, we introduce the task of speaker diarization and our motivations. In Section 5.2, we discuss previous work, and in particular the recent end-to-end speaker diarization method that constitutes one of the main pillars of our approach. Next, in Section 5.3, we introduce the modular speaker diarization system that we propose and describe its building blocks. In Section 5.4, we present the incremental clustering algorithm we propose to continually refine speaker representations as the target conversation progresses. We also present our experimental protocol and discuss the results we obtain. Finally, in Section 5.5, we conclude the chapter with our final remarks.

This chapter has been the subject of the following scientific publication:

Juan M. Coria, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. Overlap-Aware Low-Latency Online Speaker Diarization Based on End-to-End Local Seg-

mentation. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Online, 2021.

5.1 Introduction

It is hard to deny that speech recognition applications have gained substantial traction in recent years. Machine learning systems capable of transcribing speech with astonishing performance are now widely available as full-fledged services and products. Some of these are even capable of real-time processing, effectively producing high-quality transcriptions as people speak.

Speaker diarization, the task of determining "who speaks when" in an audio recording, has traditionally been considered a pre-processing step to improve speech recognition systems (Park et al., 2022). However, it is also crucial for generating metadata and summaries, searching content, and organizing transcriptions in an easily readable format. In particular, *online* speaker diarization, where a system must recognize speakers in a streaming fashion as the conversation takes place, is extremely challenging and represents one of the major unsolved problems in modern diarization.

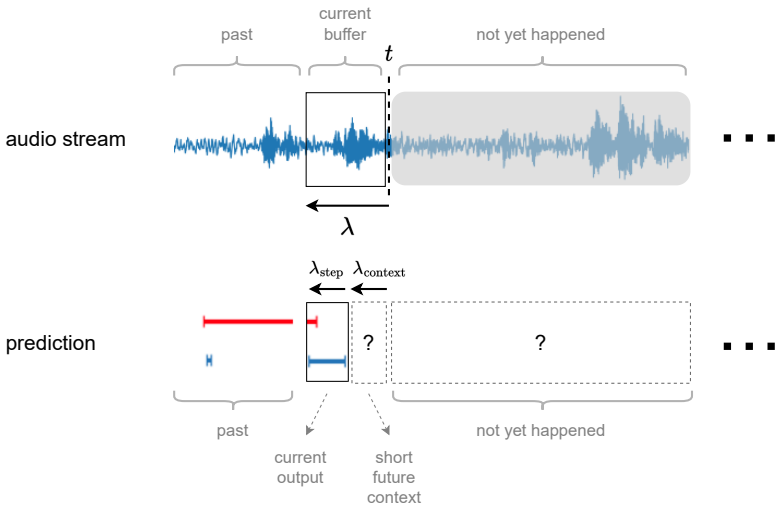


Figure 5.1: At any point in time t, an online speaker diarization system only has access to audio in the past. The latency λ, defined as the delay relative to t for which the system outputs a prediction, can be chosen to be the duration λ_{step} of the step (*i.e.* the "refresh rate" of the input buffer) plus the duration λ_{context} of a short extract of local future context (to provide useful additional information). Predictions in different colors denote different speakers.

As depicted in Figure 5.1, the main difficulty lies in the fact that future audio is inaccessible at a given point in time t of the conversation. This means that contrary to *offline* diarization, where the entire conversation is available from start to finish, known speakers may need to be re-identified and new speakers may need to be detected.

In theory, a true online speaker diarization system should process a single audio sample at a time. However, due to the limits of currently available computational resources, this input is typically defined as a short audio buffer updated every λ_{step} seconds. This introduces the concept of the algorithmic latency λ, defined as the delay relative to t for which the system outputs a prediction. In order to have the maximum responsiveness, one could set $\lambda = \lambda_{\text{step}}$, but in some cases it is also possible to allow a certain delay λ_{context} to provide a peek of the immediate future audio (with respect to the current prediction). This *local future context* may provide additional useful information for the system to exploit at the expense of responsiveness. In this sense, the algorithmic latency is defined as $\lambda = \lambda_{\text{step}} + \lambda_{\text{context}}$. It is worth noting that the *real time* latency is an additional variable at play that contributes to the responsiveness perceived by the user. This latency is defined as the time for the entire system to compute a prediction for a single state of the buffer, and it is highly dependant on both the implementation and the hardware. As a consequence, it is kept out of our study.

In the context of our work, online speaker diarization is an excellent candidate for continual adaptation in the production phase, as conversations might differ in language, microphone quality, or acoustic conditions, among others. In fact, a diarization system could learn from unlabeled audio (collected as time passes) in a target conversation in production to adapt to its specific conditions and maximize performance in full autonomy.

In order to recognize the various speakers in a conversation, diarization systems have typically relied on speaker embeddings (Madikeri et al., 2015; Snyder et al., 2018) to cast the comparison of speaker identities as a simpler scoring function. This allows to extract multiple embeddings from a recording and discover speakers through clustering (Diez et al., 2018; Lin et al., 2019). In Chapter 3, we have seen that metric learning techniques can provide a reliable way to compare speakers using a simple function like the cosine distance. We found this technique to be particularly useful in open-set problems, where the model is confronted with new classes in the production phase, instead of new instances of the same classes. Since speaker diarization does not restrict speakers to a fixed set, it can be considered an open-set problem as well.

For all these reasons, in this chapter we choose to study the problem of online

speaker diarization. More specifically, we propose a modular system with a low and adjustable latency that leverages incremental clustering of the speaker embeddings that we trained in Chapter 3. Furthermore, our study puts a special focus on the possibility of improving internal speaker representations as the conversation takes place, which opens the possibility for autonomous unsupervised learning.

5.2 Related work

In this section, we discuss relevant background work in speaker diarization. We also define the most common metric with which speaker diarization systems are evaluated: the diarization error rate.

5.2.1 Multi-stage approaches

Most dependable diarization systems consist of a cascade of several steps (Anguera et al., 2012; Diez et al., 2018) as depicted in Figure 5.2: voice activity detection to discard *non-speech* regions, speaker embedding (Madikeri et al., 2015; Snyder et al., 2018) to obtain discriminative speaker representations, and clustering (Diez et al., 2018; Lin et al., 2019; Landini et al., 2020) to group speech segments by speaker identity. The main limitation of this family of *multi-stage* approaches lies in the handling of overlapped speech, which is known to be one of its major weak points (Anguera et al., 2012). In fact, common approaches simply ignore the problem, or address it *a posteriori* as a final post-processing step based on a dedicated overlapped speech detection module (Otterson and Ostendorf, 2007; Bullock et al., 2020; Horiguchi et al., 2021; Bredin and Laurent, 2021).

5.2.2 End-to-end approaches

A new family of approaches has recently emerged, rethinking speaker diarization completely. Named end-to-end diarization (EEND), the main idea of this approach is to train a single neural network (in a permutation-invariant manner) to ingest an audio recording and produce the overlap-aware diarization output directly (Fujita et al., 2019; Fujita et al., 2019). Despite being competitive with *multi-stage* approaches, the main limitation of the *overlap-aware end-to-end* methods is the strong assumption that the number of speakers is upper bounded or even known *a priori*. While reasonable for some particular use cases (*e.g.* one-to-one phone conversations), this assumption does not hold in many other situations (*e.g.* physical meetings or conference calls).

One solution to this problem is to augment *end-to-end* approaches with mechanisms to automatically estimate the number of speakers. For instance, EEND-

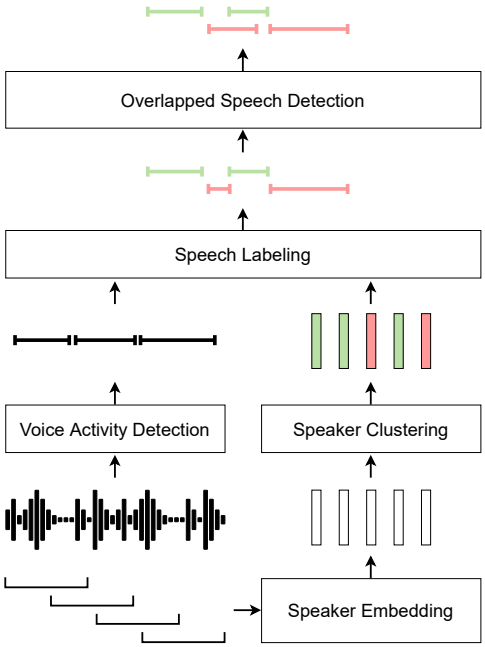


Figure 5.2: Multi-stage speaker diarization consists of voice activity detection, speaker embedding and clustering. In some cases, an overlapped speech detection module is added as well.

EDA (Horiguchi et al., 2020) extends EEND (Fujita et al., 2019; Fujita et al., 2019) with a recurrent encoder-decoder network to generate a variable number of *attractors*, which can be considered similar to speaker centroids. *Multi-stage* approaches usually do not suffer from this limitation, as they rely on a clustering step for which a growing number of techniques exists to accurately estimate the number of speakers (Park et al., 2019).

Following Fujita et al. (2019) and Bredin and Laurent (2021), end-to-end speaker diarization (also called *segmentation*) is modeled as a multi-label classification problem. In our case, we use model m from Bredin and Laurent (2021), which is trained to ingest a 5s audio chunk x and produce speaker activity probabilities $m(x) = \{s_1, \dots, s_F\}$ as depicted in Figure 5.3, where F is the number of output frames and $s_f \in [0, 1]^{K_{\max}}$, with $K_{\max}$ the estimated maximum number of different speakers in an input.

Since the diarization error rate is not affected by speaker naming (more on this in Section 5.2.4), any permutation of speakers in the output is essentially equivalent. In order to avoid any unwanted association of speakers to output indices, model gradients are computed on the speaker permutation that minimizes the binary cross entropy loss. This strategy, initially proposed in the context of source

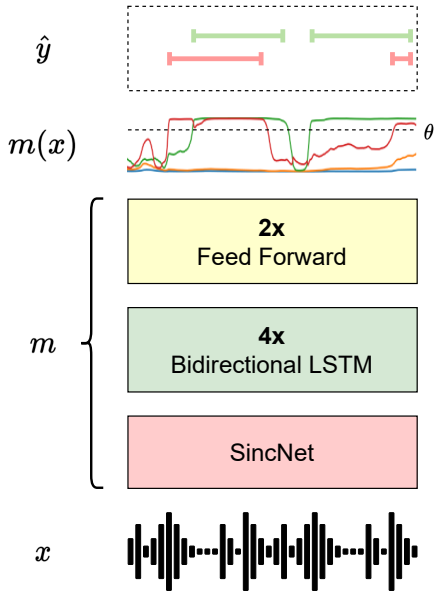


Figure 5.3: The speaker segmentation model m takes an input x and outputs speaker activity probabilities $m(x)$. Predictions $\hat{y}$ are obtained by binarizing $m(x)$ with a fixed threshold θ. Different output colors denote different speakers.

separation (Yu et al., 2017; Kolbæk et al., 2017) and later adapted to speaker diarization (Fujita et al., 2019), is known as *permutation-invariant training*, and its loss function is defined as:

$$\mathcal{L}(y, m(x)) = \min_{\text{perm} \in \mathcal{P}} \mathcal{L}_{\text{BCE}}(\text{perm}(y), m(x)) \tag{5.1}$$

where y is the reference annotation for x, $\mathcal{L}_{\text{BCE}}$ is the frame-wise binary cross entropy loss and $\mathcal{P}$ the set of all possible speaker permutations of y.

Since the optimal permutation of speakers (the one with lowest $\mathcal{L}_{\text{BCE}}$) can be defined as a one-to-one assignment problem, it is normally determined using the Hungarian algorithm (Kuhn, 1955), whose time complexity is $O(K_{\text{max}}^3)$. Consequently, in the interest of accelerating the training process, it is convenient to choose a low value for K_{max}. In a speaker count analysis performed by Bredin and Laurent (2021) on the composite training set made of DIHARD III (Ryant et al., 2020b), VoxConverse (Chung et al., 2020) and the AMI meeting corpus (Carletta, 2007) (described in more detail in Section 5.4.4), the authors found that 99% of all 5s audio chunks contained 4 speakers or less. For this reason, we also choose to set $K_{\text{max}} = 4$.

5.2.3 Online speaker diarization

As mentioned in the introduction, the primary focus of our study is low-latency online speaker diarization, which differs from its offline counterpart in several ways. In particular, while the latter assumes that the whole audio sequence is available at once (and hence can rely on multiple passes over the entire recording to output its final prediction), the former ingests a possibly infinite audio stream and can only afford a short delay between when it receives a buffer of audio and when it outputs the corresponding prediction (without the option to correct it afterwards).

These additional constraints make state-of-the-art *multi-stage* approaches like VBx (Landini et al., 2022) unfit for this setting, as they heavily rely on the possibility to pass several times over the audio sequence. EEND-like approaches are not suitable either because they expect large chunks of audio (30 seconds or more), leading to a prohibitively high latency. One notable exception is FlexSTB (Xue et al., 2021b) that astutely relies on an adaptive internal buffer to both simulate large audio chunks and support low (1s) latency.

5.2.4 Evaluation

Speaker diarization is typically evaluated using the diarization error rate (DER). This metric is defined as the sum of three types of error: false alarm (FA), missed detection (Miss), and speaker confusion (Conf). False alarm and missed detection are complementary error metrics defined as the duration of non-speech labeled as speech, and of speech labeled as non-speech, respectively. In case of correct speech detection, speaker confusion represents the duration of speech that is labeled with an incorrect speaker. More formally, the DER is defined as follows:

$$\text{DER} = \frac{FA + Miss + Conf}{\sum_s^S \text{duration}(s)} \tag{5.2}$$

where S is the set of all speech segments in the ground truth (irrespective of the speaker). In this context, note that false alarm errors can lead to a DER > 1. A visual explanation of the different error types is shown in Figure 5.4.

Since the speaker sets between the ground truth and the prediction can differ, the calculation of the DER is always preceded by a speaker mapping step in which both speaker sets are aligned. A common method to solve this alignment is to use the well-known Hungarian algorithm (Kuhn, 1955) to find the one-to-one mapping with minimal speaker confusion. This mapping procedure is depicted in Figure 5.4 on simple prediction and ground truth labels.

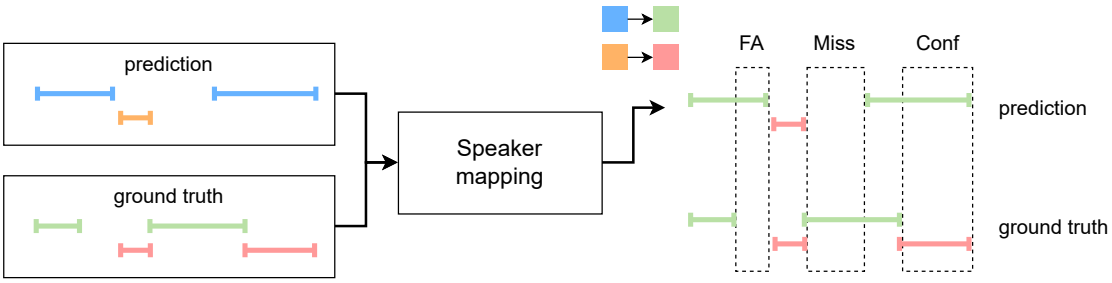


Figure 5.4: Computation of the diarization error rate (DER). First, a mapping between prediction and ground-truth speakers is computed to find a common labeling scheme. Second, the DER is determined using false alarm (FA), missed detection (Miss) and speaker confusion (Conf) errors.

In order to account for human errors in the annotation process, some evaluation protocols in the literature employ what is commonly known as a *forgiveness collar*, which is a short duration (*e.g.* 250ms on each side) around the ground truth segment boundaries that is ignored when computing the DER.

5.3 Building blocks

In our work, we propose to combine *the best of both worlds* (Kinoshita et al., 2021) and meet half-way between *multi-stage* and *overlap-aware end-to-end* diarization. To this end, we design a multi-stage pipeline where overlapped speech is a first-class citizen in every single step.

As depicted in Figure 5.5, we address online speaker diarization as the iterative interplay between two main steps: segmentation and incremental clustering. Every few hundred milliseconds (referred to as the *step*), the segmentation module first performs a fine-grained overlap-aware diarization of a 5s rolling buffer (small enough to reasonably estimate an upper bound on the local number of speakers K_{max}). This local diarization is then ingested by the incremental clustering module that relies on speaker embeddings to map local speakers to the appropriate global speakers (or create new ones), before updating its own internal state. As we will see in Section 5.4.3, since the step constitutes a lower bound for the latency of the entire system, we choose to use a step of 500ms, which allows our approach to work in scenarios requiring up to half the latency of the current state-of-the-art system FlexSTB (Xue et al., 2021b).

In this section, we describe the two main building blocks on which our online system is based: speaker segmentation and overlap-aware speaker embedding extraction.

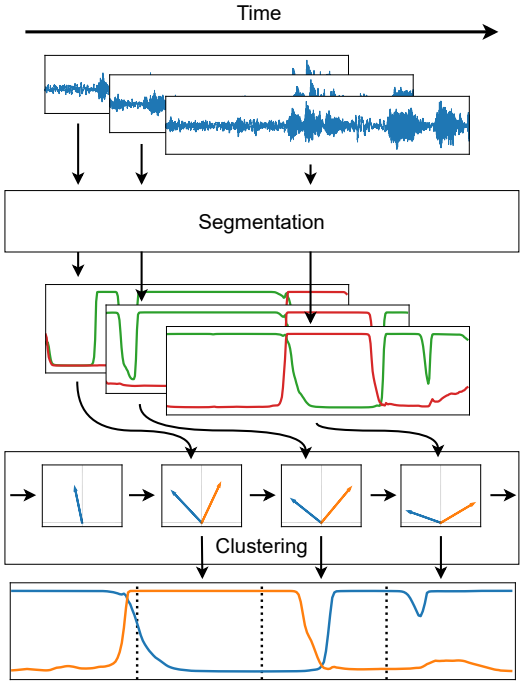


Figure 5.5: Proposed online speaker diarization approach.

5.3.1 Speaker segmentation

Our method relies on the speaker segmentation model introduced by Bredin and Laurent (2021) and summarized in Section 5.2.2. However, our work addresses a very different problem with radically different constraints. While Bredin and Laurent (2021) perform local offline speaker diarization of extremely short 5s chunks of audio, we address online speaker diarization of (possibly infinite) audio streams. Hence, our work extends Bredin and Laurent (2021) with a mechanism to track speakers over the duration of a conversation, with a latency much lower than 5s and real-time processing.

The segmentation step is the direct application of this neural network, which is used to obtain a fine-grained local speaker diarization. As shown in Figure 5.6, speakers whose activity probability exceeds a tunable threshold τ_{active} at least once during the chunk constitute the set of local speakers, and any inactive speakers are simply discarded. Active speaker probabilities are then passed unchanged (i.e. with continuous values between 0 and 1) to the incremental clustering step. In particular, it means that overlapping speech (i.e. when two or more speakers have high activity probabilities simultaneously) is handled from the very beginning of the pipeline. This is in contrast with most dependable speaker diarization ap-

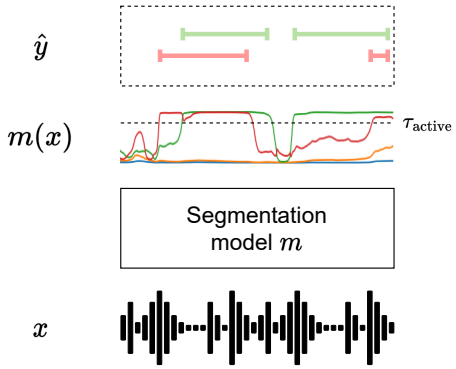


Figure 5.6: Binarization of the local segmentation output with τ_{active}.

proaches that handle overlapping speech as a post-processing step (Landini et al., 2020; Bredin and Laurent, 2021). This early detection of overlapping speech will prove very useful for the incremental clustering module.

5.3.2 Overlap-aware speaker embedding

Like most recent speaker diarization systems, we rely on neural speaker embeddings to represent and compare speakers. We use the same model architecture as in our speaker verification experiments from Chapter 3 (illustrated in Figure 5.7), which is based on the canonical x-vector architecture, trained to ingest a short audio extract and produce a single speaker embedding.

Contrary to this design choice, one of the requirements of our proposed system is to be able to extract multiple embeddings from the rolling buffer (one per speaker). A simple solution is to exploit the predicted speaker segmentation to split the rolling buffer into single-speaker areas and extract embeddings from those regions. However, extracting meaningful information from very short audio can be extremely challenging.

This is why we propose to modify the statistics pooling layer (Snyder et al., 2018) (see Figure 5.7) to return the concatenation of weighted mean μ_k and weighted standard deviation σ_k for each active speaker k (instead of the regular mean μ and standard deviation σ):

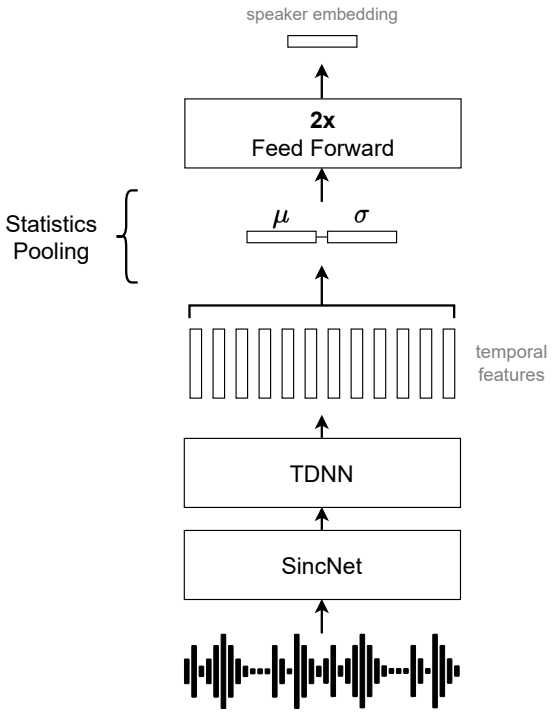


Figure 5.7: End-to-end speaker embedding model architecture (derived from x-vector) that we used in Chapter 3.

$$\begin{aligned} \mu = \frac{\sum_f x_f}{F} \quad &\longrightarrow \quad \mu_k = \frac{\sum_f w_{fk} \cdot x_f}{\sum_f w_{fk}} \\ \sigma^2 = \frac{\sum_f (x_f - \mu)^2}{F - 1} \quad &\longrightarrow \quad \sigma_k^2 = \frac{\sum_f w_{fk} \cdot (x_f - \mu_k)^2}{\left(\sum_f w_{fk}\right) - \frac{\sum_f {w_{fk}}^2}{\sum_f w_{fk}}} \end{aligned} \tag{5.3}$$

where x_f is the output of frame f of the last time-delay neural network (TDNN) layer. One straightforward option is to derive w_{fk} from the speaker activity probabilities s_{fk} obtained from the segmentation model and set $w_{fk} = s_{fk}$ directly. This way, the final (pooled) speaker embedding mostly relies on frames where the segmentation model is confident that speaker k is active. This generates exactly one embedding per active speaker in the current buffer, even when split into multiple speech turns (*e.g.* the green speaker in Figure 5.8).

However, as explained by Bredin and Laurent (2021), the segmentation model is also very good at detecting overlapped speech regions (where two or more speakers are active simultaneously). Therefore, another option is to make the speaker

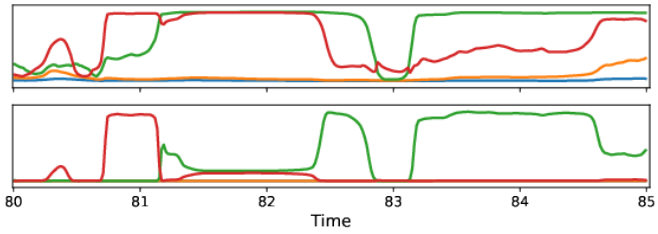


Figure 5.8: Effect of Equation 5.4 with $\gamma = 3$ and $\beta = 10$. Top row is the actual output s_f of the segmentation model on a 5s excerpt of file `DH_DEV_0007` from DIHARD III. Bottom row depicts weights w_f used for statistics pooling. Both low confidence and overlapped speech regions are weighed down.

embedding focus on frames where it is confident that speaker k is the *only* active speaker:

$$w_f = \left(s_f \cdot \underset{k}{\text{softmax}}\left(\beta \cdot s_f\right)\right)^{\gamma} \tag{5.4}$$

The effect of this transformation is illustrated in Figure 5.8. The use of softmax weighs down frames where two or more speakers are active, and the exponent $\gamma > 1$ weighs down frames where the segmentation model is not confident about the activity of a speaker. Embeddings extracted with this weighing scheme are called *overlap-aware speaker embeddings* in the rest of the chapter.

This entire procedure is illustrated in Figure 5.9. To summarize, we exploit information from the speaker segmentation model to compute per-speaker statistics, allowing the extraction of multiple embeddings from a single rolling buffer.

5.4 Proposed continual learning approach

In this section, we describe the various sub-components that constitute our incremental clustering approach. As mentioned previously, our goal is to progressively refine speaker representations as time passes in order to better recognize the speakers in the target conversation. Finally, we describe the experimental protocol we use to assess the effectiveness of our approach, and we present and discuss the results we obtain.

5.4.1 Constrained incremental clustering

Because the segmentation model is trained in a permutation-invariant manner and applied locally to the rolling buffer, one cannot guarantee that one particular

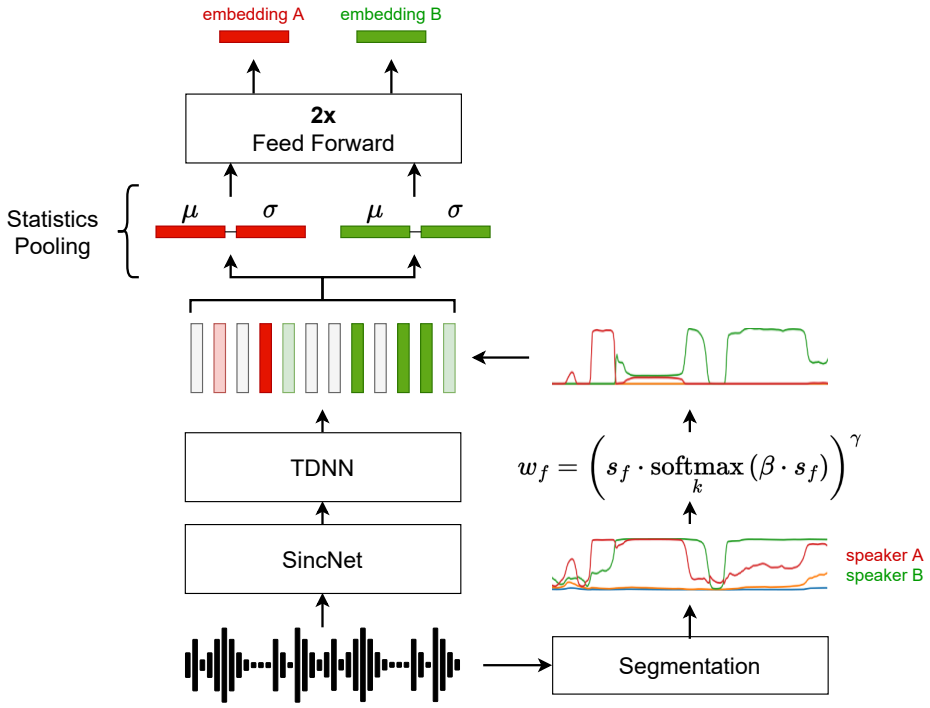


Figure 5.9: Extraction of overlap-aware speaker embeddings. The speaker activity probabilities are used to calculate the weighted mean and standard deviation per speaker in the statistics pooling layer of the speaker embedding model.

speaker consistently activates the same index over time. Figure 5.10 illustrates this limitation for two states of the rolling buffer: despite being only 500ms apart from each other and therefore having most of their audio content in common, notice how both active speakers are swapped. In this section, we describe how we use incremental clustering to circumvent this limitation by tracking speakers (and detecting new ones) over the whole duration of the audio stream.

Given the initial content of the rolling buffer, the segmentation and embedding steps are combined to extract one embedding for each active speaker in the first 5s of the audio stream. These speaker embeddings $\{c_1, \ldots, c_K\}$ are stacked to form the initial centroid matrix C with shape $K \times D$ where K is the number of active speakers so far, and D is the dimension of the speaker embeddings.

Every few hundred milliseconds (in our case 500ms), the rolling buffer is updated, and the segmentation and speaker embedding steps are combined to extract one embedding for each of the $K_{\text{buffer}} \leq K_{\text{max}}$ locally active speakers. These K_{buffer} speaker embeddings are then compared to the current state of the centroid matrix C to find the optimal mapping m^* between local and global speakers. Denoting

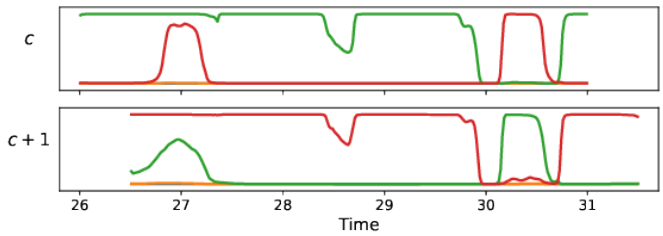


Figure 5.10: Real output s_f of the segmentation model on two consecutive positions of the 5s rolling buffer in file `DH_DEV_0001` from DIHARD III. Because of permutation-invariant training, green and red speakers are swapped.

$d(c, e)$ the distance between centroid c and local speaker embedding e, one option is to assign the kth local speaker to the closest centroid:

$$m^*(k) = \underset{c \in C}{\operatorname{argmin}}\, d(c, e_k) \tag{5.5}$$

Yet, this simple option does not take full advantage of the output of the segmentation model, as two local speakers might end up being assigned to the same centroid. This would be in contradiction to the output of the segmentation model that already chose to discriminate local speakers. Therefore, we add the constraint that any two local speakers cannot be assigned to the same centroid, while keeping the objective of minimizing the overall distance between local speakers and their assigned centroids:

$$m^* = \underset{m \in \mathcal{M}}{\operatorname{argmin}} \sum_k d(m(k), e_k) \tag{5.6}$$

where $\mathcal{M}$ is the set of mapping functions between local speakers and centroids with the following property:

$$k \neq k' \implies m(k) \neq m(k') \tag{5.7}$$

In practice, this optimal mapping is obtained by applying the Hungarian algorithm (Kuhn, 1955) on the speaker-to-centroid distance matrix, and can be seen as an incremental clustering step with *cannot-link* constraints.

5.4.2 Detecting new speakers and updating centroids

One of the many difficulties of online diarization lies in the detection of new speakers that can appear at any time. In order for our system to be effective, it must be able to detect them and re-identify them in the future. Moreover, since the available audio at the beginning of the conversation is very short, even the initial

centroids are expected to lack sufficiently discriminative information to correctly represent the target speakers. This is why our incremental clustering approach relies on continual adaptation by progressively refining the centroid matrix C with local information.

Once the optimal mapping m^* is determined, for any given local speaker k and their local embedding e_k

- if $d(m^*(k), e_k) > \delta_{\text{new}}$, they are marked as *new speaker* (i.e. it is the first time they are active since the beginning of the audio stream) and their embedding e_k is appended to the centroid matrix:

$$C \leftarrow [C; e_k]$$

- otherwise, they are marked as *returning speaker*, and their embedding e_k is used to update the corresponding centroid.

A visual example contrasting a returning speaker against a new speaker is shown in Figure 5.11.

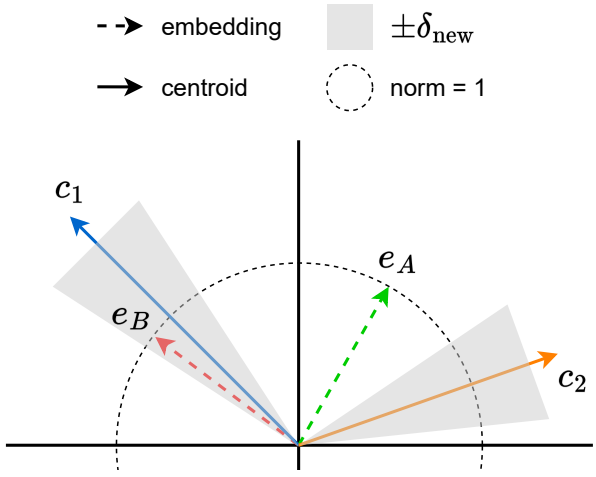


Figure 5.11: Speaker recognition in incremental clustering. Given that $d(c_1, e_A) > \delta_{\text{new}}$ and $d(c_2, e_A) > \delta_{\text{new}}$, local speaker A is considered a new global speaker that has not been seen before. In contrast, since $d(c_1, e_B) \leq \delta_{\text{new}}$, local speaker B is detected as global speaker 1.

Because of the weighing scheme described before, the quality of a speaker embedding e_k is expected to be positively correlated with the estimated duration during which local speaker k is active: $\Delta_k = \sum_f s_{fk}$. Therefore, we propose to update a centroid only when this duration is long enough:

$$c_{m^*(k)} \leftarrow \begin{cases} c_{m^*(k)} + e_k & \text{if } \Delta_k > \rho_{\text{update}} \\ c_{m^*(k)} & \text{otherwise} \end{cases} \tag{5.8}$$

where ρ_{update} is the minimum duration below which a speaker embedding e_k is considered to be too noisy to help refine the centroid. Equation 5.8 assumes that speaker embeddings e_k are unit-normalized and optimized for cosine similarity.

It is worth noting that in the categorization of continual learning methods discussed in Chapter 2, our system could be defined as a type of memory-based approach (see Section 2.4.2) in which the external memory is a progressively refined speaker centroid matrix. However, our approach possesses two key advantages with respect to memory-based methods. First, since the refinement of speaker centroids is not tied to gradient-based learning, which is usually linked to catastrophic forgetting because of drastic model parameter shifts (Hadsell et al., 2020), we effectively avoid one of the main causes of forgetting. Second, the size of the centroid matrix does not grow uncontrollably if left unbounded, as the number of speakers in a conversation is typically rather low. As a result, our system may also be a viable alternative in low-resource scenarios, where a low memory footprint is key.

5.4.3 Latency adjustment

Even though the whole $[t - 5s, t]$ buffer is used to extract embeddings and assign local speakers to an existing (or new) centroid, only the (active) speaker activity probabilities s_{fk} at its rightmost part $[t - \lambda, t]$ are output: λ effectively controls the latency of the whole system.

As mentioned in the introduction, the latency is defined as $\lambda = \lambda_{\text{step}} + \lambda_{\text{context}}$. The lowest possible value for λ corresponds to setting $\lambda_{\text{context}} = 0$, and hence $\lambda = \lambda_{\text{step}}$ (500ms in our case). In this configuration, the rightmost parts of two consecutive buffer states $[t - 5s, t]$ and $[t + \lambda - 5s, t + \lambda]$ do not overlap: $[t - \lambda, t]$ and $[t, t + \lambda]$. Therefore, they are simply concatenated, and frame-level speaker activity probabilities are passed through a final thresholding step. Local speaker k is marked as active at frame f if $s_{fk} > \tau_{\text{active}}$.

The careful reader might have noticed that at the very beginning of the audio stream, the initial buffer must be filled entirely before a first output can be provided – effectively leading to a much larger latency of 5s, an order of magnitude larger than the promised $\lambda = 500$ms. However, once this initial 5s warm-up period has passed, the latency is indeed $\lambda = 500$ms. If having a low latency from the very beginning of the stream is critical, one can simply add zero padding in the initial range $[0, \lambda]$. In practical terms, this is essentially equivalent to a scenario where

nobody speaks during the first 4.5s (at most), which should pose no particular difficulty for our system. As a result, we believe this should have minimal impact in performance, especially in longer conversations.

Figure 5.12 shows that, for cases where a longer latency λ is permitted, several positions of the rolling buffer can be combined in an ensemble-like manner to obtain a more robust output. In practice, for a given frame f, the final speaker activity probabilities are computed as the average of the speaker activity probabilities obtained from each buffer position.

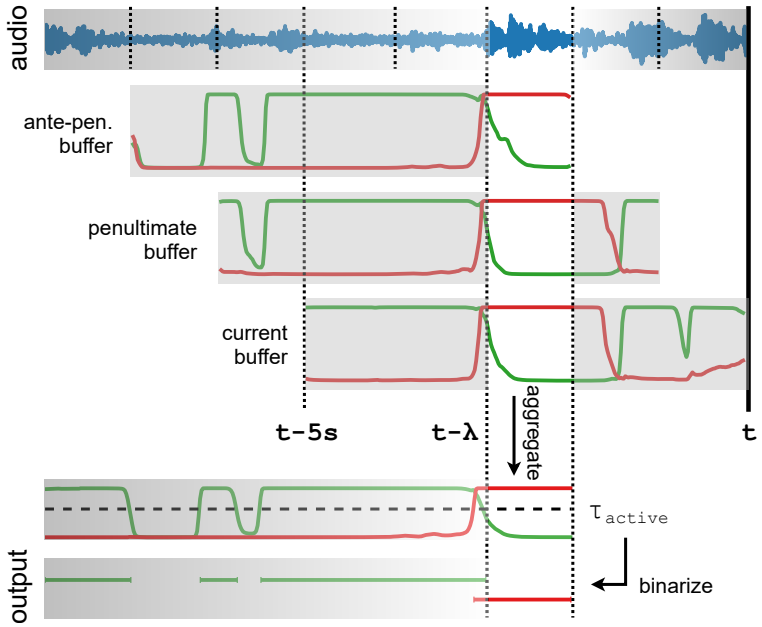


Figure 5.12: Depending on the allowed latency λ (between 500ms and 5s), multiple windows of the rolling buffer can be aggregated to obtain a (hopefully better) prediction. Speakers are colored according to their assigned clusters.

5.4.4 Experiments

In this section, we describe the corpora we use: DIHARD III (Ryant et al., 2020b,a), VoxConverse (Chung et al., 2020) and AMI (Carletta, 2007), as well as our experimental protocol and implementation details.

The DIHARD III corpus

DIHARD III (Ryant et al., 2020b,a) is the third installment in a series of corpora associated to the DIHARD speaker diarization challenges, which attempt to provide a standard evaluation corpus for single-channel diarization systems.

One of the motivations of DIHARD is to provide diarization scenarios in several challenging conditions (*e.g.* in terms of recording environment, ambient noise, reverberation, number of speakers, etc.) and from a wide variety of domains (*e.g.* meetings, audiobooks, restaurant conversations, etc.).

The third installment of the corpus is built on its previous version, providing two additional domains: *conversational telephone speech* (CTS) and *clinical*. For convenience, we provide the entire list of domains in the corpus alongside a short description in Table 5.1. As noted by Ryant et al. (2020b), the manual annotation process in the creation of DIHARD II was extremely slow and costly. Consequently, DIHARD III is first transcribed on a speaker-turn basis and then automatically force-aligned to establish time boundaries. At the same time, recycled recordings from DIHARD II are kept with their existing manual annotations.

Domain	**Description**
Audiobooks	English recordings of speakers reading book passages
Broadcast interview	Radio interviews from the decade of 1970
Clinical	Interviews for the diagnosis of autism in children
Courtroom	Arguments from the 2001 term of the U.S. Supreme Court
CTS	10-minute telephone conversations between two native English speakers
Maptask	A "leader" speaker telling a map route to a "follower" speaker
Meeting	Meeting recordings
Restaurant	Restaurant conversations
Socio field	Sociolinguistic interviews in field conditions
Socio lab	Sociolinguistic interviews in quiet and controlled conditions
Webvideo	English and Mandarin videos collected from the Internet

Table 5.1: Short description of the domains included in DIHARD III as explained by Ryant et al. (2020a).

DIHARD III contains only development and evaluation subsets, each of them further divided into a *Core* set, balanced in terms of domain duration (around 2hs per domain), and a larger *Full* set, including all available data. More detailed information about the corpus and its subsets is shown in Table 5.2.

The VoxConverse corpus

Similarly to the VoxCeleb corpora (Nagrani et al., 2017; Chung et al., 2018) (see Section 3.3.2), the VoxConverse corpus is the result from an automatic data collection pipeline from YouTube videos consisting of several steps. First, a list of candidate videos is selected based on search terms like "panel debate" and "discussion", in order to maximize the chances of alternate speaking and overlap. Then,

	Development		Evaluation	
	Core	Full	Core	Full
Number of recordings	181	254	184	259
Duration (in hours)	23.9	34.2	22.7	33.0
% of speech	78.4	79.8	77.4	79.1
% of speech overlap	10.0	10.7	8.8	9.4

Table 5.2: Description of the DIHARD III corpus as presented by Ryant et al. (2020b). The percentage of overlapping speech is computed over the entire duration (counting both speech and non-speech).

an automatic process combining face tracking, mouth movements and the audio track is used to determine when each person speaks. Finally, the data undergoes a manual verification procedure to ensure that the desired annotation quality is met.

In the same way as DIHARD III, VoxConverse provides only two subsets: development and evaluation, each further described in Table 5.3.

	Development	Evaluation
Number of recordings	216	232
Duration (in hours)	20.3	43.5
% of speech	93.2	89.6
% of overlap	3.8	3.1

Table 5.3: Description of the VoxConverse corpus as presented by Chung et al. (2020). The percentage of overlapping speech is computed over the entire duration (counting both speech and non-speech).

The AMI meeting corpus

The Augmented Multi-party Interaction (AMI) corpus (Carletta, 2007) (not to confuse with the automatic misogyny identification corpus introduced in Chapter 3) is the result of collecting a combination of real and scenario-driven meetings in English, recorded with a variety of microphones and video cameras in three different rooms. The corpus amounts to a total of 100 hours of audio and is labeled for a wide variety of tasks, including transcriptions, named entities, summaries, emotions, locations, among others.

The corpus provides official training, development, and test subsets of the Mix-

	Training	Development	Evaluation
Number of recordings	136	18	16
Duration (in hours)	80.7	9.7	9.1
% of speech	81.9	78.5	80.4
% of overlap	13.2	13.4	15.8

Table 5.4: Description of the AMI meeting corpus. The percentage of overlapping speech is computed over the entire duration (counting both speech and non-speech).

Headset audio files (Carletta, 2007) that are further described as the *Full* partitioning in (Landini et al., 2022). This data consists of the combination of the recordings from the participant headsets (instead of the far-field microphones) and is the data we use in all the experiments of this chapter. Further details about the composition of the corpus are shown in Table 5.4.

Experimental protocol

DIHARD III (Ryant et al., 2020b,a) does not provide a training set. Therefore, we split its development set into two parts: 192 files used as training set, and the remaining 62 files used as a smaller development set. For simplicity, the latter is referred to as the development set in the rest of the chapter. When defining this split[1], we made sure that the 11 domains were equally distributed between both subsets. The test set is kept unchanged. We also report performance on *DIHARD II* (Ryant et al., 2019) for comparison with FlexSTB (Xue et al., 2021b).

VoxConverse does not provide a proper training set either (Chung et al., 2020). Therefore, we also split its development set into two parts: the first 144 files (`abjxc` to `qouur`, in alphabetical order) constitute the training set, leaving the remaining 72 files (`qppll` to `zyffh`) for the actual development set. Furthermore, multiple versions of the VoxConverse test set have been circulating: we rely on version `0.0.2`[2].

AMI provides an official {training, development, test} partition of the Mix-Headset audio files (Carletta, 2007) (*Full* partitioning described in Landini et al. (2022)), so we use it in our experiments without modifications.

While the same pre-trained segmentation and embedding models were used for all three corpora, we rely on their respective development sets to optimize hyper-parameters (τ_{active}, ρ_{update} and δ_{new}) specifically for each corpus. More precisely, we

[1]available at `huggingface.co/pyannote/segmentation`
[2]available at `github.com/joonson/voxconverse`

use the `pyannote.pipeline` optimization toolkit that relies on a tree-structured Parzen estimator algorithm (Bergstra et al., 2011) to minimize the overall diarization error rate – computed with `pyannote.metrics` (Bredin, 2017a) without any forgiveness collar and including overlapped speech regions.

We also compare our system against several approaches. On the one hand, we define the VBx system (Landini et al., 2022), a multi-stage approach based on hidden markov models and x-vector, as the offline topline, as it currently holds the state-of-the-art in offline speaker diarization. However, since this system does not take into account overlapping speech, we also report the results of Bredin and Laurent (2021), who extend VBx with overlap-aware re-segmentation. On the other hand, our baseline system is the EEND-based FlexSTB (Xue et al., 2021b), which extends EEND (Fujita et al., 2019) with a speaker-tracing buffer for online diarization.

To ensure a fair comparison between all considered approaches, the optimization process is applied for all of them independently. In other words, it means that every *row* × *dataset* entry in Table 5.5 results from one dedicated optimization process. This includes the offline *topline*, the proposed online approach and its ablative variants, but excludes both FlexSTB (as we unfortunately did not have access to its implementation) and experiments on DIHARD II (where we use the hyper-parameter values tuned for DIHARD III).

Implementation details

We use a pre-trained segmentation model[3] that was trained on the concatenation of the training sets from AMI, DIHARD III, and VoxConverse. It ingests 5 second audio chunks and outputs one prediction every 16ms with $K_{\text{max}} = 4$ speakers. More details about the training process can be found in the work by Bredin and Laurent (2021) and in Section 5.3.1.

The speaker embedding model is the one presented in Section 3.3.3, based on the canonical x-vector TDNN-based architecture (Snyder et al., 2018) but with filter banks replaced by trainable SincNet features (Ravanelli and Bengio, 2018). We re-trained the architecture from scratch with the additive angular margin loss (Deng et al., 2019) using chunks of variable duration (from 2 to 5 seconds) drawn from VoxCeleb (Nagrani et al., 2017; Chung et al., 2018). We also augmented the training data with reverberation based on impulse responses from *EchoThief* and Traer and McDermott (2016), as well as additive background noise from the MUSAN (Snyder et al., 2015) corpus. To facilitate reproducibility and make related research possible, we also share the pre-trained speaker embedding

[3]available at `hf.co/pyannote/segmentation@Interspeech2021`

model and more details about the training process[4]. Furthermore, the weights used in the statistics pooling layer of the *overlap-aware speaker embeddings* were obtained with $\gamma = 3$ and $\beta = 10$. These values were not optimized with the rest of the hyper-parameters. Instead, we hand-picked them based on examples like the one in Figure 5.8.

5.4.5 Results and discussion

System	Latency	DIHARD III				AMI				VoxConverse				DIHARD II			
		FA	Miss.	Conf.	DER	FA	Miss.	Conf.	DER	FA	Miss.	Conf.	DER	FA	Miss.	Conf.	DER
topline1	∞	3.6	12.5	6.2	22.3	3.1	17.2	3.8	24.1	3.1	4.6	3.4	11.1	5.0	15.3	7.4	27.7
topline2	∞	4.7	9.7	4.9	**19.3**	4.3	10.9	4.7	**19.9**	4.6	3.0	3.5	**11.1**	5.6	13.5	7.1	**26.3**
ours	5s	5.3	10.0	9.7	**25.0**	5.0	10.0	12.4	**27.5**	3.8	4.9	8.2	**16.8**	5.7	14.0	14.4	**34.1**
ablation1	5s	4.6	11.3	9.3	25.3	3.0	16.0	11.6	30.5	4.1	5.1	11.2	20.4	5.1	15.5	13.6	34.3
ablation2	5s	2.1	1.4	6.9	10.4	1.0	1.1	15.5	17.7	0.5	0.7	9.1	10.3	2.2	1.6	12.0	15.8
baseline	1s	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	36.0
ours	1s	6.2	9.7	11.8	27.6	6.6	9.4	14.4	30.4	5.1	3.3	11.7	20.1	5.8$^\bullet$	14.4$^\bullet$	14.9$^\bullet$	**35.1**$^\bullet$

Table 5.5: DER obtained on test sets. FA, Miss. and Conf. stand for false alarm, missed detection and speaker confusion rates respectively ($^\bullet$ = hyper-parameters optimized with latency $\lambda = 1$s for fair comparison). Values in bold denote the best performing system for each latency (excluding the oracle-based *ablation2*).

Our main set of experiments is summarized in Table 5.5. We start by reporting the performance of the offline toplines *topline1* that consists of VBx (Landini et al., 2022), and of *topline2*, which adds the overlap-aware re-segmentation step introduced by Bredin and Laurent (2021). System *ours* represents our full proposed approach, and *ablation1* and *ablation2* are variants of *ours* with different characteristics. In *ablation1* Equation 5.4 is not applied, and in *ablation2* we replace the local segmentation model with an oracle segmentation. Finally, the performance of the online FlexSTB approach is reported as the *baseline* system.

Overlap-aware speaker embedding. The comparison between *ours* and the first ablative experiment *ablation1* shows that the overlap-aware weighing scheme introduced in Equation 5.4 brings a relative performance improvement of 10% on AMI, 18% on VoxConverse and 1% on DIHARD III. Given that they respectively contain 17%, 3%, and 11% of overlapped speech (computed over the total speech duration), there is still room for improvement on this specific aspect. In particular, while we hand-crafted this weighing scheme, it should be possible to train the segmentation and speaker embedding models jointly for the latter to fully take advantage of the former's capability at detecting and separating simultaneous speakers. Moreover, this would avoid the propagation of segmentation errors across modules.

[4]at hf.co/pyannote/embedding

Overlap-aware speaker segmentation. In the ablative experiment *ablation2*, we replace the segmentation model by an oracle that provides a perfect binary (*i.e.* $s_{fk} \in \{0, 1\}$) overlap-aware segmentation. As expected, missed detection is where most of the difference occurs (caused by overlapped speech), while speaker confusion only marginally improves. The problem of overlapped speech detection still represents a major weakness of both offline and online speaker diarization.

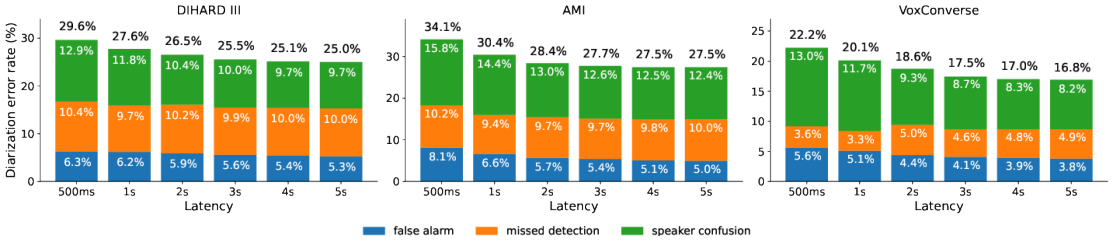


Figure 5.13: Impact of the latency on the overall DER performance.

Adjustable latency. In Figure 5.13, we show how the performance of our online approach is affected as we decrease the allowed latency from $\lambda = 5$s to $\lambda = 500$ms. The speaker confusion consistently increases as the latency decreases, while false alarm and missed detection remain constant. This can be explained by the ensemble-like aggregation process described in Section 5.4.3 that combines more views of the same problem as the allowed latency and the local future context duration increase. Note that we kept the hyper-parameters (τ_{active}, ρ_{update}, δ_{new}) optimized for latency $\lambda = 5$s and still obtain reasonable performance for lower latencies. However, it is also possible to re-optimize the hyper-parameters for a specific latency. This is what we did for the $\lambda = 1$s setting marked with $\bullet$ in Table 5.5 for comparison with the FlexSTB (Xue et al., 2021b) *baseline* system. Not only do we get better overall performance, but our approach also has the advantage of a lower memory footprint, as it never ingests nor runs inference on more than 5s of audio at a time (compared to 100s of FlexSTB) and keeps a single vector per speaker in memory (compared to 100s of acoustic features and per-speaker scores in FlexSTB). Furthermore, our approach with $\lambda = 3$s reaches the same performance as the official baseline (Ryant et al., 2020b) of the DIHARD III challenge (25.5% vs 25.4%), which addresses offline speaker diarization.

Continual learning. In order to understand whether our system progressively adapts to the target conversation, we compare the performance of *ours* over time against the offline *topline2* (with overlap-aware re-segmentation (Bredin and Laurent, 2021)) on DIHARD III. The results of this comparison are shown in Figure 5.14. While the performance of *topline2* remains relatively constant, our system keeps improving as conversations unfold, almost bridging the gap after 5 minutes of conversation. As new information becomes available, our system

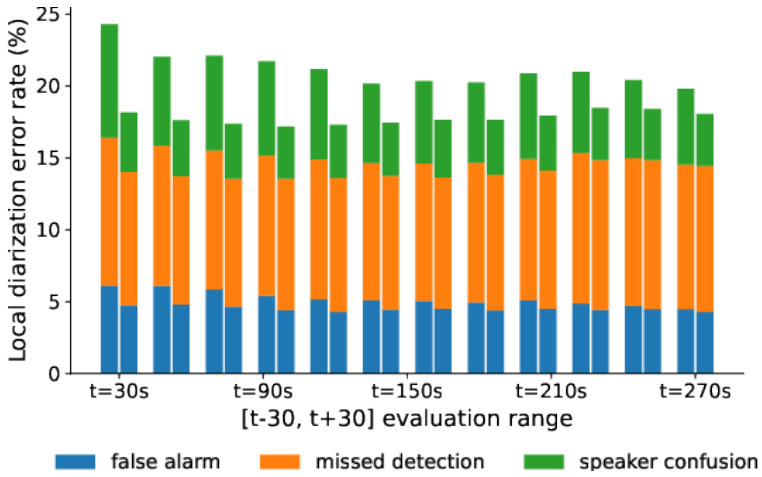


Figure 5.14: Evolution of DER as conversations unfold. Left: online approach *ours* with $\lambda = 5$s. Right: offline *topline2* (Bredin and Laurent, 2021). Local DER is computed on the 223 DIHARD III (test) conversations that are longer than 300s.

learns better speaker centroids, hence decreasing the speaker confusion. While very long conversations can become rather expensive (if not impossible) to process with most offline models, our system can handle daylong audio streams at a practically constant memory cost, while getting progressively better at the same time.

Speaker number estimation. As mentioned before, one of the key challenges in online diarization is the detection of new speakers. In Figure 5.15, we show the total number of speakers predicted by our 5s-latency system for each conversation and each (test) corpus. We observe that the predicted number of speakers tends to be underestimated in DIHARD III (Figure 5.15a), while the opposite is true in VoxConverse (Figure 5.15c) and AMI (Figure 5.15b). This highlights the necessity for better new-speaker detection techniques. In our system, new-speaker detection is mainly controlled by the δ_{new} hyper-parameter, which is optimized for each corpus and kept unchanged throughout the entire conversation. However, clusters may be spread differently depending on the target conversation and the speaker they represent. We believe that new-speaker detection errors (and hence speaker confusion) may be further reduced by adopting a flexible δ_{new}. A first step in this direction could be per-speaker δ_{new} values that are automatically determined from additional cluster statistics like the variance.

Offline vs. online. Finally, an important limitation of our approach is the gap to state-of-the-art offline solutions like *topline2*. Since *topline2* relies on the exact same pre-trained segmentation model as our proposed approach, most of the re-

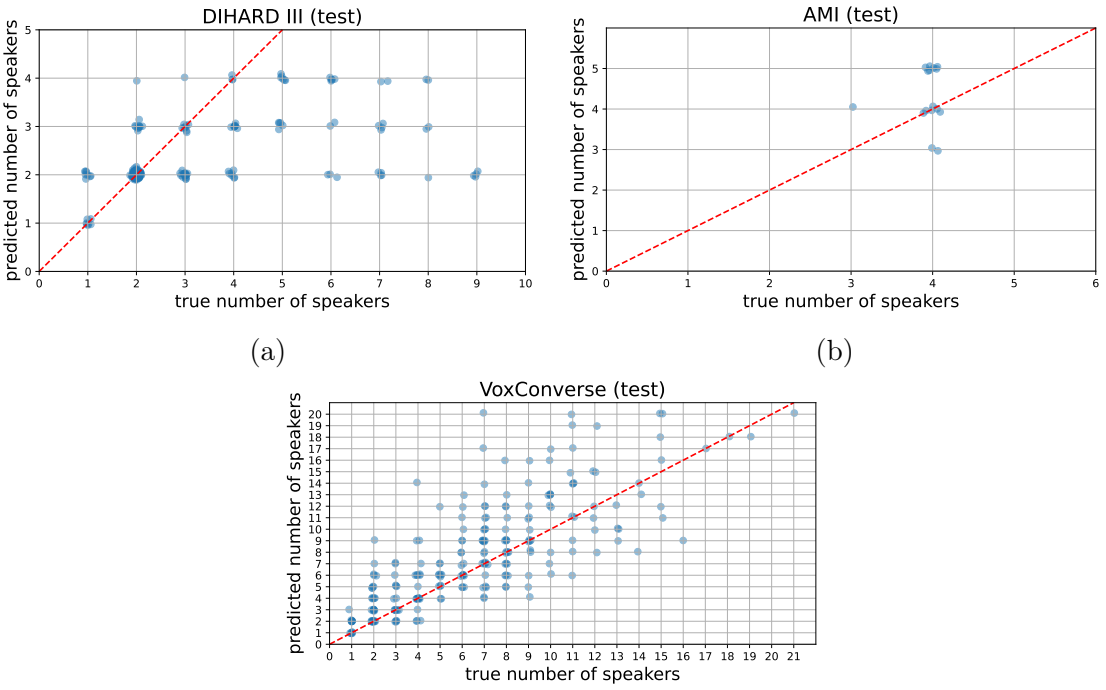


Figure 5.15: Comparison between predicted and true numbers of speakers for the test sets of (a) DIHARD III, (b) AMI and (c) VoxConverse. Each conversation is marked as a blue point, while the dashed red lines represent correct predictions.

ported decrease in performance with $\lambda = 5$s is caused by speaker confusion errors (relative +100% for DIHARD III, +160% for AMI, +130% for VoxConverse). Consequently, there is still room for improvement in order for incremental clustering to be on par with offline multi-pass clustering.

5.5 Conclusion

In this chapter, we have presented an overlap-aware online speaker diarization system combining the local speaker segmentation of a 5s-long rolling buffer with incremental clustering of task-specific speaker embeddings (as defined in Chapter 3). A wide array of experiments on the following three large speaker diarization corpora: DIHARD III (Ryant et al., 2020b), AMI (Carletta, 2007) and VoxConverse (Chung et al., 2020), allows us to reach the following conclusions.

First, our approach outperforms FlexSTB (Xue et al., 2021b) (a state-of-the-art online speaker diarization baseline), with a substantially lower memory consumption.

Furthermore, apart from handling overlapping speech at every stage, it benefits from an adjustable latency between 500ms and 5s by design, without the need for further training.

Second, and perhaps more importantly, our system is also capable of incremental learning, progressively bridging the gap to offline performance as conversations unfold by continually refining speaker centroids. In fact, our approach can also be considered a type of low-resource memory-based method to continual learning that does not depend on gradient-based learning, avoiding catastrophic forgetting issues coming from model parameter shifts. These advantages may also make our method preferable to an offline system when recordings are long and resources low.

Considering all of the above, we believe our system is particularly well-suited to continual adaptation in the production phase, as it is designed to learn in a completely autonomous and unsupervised manner.

Furthermore, our work also highlights several difficulties of continual learning in online speaker diarization, such as new-speaker detection and the necessity for more meaningful metrics. In particular, since speakers tend to be active throughout the entire conversation, finding an accurate definition of catastrophic forgetting in this context can be challenging. If we had to find an equivalent measure, we would define it as an increase in the confusion involving a particular speaker.

Since our system is designed not to erase speaker centroids, it is effectively unable to completely "forget" a speaker, even if it does not appear for long periods of time. Yet, if a centroid is assigned conflicting embeddings from different speakers, divergence is a likely outcome in the long run, where the divergent centroid would arrive at a position that is far from the true (but unknown) centroid. Although our system is likely to recover from this situation by creating a new centroid for the divergent speaker, we believe that an interesting direction for future work is the design of a proper measure of speaker divergence. As an example, one could run a second pass of the system using the centroid matrix obtained after the first run (skipping the centroid updates as defined in Section 5.4.1). In this context, a divergence metric could be defined as the ratio of speaker embeddings that are assigned to the correct centroid.

Chapter 6

Continual self-adaptation for speaker segmentation

"Uphold privacy and know that you can have it. Know that we must have it."
— Ann Cavoukian

One of the inconveniences of the system introduced in the previous chapter is that it is tightly coupled to the pre-trained speaker segmentation and embedding models, which are frozen throughout the whole adaptation procedure. This represents a considerable disadvantage with respect to gradient-based continual learning techniques. Contrary to speaker embeddings, which may be more robust to domain changes (as we saw in Chapter 3), development-production mismatch issues in the speaker segmentation model could be catastrophic to the adaptation capabilities of our approach, as it blindly trusts its output. In this chapter, we propose a training scheme to address this shortcoming by continually adapting the segmentation model to a target domain as new conversations become available.

Importantly, the approach that we propose in this chapter also benefits from autonomous self-supervised learning, and can even be defined as an *ephemeral learning* technique, in which sensitive and personally identifiable data is immediately discarded after training and never stored permanently. In order to learn continually without ground-truth annotations, we investigate the possibility of using the model's own predictions as a training signal in a similar manner to knowledge distillation (Hinton et al., 2015).

The content of the chapter is organized in the following way. In Section 6.1, we establish the context and motivation of our work. In Section 6.2, we propose an

initial self-supervised but non-continual training scheme as a first step towards our goal, and we show its effectiveness with respect to a pre-trained baseline. Next, in Section 6.3, we propose a continual learning extension to this training scheme, which also outperforms the pre-trained baseline and is even on par with the initial approach, while observing conversations one at a time sequentially. Finally, we conclude and provide final remarks in Section 6.4.

This chapter has also been the subject of the following scientific publication:

Juan M. Coria, Hervé Bredin, Sahar Ghannay, and Sophie Rosset. Continual Self-Supervised Domain Adaptation for End-to-End Speaker Diarization. In *IEEE Spoken Language Technology Workshop (SLT)*, Qatar, 2022.

6.1 Introduction

Whether multi-stage or end-to-end, speaker diarization models are generally trained to perform well on a given corpus with its own set of specific assumptions and properties (*e.g.* microphone quality, noise, speaker accent, language, etc.) shared among recordings, and that we typically refer to as a *domain*. However, as illustrated in Figure 6.1, given a model m_0 trained on a corpus $\mathcal{D}_0$ from source domain A, it is well known that the performance of m_0 on a given corpus $\mathcal{D}_1$ from a new target domain B is bound to be substantially worse than on data from domain A, a problem known as *domain mismatch*.

In domain adaptation, whose procedure is also depicted in Figure 6.1, the goal is to fix this mismatch by fine-tuning the *out-of-domain* model m_0 on the corpus $\mathcal{D}_1$ from target domain B, for which we want to obtain good performance. In particular, end-to-end training makes speaker diarization models suitable for domain adaptation because fine-tuning a single model is simpler than doing so for multiple modules. Nevertheless, domain adaptation remains expensive for two reasons:

- the large target-domain corpus $\mathcal{D}_1$ needs to be collected "a priori"
- $\mathcal{D}_1$ needs to be manually annotated

In this chapter, we study continual domain adaptation for the speaker segmentation model (also called end-to-end speaker diarization) introduced in the previous chapter. We propose a continual and fully self-supervised training scheme that achieves an average 17% relative improvement over a pre-trained baseline without a single manually annotated conversation. Our approach also rivals (and sometimes outperforms) non-continual variants trained on the whole target domain at once. Furthermore, since only a single conversation at a time is used for training,

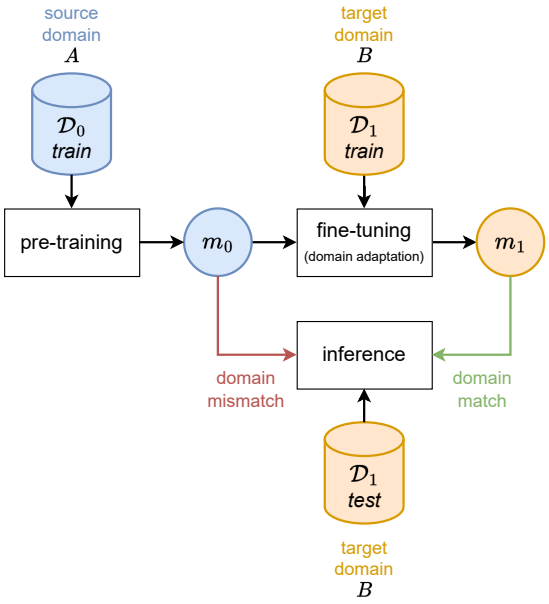


Figure 6.1: Domain mismatch and domain adaptation.

every new conversation can be discarded as soon as it is processed, avoiding any potential unwanted access.

6.1.1 Removing the need for manual annotation

A straightforward solution to avoid the need for manual annotations is to leverage unsupervised learning techniques. Recently, self-supervised methods for neural network training have gained enormous popularity because of their ability to find useful training signals from unlabeled data. In fact, many popular models trained in a self-supervised manner rely on auxiliary tasks derived from the underlying structure of the data, which usually consist in predicting artificially missing or distorted parts of the input (Devlin et al., 2019; Chung and Glass, 2020; Chen et al., 2020).

However, deriving a useful training signal solely from auxiliary tasks can be extremely challenging in some cases, for example in the context of speaker diarization. The technique of pseudo-labels (Lee et al., 2013) does not depend on such auxiliary tasks. Instead, it is based on the concept of knowledge distillation (Hinton et al., 2015), where a training signal is obtained via a pre-trained *teacher* model m_t, whose output is used as ground-truth to train the *student* model m_s.

This method has shown great promise in end-to-end speaker diarization (Takashima

et al., 2021) by leveraging well-performing pre-trained models. A similar study on domain adaptation for speech enhancement (Tzinis et al., 2022) even goes one step further, showing that periodically updating m_t while fine-tuning m_s on the target domain can significantly increase performance.

6.1.2 Continual and ephemeral learning

The need to "a priori" collect a large target-domain corpus can be eliminated by relying on continual learning (Hadsell et al., 2020). As discussed in Chapter 2, this paradigm is defined by a training scheme in which the training corpora $\mathcal{D}_1, \ldots, \mathcal{D}_L \in \mathbf{S}$ appear in sequence as they become available. In our case, this happens in production, after the initial model development phase. In this chapter, we assume that $\mathbf{S}$ is sampled from a single (target) domain with its own particular set of properties, and that each $\mathcal{D}_i$ is a single conversation from this domain. We believe this represents a realistic use case scenario, for example in the context of a home assistant for speech and speaker recognition that may improve itself on the target household after every interaction with the involved users.

As we have seen in Chapter 2 and Chapter 4, the naive approach of keeping all past $\mathcal{D}_{j<i}$ (past conversations in our case) for future training raises both time and space costs substantially, and storing past data permanently may also be problematic or even impossible in some cases. In particular, audio conversations are usually regarded as sensitive and personally identifiable data that should be protected from third-party access, which complicates the storing and retrieval for model training. Our proposed approach can be considered *ephemeral*, as we forbid any storage of (and access to) past conversations in order to protect user privacy.

After sequential training on conversations from the target domain, we expect the system to perform well on both past and future conversations of that domain. As defined in previous chapters, the improvement on past conversations is usually referred to as *backward transfer*, while *forward transfer* is used to denote the improvement on future conversations. Unfortunately, gradient-based continual learning is prone to *catastrophic forgetting* (French, 1999; Hadsell et al., 2020), whereby the performance on past conversations sharply deteriorates as the model is trained on new ones.

However, if we consider our target scenario of an adapting home assistant, forgetting specific conversation details should not constitute a major problem as long as there is an overall improvement at speaker diarization in the target domain. For example, consider an assistant system controlling living-room appliances that is exceptionally moved outdoors and later positioned at its initial location indoors. Given the target domain of "living-room conversations", it would be unfair to pe-

nalize the model a few days later for losing performance on recordings with outdoor acoustics and noise from wind or cars. In other words, we allow our model to forget as long as it involves out-of-domain knowledge. For this reason, we choose to focus on forward transfer, where a well-performing system should benefit from a boosted performance in future target-domain conversations.

6.2 Self-supervised adaptation

In this section, we propose an initial non-continual self-supervised training scheme based on pseudo-labels that we consider a first step towards our continual domain adaptation goal. We first discuss background work related to the subject, then introduce our proposed approach and experimental protocol, and finally we present and discuss the results we obtain.

6.2.1 Related work

The technique of *pseudo-labels* (Lee et al., 2013; Kahn et al., 2020) constitutes an interesting alternative to address the manual annotation issue, as it consists in using the predictions of a pre-trained system as annotations in a teacher-student training scheme. This training procedure is usually linked to knowledge distillation (Hinton et al., 2015), in which a model m_s (known as the *student*) is trained to produce the predictions $\hat{y} = m_t(x)$ from a model m_t (known as the *teacher*), given the same input x. An illustration of this training scheme is shown in Figure 6.2.

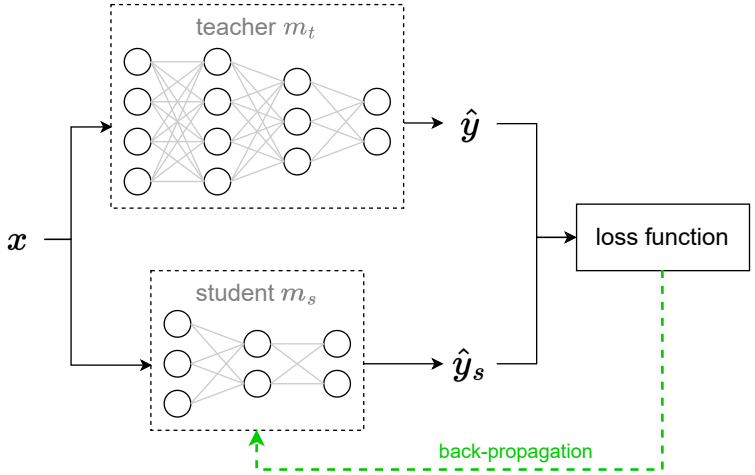


Figure 6.2: Teacher-student training schemes train a student model m_s to match the predictions $\hat{y}$ of a teacher model m_t.

Knowledge distillation has typically been applied to scenarios in which m_s is much smaller (in number of parameters) than m_t, for example to deploy m_s in low-resource devices (Sanh et al., 2019). In contrast, pseudo-labels were originally designed for semi-supervised scenarios to boost performance by leveraging both labeled and unlabeled data, and can be defined in more complex ways (*e.g.* by combining the output of multiple teachers).

As mentioned in the introduction, a similar idea has been applied to speaker diarization in Takashima et al. (2021). In their work, the authors propose to use a committee-based teacher model, whose pseudo-labels are a combination of predictions from multiple systems that are known to perform well on the task. In contrast, our training scheme requires that the teacher model m_t and the student model m_s be the same, so that m_s can progressively replace m_t in the continual learning extension we will discuss in Section 6.3. Consequently, even though it can provide a better training signal, continually training multiple models in a committee-based method brings additional computational costs (for both training and inference) that we prefer to avoid.

At the same time, a risk of the model being both teacher and student is divergence, as matching its own predictions may progressively reinforce errors, a phenomenon known as *confirmation bias*. Previous work has typically relied on introducing additional factors of variation to the teacher output $\hat{y}$ that do not interfere with the quality of the pseudo-label. For example, Takashima et al. (2021) rely on the fact that m_t is a combination of pre-trained models, as it is less likely that the entire set of teacher models will make the same mistakes. Alternatively, the approach of Tzinis et al. (2022) for speech enhancement proposes to recombine the signal and noise outputs of the teacher to form new artificial training examples, which can be seen as a form of augmentation.

6.2.2 Proposed approach

As in Chapter 5, we choose to use the end-to-end speaker segmentation architecture from Bredin and Laurent (2021) (illustrated in Figure 6.3 for convenience) as our pre-trained out-of-domain model m_0. As illustrated in Figure 6.4, m_0 is first used to produce the pseudo-labels for the entire set of conversations of the target domain. Only then, m_0 is trained using those same pseudo-labels as ground-truth. In this initial approach, we train m_0 on the concatenation of all target-domain conversations $[\mathcal{D}_1; \ldots; \mathcal{D}_L]$ at once, which makes it non-continual. As such, pseudo-labels are computed once at the beginning using m_0 (with a random augmentation *aug* as explained in the following paragraphs of this section) and they are never updated afterwards. This difference will prove to be an interesting discussion point with regards to pseudo-label quality in the comparison with the

continual learning extension that we introduce in Section 6.3.

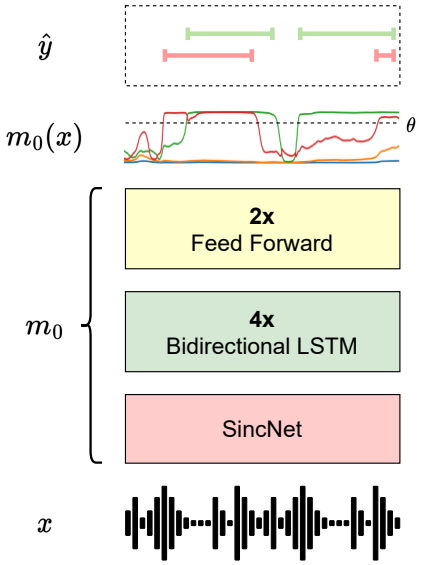


Figure 6.3: Architecture of the pre-trained out-of-domain segmentation model m_0 (Bredin and Laurent, 2021) (introduced in Chapter 5).

When defining the pseudo-labels $\hat{y}$, one could choose an exact copy of the continuous $m_0(x) \in [0, 1]^{K_{\max} \times F}$, so that $\hat{y} = m_0(x)$. However, as evidenced by the real example of $m_0(x)$ shown in Figure 6.3, $m_0(x)$ can be rather noisy and fail to provide a useful training signal in some input regions. Consequently, we define $\hat{y}$ as a binary version of $m_0(x)$ using a threshold θ, which we manually set to $\theta = 0.5$ for simplicity.

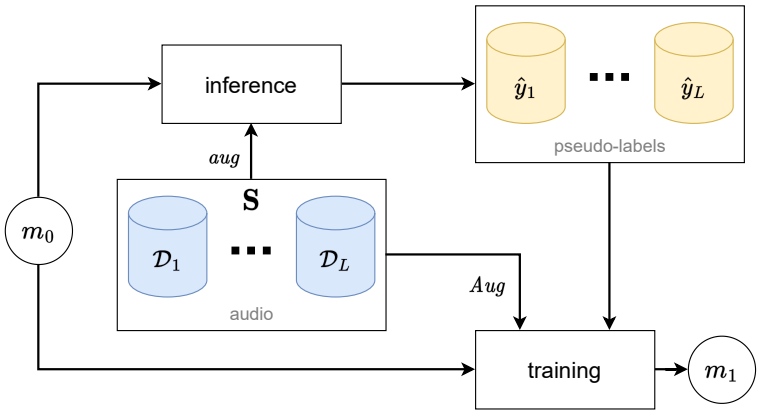


Figure 6.4: Initial non-continual self-supervised training scheme.

As mentioned before, a risk of the model being both teacher and student is divergence, as matching its own predictions may progressively reinforce errors. To limit this, we rely on data augmentation. As shown in Figure 6.4, we calculate pseudo-labels $\hat{y}$ before training with a weak noise-based augmentation *aug* applied to the input audio x:

$$\hat{y}_{kf} = \begin{cases} 1 & \text{if } m_0(aug(x))_{kf} \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

where k denotes speakers and f denotes frames. Our hypothesis is that generating $\hat{y}$ from a weak perturbation of x may help to prevent divergence by providing a slightly distorted view of each input, acting as a form of regularization. Moreover, as depicted in Figure 6.4, we also use $Aug(x)$ as inputs during training to improve robustness to strong perturbations, which we define as the addition of both background noise and reverberation.

Finally, in order to further discourage divergence and overfitting, we rely on the well-known technique of early stopping based on a *stopping criterion*. As shown in Figure 6.5, while the performance on the training set tends to keep improving (due to the nature of gradient descent), validation performance usually peaks after a certain number of epochs, only to reach a plateau or even decrease with further training, which is typically considered a sign of overfitting. In this context, early stopping is designed to automatically halt training when validation performance does not improve after a certain number of epochs (called the *patience* of the algorithm).

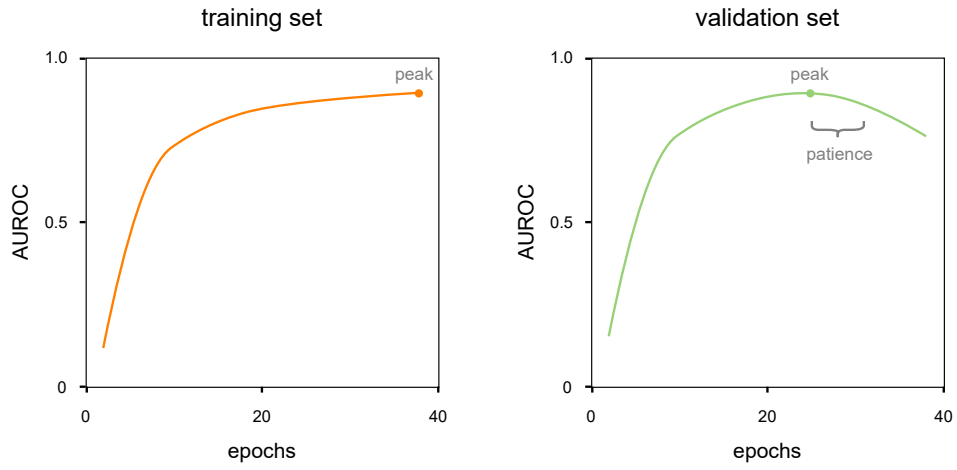


Figure 6.5: Early stopping based on the AUROC metric.

In our case, the stopping criterion is based on a performance metric known as the area under the receiver operating characteristic curve (AUROC for short) (Bradley, 1997). Similarly to the detection error trade-off (DET) curve (see Section 3.3.4), the ROC curve is determined by relating false positives and true positives at different decision thresholds θ_j. As shown in the example of Figure 6.6, this is equivalent to applying multiple thresholds $\theta_j \in [0, 1]$ to $m_0(x)$ and comparing the results to the pseudo-labels that act as the ground truth. Notice that, as with any measure based on estimates of the ground truth, this metric may not approximate actual performance correctly if pseudo-labels contain errors.

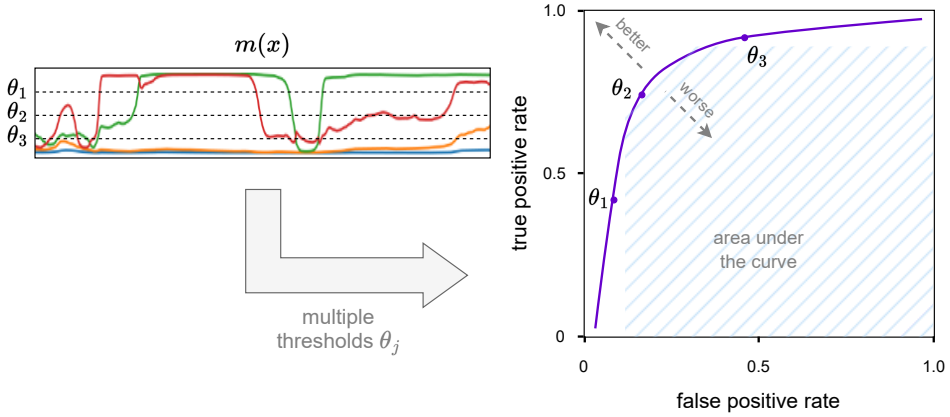


Figure 6.6: The receiver operating characteristic (ROC) curve of a model m is determined by computing the false positive rate and true positive rate of its predictions $m(x)$ at different thresholds θ_j. The area under the ROC curve (AUROC) metric is then defined in the interval $[0, 1]$, and it approaches 1 as the predictions resemble the reference labels.

6.2.3 Experimental protocol

In this section, we describe the way we conduct our experiments, which includes the corpus we use (with the domains it contains), the evaluation metrics, and the implementation details.

Corpus

We experiment on *DIHARD III* (Ryant et al., 2020b), a corpus we have also used in Chapter 5, and that we described in detail in Section 5.4.4. In particular, we are interested in this corpus because it contains conversations from 11 different domains, whose descriptions are shown in Table 6.1.

Domain	Description
Audiobooks	English recordings of speakers reading book passages
Broadcast interview	Radio interviews from the decade of 1970
Clinical	Interviews for the diagnosis of autism in children
Courtroom	Arguments from the 2001 term of the U.S. Supreme Court
CTS	10-minute telephone conversations between two native English speakers
Maptask	A "leader" speaker telling a map route to a "follower" speaker
Meeting	Meeting recordings
Restaurant	Restaurant conversations
Socio field	Sociolinguistic interviews in field conditions
Socio lab	Sociolinguistic interviews in quiet and controlled conditions
Webvideo	English and Mandarin videos collected from the Internet

Table 6.1: Short description of the domains included in DIHARD III as explained by Ryant et al. (2020a).

As shown by the more detailed description presented in Table 6.2, these domains differ greatly in number of speakers and difficulty (of which the diarization error rate is a good proxy). In particular, notice that *webvideo* may not in fact qualify as a domain, as it is a collection of English and Mandarin audio from video sharing platforms. Indeed, its set of shared properties among recordings may be rather small (*e.g.* differences in microphone quality, noise, language, accent, etc.).

Domain splits

In order to craft an experimental protocol for our study, a straightforward approach is to use a set of domains DH_{dev} to determine the best hyper-parameters, and another disjoint set of domains DH_{test} to evaluate the best-performing configuration (where DH stands for "DIHARD"). However, this seems a waste of domains spanning a wide range of different characteristics, and for which we would also want to determine the effectiveness of our method. In particular, specific domains that may be assigned to DH_{dev} by misfortune could in fact represent an interesting success (or failure) case whose study we miss. As a result, in order to perform our evaluation on each of the 11 domains, we cross-validate hyper-parameter optimization making sure not to leak target-domain knowledge neither in model weights nor in hyper-parameters.

To do this, we split the set of domains in DIHARD III *Full* (Ryant et al., 2020b) into sets DH_A and DH_B as shown in Table 6.2. Given the potential differences in domain difficulty, we balance DH_A and DH_B by evening the performance of the VB-HMM baseline (track 2) (Ryant et al., 2020b) between both sets, which we use as a proxy for the difficulty of each domain.

Subset	Domain	Recordings		Duration		Spk / Rec.		Base DER
		dev	test	dev	test	dev	test	
DH_A	Broadcast Interview	12	12	2.1h	2.0h	3.8	3.7	4.6
	Court	12	12	2.1h	2.0h	6.9	7.3	8.9
	Socio Lab	16	12	2.7h	2.0h	2.0	2.0	12.9
	CTS	61	61	10.2h	10.2h	2.0	2.0	20.4
	Meeting	14	11	2.4h	1.9h	5.4	3.9	33.5
	Restaurant	12	12	2.0h	2.1h	7.2	6.4	49.5
DH_B	Audiobooks	12	12	2.0h	2.0h	1.0	1.0	5.2
	Maptask	23	19	2.5h	2.1h	2.0	2.0	11.0
	Socio Field	12	22	2.0h	2.3h	3.5	2.3	18.2
	Clinical	48	51	4.3h	4.4h	2.0	2.0	21.6
	Webvideo	30	35	1.9h	2.1h	4.0	4.1	41.8

Table 6.2: Partitioning of DIHARD III domains. The average number of speakers per recording ("Spk / Rec.") and the DER of the VB-HMM baseline for track 2 (Ryant et al., 2020b) ("Base DER") are evidence of domain differences in difficulty.

Before experimentation, we define a hyper-parameter space consisting of the possible values with which we choose to test our approach. A configuration h extracted from this space contains values for the learning rate, batch size and sound-to-noise

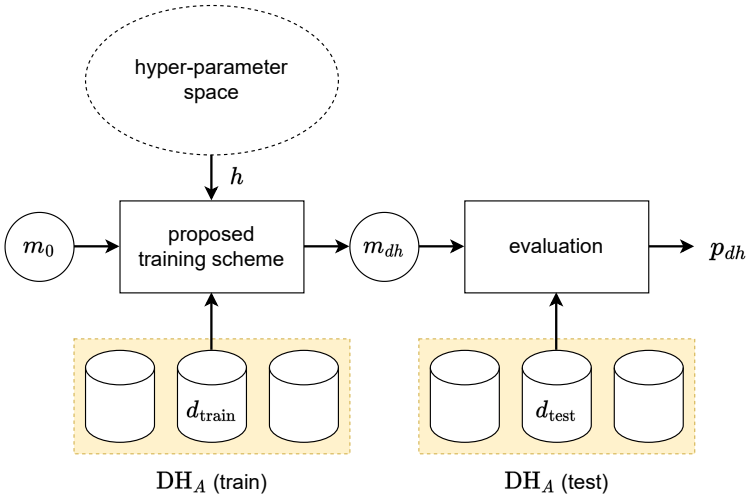


Figure 6.7: Experimental procedure to evaluate performance on a each hyper-parameter configuration h and domain d in DH_A.

ratio ranges for the augmentation functions. As illustrated in Figure 6.7, since our goal is to adapt model m_0 to a *single* domain, for every hyper-parameter configuration h and every domain $d \in \mathrm{DH}_A$, we train m_0 on d_{train} and evaluate the resulting model on d_{test} to determine its performance p_{dh}. Then, we obtain the overall performance of each h by averaging over all domains in DH_A:

$$p_h = \frac{1}{|\mathrm{DH}_A|} \sum_{d \in \mathrm{DH}_A} p_{dh} \tag{6.2}$$

After this procedure, we determine the best-performing configuration h_{best} in the following way (lower is better in our case):

$$h_{\text{best}} = \underset{h}{\operatorname{argmin}}\, p_h \tag{6.3}$$

Lastly, h_{best} is used to train m_0 with our training scheme on each domain $d \in \mathrm{DH}_B$ independently, as depicted in Figure 6.8. For each domain, we use d_{train} for training, and d_{test} for evaluation, resulting in the final per-domain test performance p_d. Crucially, the same process is repeated inverting the roles of DH_A and DH_B to report the performance of the domains in DH_A.

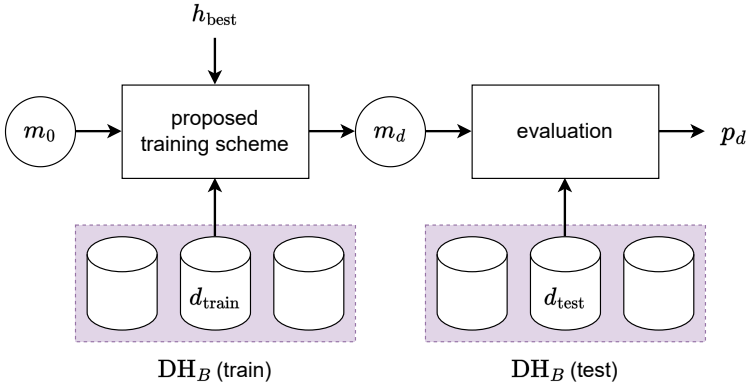


Figure 6.8: Experimental procedure to evaluate the hyper-parameter configuration h_{best} on each domain in DH_B.

Evaluation

Evaluating the quality of a *local* speaker segmentation model with the usual diarization error rate (described in detail in Section 5.2.4) requires an additional speaker tracking algorithm to link speaker identities from different audio chunks,

such as the online speaker diarization system introduced in the previous chapter. Instead of relying on an additional tracking algorithm like a recording-wise clustering, we compute what we call the average chunk-wise diarization error rate, or CDER for short.

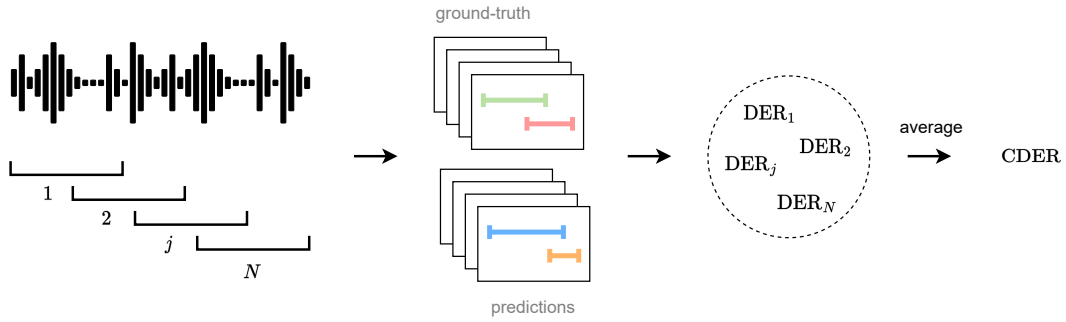


Figure 6.9: The chunk-wise diarization error rate (CDER).

As depicted in Figure 6.9, we split the recording into 5s chunks with a 500ms shift (to reproduce the functioning environment of the system presented in Chapter 5) and we compute the diarization error rate (DER) of each chunk with respect to their reference labels. Then, we obtain the CDER by averaging the resulting DER values. We think this evaluation method can clearly separate the performance of the speaker segmentation model from the one of a clustering algorithm, whose function is to "stitch" consecutive 5s prediction chunks. In other words, the CDER may be correlated to a conversation-level DER in a scenario with perfect clustering, whose evaluation is out of the scope of our study. As in the previous chapter, we compute the DER values with `pyannote.metrics` (Bredin, 2017a) without forgiveness collar and including all overlapping speech.

Implementation details

As discussed in Section 6.2, we use the architecture introduced by Bredin and Laurent (2021) as model m_0 (SincNet (Ravanelli and Bengio, 2018) trainable feature extraction, 4 LSTM (Hochreiter and Schmidhuber, 1997) and 2 fully-connected feed-forward layers) that we pre-train with manually obtained labels on the training set of the AMI meeting corpus (Carletta, 2007) (see Section 5.4.4). This corpus consists of meetings in a strictly controlled environment, which is radically different from most of the domains we can find in DIHARD III. As in the previous chapter, we use the Mix-Headset *Full* partitioning described in Landini et al. (2022). The model we obtain achieves a DER of 17.5% on the AMI corpus test set.

Both strong and weak augmentations *Aug* and *aug* apply random noise, but only *Aug* has a 50% chance of applying a random room impulse response (making

it stronger than *aug*). Noise is sampled from MUSAN (Snyder et al., 2015) (short for "a music, speech and noise corpus") excluding speech, and impulse responses are sampled from *EchoThief* (Warren, 2021) and Traer and McDermott (2016). We use the Adam optimizer (Kingma and Ba, 2015) and fine-tuning is stopped after 3 epochs of no improvement on the development set of the target domain. The hyper-parameter values that we choose to experiment with are shown in Table 6.3.

Hyper-parameter	**Values**
learning rate	$\{10^{-3}, 10^{-4}, 10^{-5}\}$
batch size	$\{16, 32, 64, 128\}$
noise SNR range	{0dB-5dB, 5db-10dB, 10dB-15dB}

Table 6.3: Hyper-parameter search space in our experiments. SNR denotes the sound-to-noise ratio.

6.2.4 Results and discussion

System	**aug?**	**Aug?**	**Average**	**Broadcast Interview**	**Court**	**Socio Lab**	**CTS**	**Meeting**	**Restaurant**	**Audiobooks**	**Maptask**	**Socio Field**	**Webvideo**	**Clinical**
pre-trained	NA	NA	51.4	35.9	22.0	64.8	28.3	86.8	47.5	28.2	40.9	56.8	73.0	80.9
whole1			50.7	33.3	**21.1**	63.5	26.4	88.6	46.8	26.0	41.0	56.2	71.9	82.8
whole2		✓	45.5	**29.4**	21.5	48.5	**23.6**	78.5	**42.9**	**24.8**	**39.7**	36.7	69.4	85.4
whole3	✓	✓	**41.7**	30.5	21.9	**37.7**	25.1	**57.6**	**42.9**	25.6	48.3	**35.8**	**64.6**	**69.1**
whole4	✓		48.2	34.4	24.0	44.8	27.7	82.4	47.0	28.6	52.5	44.6	70.1	74.1
topline	NA	✓	20.5	9.1	7.5	14.6	14.6	38.4	38.7	3.1	12.6	21.9	39.6	25.5

Table 6.4: CDER of the non-continual self-supervised approach on each d_{test}, averaged over 10 runs to limit the effect of randomness. Bold values denote the best model (not counting the supervised topline). Entries named *whole* denote variants of our approach.

The results of this initial set of experiments are summarized in Table 6.4. We use *pre-trained* to denote the performance of the initial pre-trained model, and *topline* to denote the performance of supervised fine-tuning on each domain. Since our self-supervised training scheme trains the model on the whole set of conversations of the target domain at once, we decide to call them *whole*, where the numbers denote variants with different combinations of augmentation strategies.

Our best performing model is *whole3*, which outperforms the pre-trained baseline by a relative 19% on the average CDER (51.4% → 41.7%). This shows that the self-supervised training scheme with pseudo-labels is indeed effective in adapting the pre-trained model to the target domain. In fact, the performance of *whole3* is better than pre-trained across all domains individually, with the sole exception of *maptask*. As expected, there is also a large performance gap with respect to the fully supervised *topline*. Our technique based on pseudo-labels still has considerable room for improvement.

Concerning the augmentation functions, the best average performance is obtained when combining both the pseudo-label augmentation *aug* and the training augmentation *Aug* in *whole3*, which seems to support the hypothesis that they can improve the training signal. However, our results also suggest that *Aug* may have a stronger impact than *aug*. Indeed, when looking at *whole2* (for the effect of *Aug*) and *whole4* (for the effect of *aug*), the average performance obtained by *whole2* is better than *whole4* by almost 3% (45.5% vs 48.2%).

Finally, notice that performance deteriorates with respect to *pre-trained* in some cases, like *whole2* on the *clinical* domain (80.9% → 85.4%), or *whole3* on the *maptask* domain (40.9% → 48.3%). Since pseudo-labels are computed once before training and never updated, we think this may be caused by poor pseudo-labels that fail to provide useful information for the model to exploit, which may contribute to the reinforcement of errors.

6.3 Continual learning extension

Having demonstrated the effectiveness of pseudo-labels for non-continual domain adaptation, the remaining problem we aim to solve is the necessity to collect a large target-domain corpus "a priori". To do this, in this section we propose to add a continual learning extension to the previously described self-supervised training strategy.

6.3.1 Proposed approach

As discussed in previous chapters, continual gradient-based training is prone to catastrophic forgetting (French, 1999; Hadsell et al., 2020), whereby performance on past conversations $\mathcal{D}_i$ deteriorates as new training stages occur. As explained in the introduction, instead of preventing forgetting completely, we propose to focus on maximizing the overall performance on the target domain d, and allow forgetting as long as it does not involve in-domain knowledge. Put simply, we want the model to progressively improve at speaker diarization within the boundaries

of the target domain. Since future conversations in production are restricted to belong to the target domain, this is equivalent to maximizing forward transfer.

Trivially, one could maximize performance on the target domain by training on all past $\mathcal{D}_{j \leq i}$ at each training stage i, but this introduces two problems. First, the cost of training on $\mathcal{D}_i$ grows linearly with the number of conversations, which can in theory be infinite. Second, storing conversations permanently may not be possible, as they are usually considered sensitive data that needs to be guarded from third-party access. To avoid these issues, we train on one conversation $\mathcal{D}_i$ of domain d at a time and discard $\mathcal{D}_i$ immediately after. Hence, our approach could be considered *ephemeral*, as any data from training stages $j < i$ is inaccessible.

As depicted in Figure 6.10, given the initial model m_0 pre-trained on an out-of-domain corpus (in our case the AMI meeting corpus (Carletta, 2007)), we fine-tune model m_{i-1} on a single conversation $\mathcal{D}_i$ of the target domain at a time, resulting in a new model m_i. To do this, we rely on the pseudo-labels produced by m_{i-1}. In other words, we repeat the training strategy introduced in Section 6.2 for each conversation in the sequence, using m_{i-1} as a starting point each time. Moreover, we rely on both augmentation functions *Aug* and *aug* during training and pseudo-label computation, respectively.

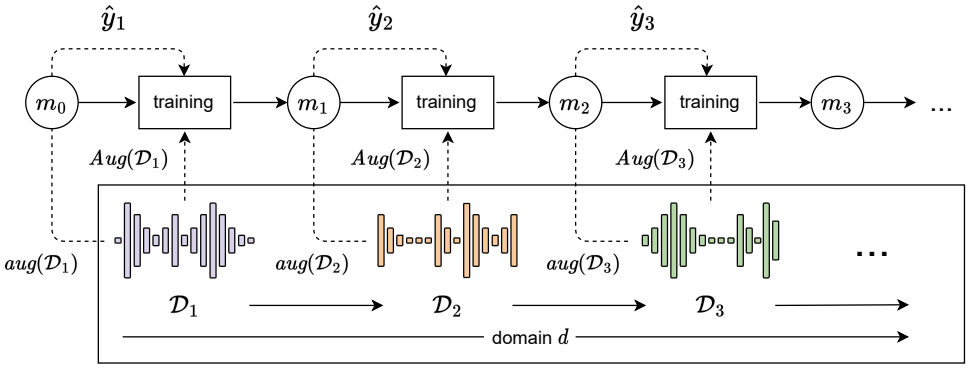


Figure 6.10: Continual training over conversations $\mathcal{D}_i$ of domain d using pseudo-labels $\hat{y}_i$. Model m_0 (pre-trained on an out-of-domain corpus) produces the first pseudo-labels $\hat{y}_1$. From then onward, each model m_{i-1} produces pseudo-labels $\hat{y}_i$ and is then trained only on conversation $\mathcal{D}_i$, resulting in a new model m_i.

Since the model has no access to a validation set in the production phase, we create one for each conversation by extracting approximately 30% of $\mathcal{D}_i$. Specifically, as shown in Figure 6.11, we ensure that both training and development sets for $\mathcal{D}_i$ are spread uniformly throughout the entire recording by interleaving 40s windows for training with 20s windows for validation. We denote the subset containing all training windows as train_i, and the one containing all validation windows as dev_i.

We then calculate the AUROC on dev_i after each training epoch on train_i. When the validation AUROC does not improve for a certain number of epochs, we stop training and wait for the next conversation $\mathcal{D}_{i+1}$. It is worth noting that this procedure trusts the pseudo-labels completely. Hence, it carries the potential risk of confirmation bias.

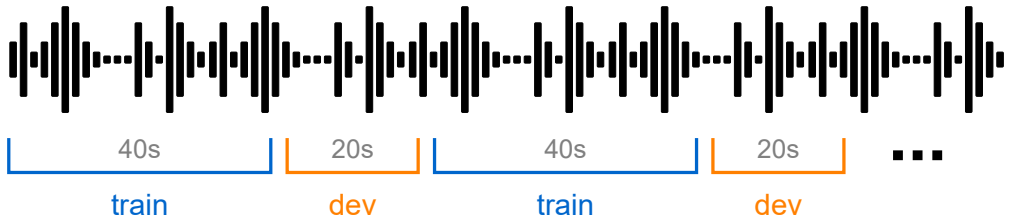


Figure 6.11: Partitioning of a conversation into training and validation sets.

Our work is similar to Tzinis et al. (2022), where the trained speech enhancement model is both teacher and student at the same time and the teacher is regularly updated to improve the quality of the pseudo-labels. However, while each training stage in Tzinis et al. (2022) is run on the entire target-domain corpus $\mathbf{S}$ at once, we train on a single conversation $\mathcal{D}_i$ at a time, with no access to previous conversations $\mathcal{D}_{j<i}$.

Given the effectiveness of the initial non-continual training scheme, we believe that the combination of augmentation and stopping criterion may be capable of favoring forward transfer in this continual learning extension by discouraging overfitting to the current conversation.

6.3.2 Experimental protocol

In our experiments, we use the same domain splitting technique that we described in Section 6.2.3 for cross-validation, with the difference that we train on the target domain d sequentially, using train_i and dev_i. The evaluation CDER is also computed on the same held-out test sets as before. Since these target-domain test conversations could be considered possible future conversations, tracking their performance across the training sequence allows us to measure forward transfer in more detail.

We use a separate Adam optimizer (Kingma and Ba, 2015) for each new conversation, and training on train_i is stopped after 3 epochs of no improvement on dev_i. Training sequences are sorted alphabetically according to existing recording identifiers.

6.3.3 Results and discussion

Table 6.5 summarizes the entire set of experiments. As before, we include the performance of the *pre-trained* baseline and the supervised *topline*, both of which are non-continual. We show the performance of several variants of our approach (denoted *ours*), as well as the fully supervised equivalents (denoted *sup*) that rely on true labels.

System	Labels?	Aug(x)?	Average	Broadcast Interview	Court	Socio Lab	CTS	Meeting	Restaurant	Audiobooks	Maptask	Socio Field	Webvideo	Clinical
pre-trained	NA	NA	51.4	35.9	22.0	64.8	28.3	86.8	47.5	28.2	40.9	56.8	73.0	80.9
ours1	pseudo		56.7	32.2	**21.1**	74.2	25.9	92.3	47.2	27.5	50.1	55.0	99.9	97.9
ours2	pseudo	✓	**42.8**	**30.4**	21.5	**39.3**	**25.3**	**46.8**	44.3	**25.9**	**34.3**	**33.6**	69.2	99.8
ours3	pseudo w/ *aug*	✓	44.3	30.7	21.9	40.8	28.1	48.2	**43.8**	27.6	44.5	39.0	**68.3**	**94.9**
sup1	true		**22.4**	14.6	**8.2**	**16.6**	**15.1**	**38.7**	40.7	**3.2**	**12.5**	24.0	**44.3**	**28.6**
sup2	true	✓	**22.4**	**9.4**	8.7	17.2	15.7	41.3	**40.6**	4.0	13.1	**22.8**	44.6	29.6
topline	true	✓	20.5	9.1	7.5	14.6	14.6	38.4	38.7	3.1	12.6	21.9	39.6	25.5

Table 6.5: CDER of continual models on each d_{test} at the end of the training sequence, averaged over 10 runs to limit the effect of randomness. Bold values denote the best model per domain and type of supervision (not counting the supervised topline).

Continual self-supervision

Our best model *ours2* outperforms *pre-trained* across all domains with a relative improvement of 17% on the average CDER (51.4% → 42.8%), except on *clinical* where the model diverges (80.9% → 99.8%). Surprisingly, *ours2* closely follows our best non-continual system *whole3* on average, with only a 1.1% difference (42.8% vs 41.7%), and even outperforms it in some domains like *meeting* (46.8% vs 57.6%), *maptask* (34.3% vs 48.3%) and *socio field* (33.6% vs 35.8%). This suggests that the quality of pseudo-labels may be improving in these domains as new conversations appear, while pseudo-labels in *whole* systems cannot improve because the teacher model is frozen. This seems to support the hypothesis that the model progressively adapts to the target domain.

Finally, note that supervised continual training (*sup1* and *sup2*) also performs well, rivaling the non-continual supervised *topline* with a CDER absolute difference of 1.9% (22.4% vs 20.5%).

Forward transfer

The results from Table 6.5 only measure performance at the end of the conversation sequence. In Chapter 4, we have seen that this may not fully explain how knowledge is being transferred during continual training.

Figure 6.12 shows the d_{test} CDER after training on each conversation $\mathcal{D}_i$ for each domain. Model *ours2* seems to improve with new conversations in 6 of the 11 domains: *broadcast interview*, *court*, *socio lab*, *meeting*, *maptask* and *socio field*. However, it often reaches a plateau rather early (and still far behind supervised models). Although forward transfer seems to be at play in these domains, this behavior suggests that keeping the same training strategy throughout the entire sequence might be sub-optimal.

In some domains like *conversational telephone speech (CTS)*, *restaurant* and *audiobooks*, the model even starts to slowly diverge after having reached its peak performance. As a result, it might be useful to anticipate this "hinge" moment in continual adaptation to change the training strategy. In particular, similarly to how a plateau may be caused by the *stagnation* of pseudo-label quality, divergence might start to occur when there is a *drop* in pseudo-label quality. In this context, updating the teacher model less frequently (*e.g.* by using m_{i-5} instead of m_{i-1} to generate pseudo-labels), could be a promising alternative to avoid snowballing into a rapid degradation of the training signal.

At the same time, our proposed approach diverges from the very beginning in the *clinical* domain, which could be caused by unreliable pseudo-labels from the initial model m_0, while performance across conversations is extremely unstable for all models in *webvideo*. As discussed before, we believe this instability may be caused by *webvideo*'s loose definition as a domain, suggesting that a well-defined set of shared characteristics may be key to benefit from forward transfer across conversations. Interestingly, this is also supported by our experiments on sequence labeling from Chapter 4, where high levels of forward transfer were observed when sharing a large set of characteristics across languages (*e.g.* task, domain, people names, city names, etc.).

Lastly, note that supervised models show a substantial error increase in a handful of conversations of *restaurant*, which could be caused by ambiguous or inaccurate annotations. Since manual annotations are never used in our training scheme, it may be an interesting alternative to supervised training when annotations are unreliable.

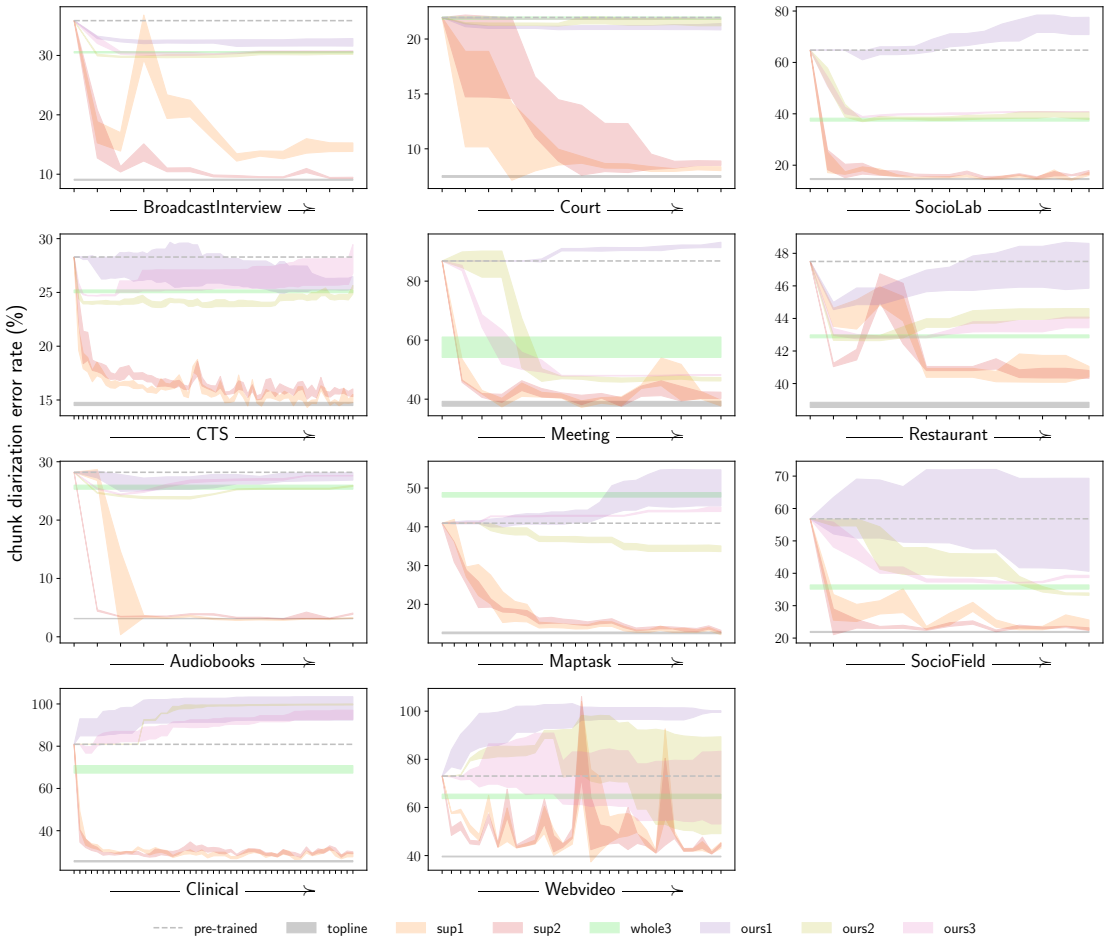


Figure 6.12: CDER on d_{test} as a function of training conversations $\mathcal{D}_i$ appearing sequentially. Curves follow the average and standard deviation across 10 runs. Each system is referenced with its identifier from Table 6.4 and Table 6.5.

A closer look at forward and backward transfer

Despite single-conversation performance not being particularly relevant to our domain adaptation problem, it remains interesting to look at their fluctuation to better understand how knowledge is transferred across the training sequence. Furthermore, this can provide valuable insights for future research directions.

In order to investigate this, we measure the CDER on conversations $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_{L-1}$ and $\mathcal{D}_L$ (*i.e.* the first two and last two conversations) in two contrasting scenarios: successful adaptation (*meeting*) and divergence (*clinical*). Taking the example of the performance matrix presented in Chapter 2 and Chapter 4, this corresponds to the two first and two last rows of P, as shown in Figure 6.13.

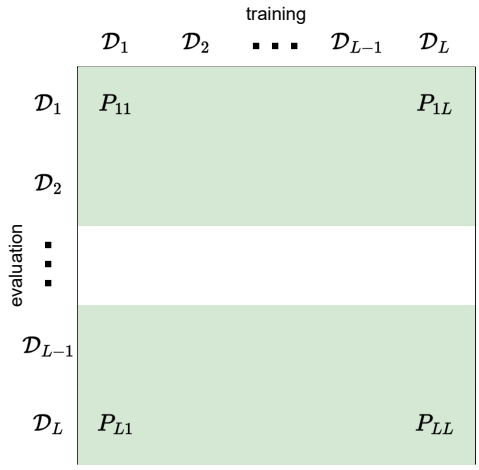


Figure 6.13: Rows of the performance matrix P (highlighted in green) that we selected to investigate single-conversation performance.

The results are shown in Figure 6.14, where we append the performance of *pre-trained* at the beginning of the sequence for reference. First, we observe that performance across conversations differs substantially, suggesting a certain variability in difficulty even in conversations from the same domain. Despite this, forgetting seems to be limited, as sharp error increases in a conversation after training on it are rare when there is no divergence.

Second, since true labels are never seen by *ours2* and *ours3*, it is interesting that performance tends to improve with new conversations. Overall, although both supervised and self-supervised models seem to benefit from transfer in both directions, supervised variants remain the clear winner (as expected), especially in *clinical*. Although not as strong, we believe that the transfer observed in our self-supervised models may progressively improve pseudo-labels as well as model quality estimation for the stopping criterion. It may also explain performance fluctuations in *webvideo*, as very dissimilar conversations might limit transfer.

The role of augmentation

Our results show that augmentation *Aug* (present in *ours2* and *ours3* but not in *ours1*) is key in achieving good self-supervised performance, although not so much in supervised systems, where both variants obtain equal performance on average. The example of *maptask* in Figure 6.12 is particularly interesting, as *Aug* makes the difference between learning and diverging.

On the other hand, *aug* (present only in *ours3*, *whole3* and *whole4*) seems to be more useful in *whole* systems than in continual training. Nevertheless, we believe

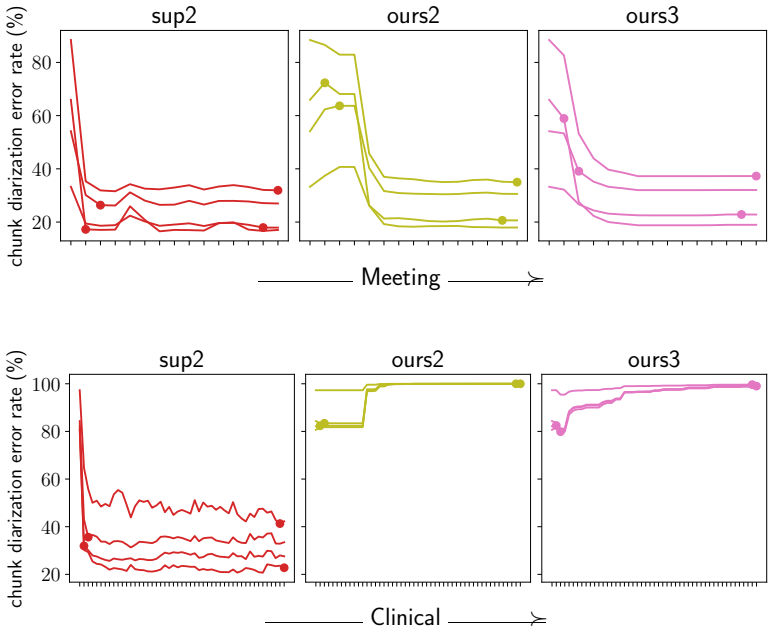


Figure 6.14: CDER on conversations $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_{L-1}$ and $\mathcal{D}_L$. Each curve represents the CDER (averaged over 10 runs) of a single conversation computed *immediately after* training on each conversation of the entire sequence. A dot denotes the position of the target conversation within the sequence. Note that curves are shifted right by one step to accommodate for the initial performance, corresponding to the *pre-trained* baseline.

that *aug* may prevent reinforcing errors at the beginning of continual training when pseudo-label quality is low because of domain mismatch, although failing to prevent divergence in the long term. Figure 6.14 is a good example of this, as *ours3* is better than *ours2* in the beginning of the training sequence for both *meeting* and *clinical*. As a result, it may be more interesting to apply *aug* only during the first conversations of the sequence to obtain the best qualities from both variants.

6.4 Conclusion

In this chapter, we have proposed a training scheme for domain adaptation in speaker segmentation. We trained a model on conversations from a target domain sequentially and in a fully self-supervised way, removing the need for annotating data and for storing personally identifiable data permanently. After a variety of experiments on the 11 DIHARD III (Ryant et al., 2020b) domains covering a wide

range of scenarios, we reach the following conclusions.

First, our training scheme achieves an average 17% relative improvement over a pre-trained baseline, even rivaling a non-continual self-supervised topline, which shows that it is indeed possible for the segmentation model to self-adapt to a target domain continually. In fact, our approach can run locally and autonomously in the background with little to no human involvement (*e.g.* in a home voice assistant). Since speaker segmentation constitutes one of the key modules in the online speaker diarization system that we introduced in Chapter 5, we believe that the combination of our continual training scheme with this online system is a very promising future research direction that could boost the adaptability of speaker recognition systems to the environments of the production phase.

Second, even though our approach is effective on a variety of domains, our results also highlight several areas for improvement. In particular, clearly identifying convergence and divergence cases is extremely important for the reliable deployment of our method.

Finally, even in convergence scenarios, the gap between self-supervised and supervised models remains large. We believe that it is key for future research to reduce this gap in order to raise the attractiveness of self-supervised approaches and eliminate the tedious, error-prone and costly manual annotation process. A possible way to do this could be to pre-train the initial segmentation model for few-shot learning to new domains, for example by leveraging meta-learning techniques.

Chapter 7

Conclusion

"Every move you make opens a whole new world of possibilities..."
— Matt Haig, *The Midnight Library*

In this chapter, we provide a summary and overview of the contributions presented in this manuscript. In Section 7.1, we summarize the research conducted in the different chapters throughout the thesis. In Section 7.2, we discuss our open source contributions, and in particular *diart*: an online speaker diarization library and toolkit written in Python that we developed for the study presented in Chapter 5. In Section 7.3 we list our contributions, and in Section 7.4 we discuss possible directions that we consider interesting and promising for future research.

7.1 Summary

In this thesis, we have investigated and discussed different ways of addressing the continual learning problem in written and spoken language applications. In particular, our primary focus was on the production phase of a machine learning model, which is characterized by the presence of unlabeled data arriving sequentially as time passes. As we have discussed previously, we believe this problem requires autonomous and flexible learning systems that are capable to adapt to their production environments without human supervision, which is extremely labor-intensive and resource-consuming.

In Chapter 2, we have discussed background work on three key research fields related to this thesis: neural representations, transfer learning and continual learning. On the one hand, we found transfer learning relevant to leverage knowledge

acquired by the model during its development phase, as well as to gradually improve and refine that knowledge in production. On the other hand, we saw that the catastrophic forgetting problem in continual learning has been linked to drastic model parameter changes in gradient-based learning (Hadsell et al., 2020). Consequently, we have identified task-specific representations as a promising research direction, as they do not rely on this type of learning for refinement.

In Chapter 3, we began our journey by investigating ways to learn task-specific representations in both written and spoken language tasks. In a comparison of metric learning loss functions, we learned that these techniques may be better suited for open-set tasks, as they benefit more from the intra-class compactness and inter-class separability encouraged by metric learning. These observations forked our research in two directions that we have explored in the subsequent chapters.

Given the ineffectiveness of task-specific representations on closed-set tasks, in Chapter 4 we took a step back from our initial hypothesis to study the continual adaptation of a contextual word embedding model to new languages, for example in the context of a dialogue system that is progressively deployed in different countries. Surprisingly, we observed high levels of forward transfer, which lead to the discovery of fast recovery capabilities.

In Chapter 5, we explored the possibility of leveraging the task-specific representations obtained in Chapter 3 for the open-set task of online speaker diarization, targeting the adaptation to a live conversation. We believe this is a relevant but often ignored use case, as most diarization systems require access to the entire conversation at once. Hence, we have proposed a modular system that progressively refines internal speaker representations with incremental clustering, and we have demonstrated its ability to continually improve without supervision.

Finally, in Chapter 6 we attempted to solve one of the main shortcomings of the system introduced in Chapter 5. We have proposed a training scheme to continually adapt the speaker segmentation model to new unlabeled conversations from a target domain that appear sequentially. Since conversations are typically considered sensitive data, we also make sure not to store recordings permanently. For example, a home voice assistant could rely on our approach to progressively improve live speaker recognition performance after each conversation with the involved users. We show that our training scheme outperforms a pre-trained baseline, and is even on par with non-continual self-supervised models.

Overall, we believe our work has brought us a step closer to the design of effective autonomous and self-learning systems for language-related applications, capable of exploiting the unlabeled data available in their production environments. On the

one hand, our hypothesis of leveraging representations that are tailored to a specific task has proven to be effective in the context of online speaker diarization (Chapter 5). On the other hand, both our supervised (Chapter 4) and self-supervised (Chapter 6) studies on continual fine-tuning have shown that it is possible to benefit from high forward transfer under the right conditions, even in the face of forgetting.

However, although a step in the right direction, our work has barely scratched the surface of post-deployment continual adaptation, as there is still a wide array of problems that we have not addressed and questions we have not answered. In particular, many complex tasks we have left out from our study, like question answering, information retrieval, speech recognition and translation, could all benefit from continual adaptation after model deployment. Furthermore, a key limitation in the adoption of continual adaptation techniques is the performance gap with respect to well-established yet non-continual solutions, which is extremely difficult to address given the limited context that is available to continual systems.

At the same time, two key questions arising from this thesis remain unanswered. First, although we open a discussion for continual learning through representation refinement, more research is needed to understand how well-established forgetting and transfer definitions translate to this scenario, and whether the representation models themselves can adapt to the production environment as well. Lastly, the underlying cause of the high forward transfer we observed remains unclear, which we believe requires further investigation on both theoretical and practical grounds.

7.2 Reproducible research

One of the contributions of this thesis is the source code associated to each of our studies, which allows anyone to reproduce the results discussed throughout the manuscript. In the following list, we present the links to the public and open source GitHub repositories corresponding to each chapter.

- **Chapter 3 (speaker verification and misogyny categorization)**
  `github.com/juanmc2005/SpeakerEmbeddingLossComparison`
  `github.com/juanmc2005/MetricAMI`
- **Chapter 4 (slot filling and named entity recognition)**
  `github.com/juanmc2005/ContinualNLU`
- **Chapter 5 (online speaker diarization)**
  `github.com/juanmc2005/StreamingSpeakerDiarization`
- **Chapter 6 (speaker segmentation)**

github.com/juanmc2005/CSDA

An additional work of this thesis has been the further development of the source code corresponding to Chapter 5. Indeed, we believe it could become a useful tool for machine learning practitioners in the field of online speaker diarization. In the following section, we discuss this specific contribution in more detail.

Diart: a real-time speaker diarization library

Speaker diarization in real-time holds the potential to accelerate and cement the adoption of speaker recognition technology in our everyday lives. However, although current offline systems achieve outstanding performance in pre-recorded conversations, additional problems of online diarization, like limited context and low latency, require flexible and efficient solutions enabling both research and production-ready applications.

A first version of the source code for Chapter 5 included a demonstration script capable of streaming audio from a local microphone and drawing the predictions of our proposed system in real time. Given its initial popularity and the feedback of the GitHub community, we decided to develop this project further and transform it into a toolkit for online speaker diarization. This source code has also been the subject of the following scientific publication, which is currently under review:

Juan M. Coria, Hervé Bredin, Sahar Ghannay, Sophie Rosset, Khaled Zaouk, Ingo Fruend, Bertrand Higy, Amit Kesari, and Yagna Thakkar. Diart: A Python Library for Real-Time Speaker Diarization. Submitted to *The Journal of Open Source Software*, 2022.

Diart is a Python library for real-time speaker diarization that leverages data structures and pre-trained models available in `pyannote.audio` (Bredin et al., 2020) to implement production-ready real-time inference on a variety of audio streams, like local and remote audio/video files, microphones, and even WebSockets. The following code extract demonstrates the simplicity of running live inference over a microphone with diart, while plotting predictions in real time as they are produced by the system.

```
from diart import OnlineSpeakerDiarization
from diart.sources import MicrophoneAudioSource
from diart.inference import RealTimeInference

system = OnlineSpeakerDiarization()
mic = MicrophoneAudioSource(system.config.sample_rate)
inference = RealTimeInference(system, mic, do_plot=True)
prediction = inference()
```

The library was also designed to facilitate research by providing fast batched inference and hyper-parameter tuning thanks to and in full compatibility with Optuna (Akiba et al., 2019). As shown in the code extract below, the `Optimizer` class allows users to tune hyper-parameters, while `Benchmark` allows to evaluate a system. Both these classes can be applied on entire corpora, and achieve significantly faster run times than `RealTimeInference` by applying models in mini-batches and simulating a stream.

```
from diart import OnlineSpeakerDiarization
from diart.inference import Benchmark
from diart.optim import Optimizer

# Data directories
audio = "/audio/dir"
labels = "/ground-truth/dir"
# Optimizer output directory
output = "/output/dir"

# Optimize hyper-parameters
optimizer = Optimizer(audio, labels, output)
optimizer(num_iter=100)

# Evaluate the system
system = OnlineSpeakerDiarization(optimizer.best_config)
benchmark = Benchmark(audio, labels)
benchmark(system)
```

Diart also follows an object-oriented design fully capable of extension and customization. Streaming is powered internally by ReactiveX extensions (RxPY), but available "blocks" allow users to mix and match different operations with any streaming library they choose. For instance, the following code extract shows how to combine diart blocks with ReactiveX to continually extract overlap-aware speaker embeddings (see Section 5.3.2).

```
import rx.operators as ops
import diart.operators as dops
from diart.sources import MicrophoneAudioSource
from diart.blocks import (
    SpeakerSegmentation as Segmentation,
    OverlapAwareSpeakerEmbedding as Embedding
)

seg = Segmentation.from_pyannote("pyannote/segmentation")
emb = Embedding.from_pyannote("pyannote/embedding")
sample_rate = segmentation.model.get_sample_rate()
```

```
mic = MicrophoneAudioSource(sample_rate)

stream = mic.stream.pipe(
    # Reformat stream to 5s duration and 500ms shift
    dops.rearrange_audio_stream(sample_rate),
    ops.map(lambda wav: (wav, seg(wav))),
    ops.starmap(emb)
).subscribe(on_next=lambda e: print(e))

mic.read()
```

A prototyping tool with a command-line interface is also provided to quickly run inference, evaluation, profiling and optimization without writing any Python code.

Finally, measurements of the computation time show that it takes 165ms to provide the output of a single state of the rolling buffer on a CPU Intel Cascade Lake 6248 (20 cores at 2.5Ghz), while the number decreases more than three-fold on a GPU Nvidia Tesla V100 SXM2, achieving a processing time of 50ms. Considering updates of the rolling buffer every 500ms (which can also be customized), this means that our implementation is well-suited for real-time applications, as a prediction can be obtained before receiving the next buffer state.

All in all, we hope diart's flexibility, efficiency and customization will allow for exciting new research and applications in online speaker diarization.

7.3 Contributions

Throughout the work of this thesis, we have made the following contributions:

A systematic comparison of metric learning loss functions. In Chapter 3, we performed a comparison of loss functions for metric learning that was lacking in the literature at the time. In speaker verification, we have shown that additive angular margin loss (Deng et al., 2019) is superior to the other loss functions compared in the study. In misogyny categorization, we have shown that metric learning loss functions do not perform better than the standard cross entropy loss.

A settlement on the usefulness of metric learning. In Chapter 3, we found that metric learning may be useful in the context of open-set tasks, but ineffective on closed-set tasks. This seems to provide an explanation of why research on metric learning techniques has sometimes been contradicting in the past. Indeed, some studies on closed-set tasks have claimed that their improvement is at best marginal (Musgrave et al., 2020), while other studies on open-set tasks have claimed a significant improvement (Srivastava et al., 2020). By comparing these

techniques on both types of problems, we have found a lacking aspect in previous comparisons that may explain why this is the case. However, confirming this hypothesis requires a larger study across a wide variety of tasks from different domains and modalities.

A fully end-to-end speaker embedding model. Previous work on speaker verification relies on costly feature extraction algorithms like MFCC and data-hungry post-processing modules like PLDA (Ioffe, 2006) for speaker embedding. In Chapter 3, we trained a simpler fully end-to-end model relying on SincNet trainable features (Ravanelli and Bengio, 2018) and the additive angular margin loss (Deng et al., 2019) that achieved competitive performance. Furthermore, we have shared this model with the speaker recognition community to accelerate future research and power practical applications.

A state-of-the-art misogyny categorization model. In Chapter 3, we trained a model for the task of misogyny categorization on tweets that outperformed the state-of-the-art model to date by an absolute 4.4% difference in F1 score. However, performance still remains low on this challenging task.

A better understanding of forward transfer. In Chapter 4, we saw that a sequence labeling model that is continually trained on the adaptation axis of languages suffers from forgetting (*i.e.* performance loss of previously learned languages). Although this was expected, we were also surprised to discover that the model was able to leverage previous knowledge in the form of forward transfer (*i.e.* performance gain of new languages with respect to monolingual). Additional experiments provided evidence that model parameters were shifting towards a better multilingual initialization, while moving away from previous-language optima. Our experiments from Chapter 6 on the domain axis in speaker diarization also seemed to benefit from a similar phenomenon. This is an interesting property that could be exploited in both future research and practical applications.

A better understanding of the localization of progressively acquired knowledge. In the same vein, our experiments from Chapter 4 also provided evidence of where the knowledge facilitating forward transfer is stored. We found that most of this knowledge was stored in the Transformer-based encoder BERT (Devlin et al., 2019), and not in the appended word classifier. In this context, we have shown that the common strategy of freezing general-purpose model parameters to avoid forgetting can be highly counterproductive, as it may completely erase the possibility of forward transfer, and hence of progressive improvement.

Fast recovery capabilities. In Chapter 4, we have shown that it is possible to leverage high forward transfer for fast recovery capabilities, allowing the model to fully restore and even improve lost performance on previous languages with just a

few training epochs. However, it remains unclear whether a cost-effective solution exists to avoid forgetting with this technique.

A state-of-the-art online speaker diarization system. In Chapter 5, we have proposed a modular system for online speaker diarization consisting of three steps: speaker segmentation, speaker embedding and incremental clustering. We have shown that our approach outperforms the state-of-the-art with a lower memory consumption as well as an adjustable latency, which can even reach half the latency of the previous best system. Furthermore, our system adapts continually, progressively bridging the gap to offline performance as time passes. Nevertheless, there is still a large gap between offline and online performance, specifically in speaker confusion. This shows that our incremental clustering algorithm has considerable room for improvement.

A Python toolkit for online speaker diarization. As discussed in Section 7.2, we have released an open source implementation of our online speaker diarization system from Chapter 5 in the form of a Python library and toolkit, that works on any live conversation in real time. This allows anyone to use, optimize, evaluate and even customize our system with little effort.

A training scheme for continual self-supervised domain adaptation in speaker diarization. In Chapter 6, we have proposed a training scheme combining continual learning and pseudo-labels to progressively adapt the speaker segmentation model (used in Chapter 5) to a new domain one conversation at a time. We found our approach to be effective, outperforming a pre-trained baseline by a relative 17% without storing sensitive user data. However, our method is still behind its fully supervised equivalent.

7.4 Future directions

As mentioned in Section 7.1, we believe two important yet unaddressed research directions arise from our work: the improvement of continual learning with task-specific representations, and the better understanding of naturally occurring forward transfer. In this section, we list what we believe to be important future research topics to answer key questions from each of these directions.

7.4.1 Continual learning with task-specific representations

Meaningful metrics. In Chapter 5, we saw that forgetting could be more accurately defined as a centroid divergence problem in the context of incremental clustering. Moreover, we believe it is also not fully applicable to the task of online

diarization itself. Indeed, there is a low risk of forgetting speakers, as they tend to speak regularly throughout the target conversation. In fact, this may act as a form of replay (see Section 2.4.2). We believe that task-specific definitions of both forgetting and transfer metrics are crucial to understand and address the difficulties of continual adaptation in different types of applications.

Continual refinement of the embedding model. Although we have proposed a way to progressively refine speaker centroids within a fixed embedding space, it should also be possible to refine the embedding model in the same way as the speaker segmentation model in Chapter 6. Indeed, continually improving intra-class compactness and inter-class separability for the target domain holds the potential to substantially decrease speaker confusion. A technique boosting these properties while ensuring backward compatibility (so that previous centroids remain meaningful) could provide significant gains to incremental clustering.

Joint speaker segmentation and embedding. The system we have proposed in Chapter 5 is based on two fundamental pre-trained building blocks, which are the speaker segmentation and speaker embedding models. However, these models are pre-trained separately on different data from potentially different domains, and are not allowed to learn from each other. We believe it should be possible to train a single model to perform these two tasks at once, which could allow joint transfer between them and hence achieve better performance. Furthermore, it could be free to learn the transformation function we presented in Equation 5.4 to find improved weights for embedding extraction. Nevertheless, the difficulty of this study lies in finding the right training data and multi-task loss function to favor joint transfer and discourage interference.

A combination of our proposed approaches. We believe that the continual self-supervised training scheme from Chapter 6 could provide a substantial boost in the continual adaptation capabilities of the online speaker diarization system proposed in Chapter 5. For example, the speaker segmentation model could be fine-tuned in the background using the past audio and predictions from the target conversation, while a previous snapshot of the segmentation model is being used for real-time inference.

Incremental clustering constraints. The incremental clustering algorithm we have proposed in Chapter 5 is not without its flaws. In particular, it requires several hyper-parameters to be tuned beforehand, and it is based on rigid and hand-crafted rules. Alleviating some of these constraints could give more flexibility to the system, for example by determining the value of δ_{new} automatically based on the distribution of speaker embedding distances computed in the past.

7.4.2 Better understanding of forward transfer

The causes of high forward transfer. In Chapter 4 and Chapter 6, we saw that forward transfer can appear without any specific encouragement from our part. Although we hypothesized that this is due to shared properties of the data in each training stage, we have not answered this question definitely. We think that further research is needed in this area to fully understand the causes underlying this phenomenon, as well as its applicability in other situations.

Fast model recovery. In Chapter 4, we discovered that the model could recover its lost performance (due to forgetting) in a matter of a few training epochs. We hypothesize that these capabilities can be exploited in a variety of scenarios. For instance, it could be used to efficiently bootstrap the training process of a multilingual model at the beginning of a long and tedious data collection and annotation process. Additionally, it could even provide a way to avoid catastrophic forgetting by interleaving "recovery cycles" with training stages. However, it remains unclear whether doing so in a cost-effective manner is possible.

Localization of transferred knowledge. In Chapter 4, we learned that the knowledge allowing high forward transfer was mostly stored in the contextual word embedding model, and not in the task-specific classifier. However, given the depth of the model, it would be useful to determine the relevance of each layer more precisely. We hypothesize that certain layers (*e.g.* the lower layers) could contain more general-purpose knowledge than others, which could allow for the development of new constraint-based continual learning methods (see Section 2.4.3) that do not interfere with forward transfer.

Bibliography

Abdi, H. (2007). The Kendall Rank Correlation Coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, pages 508–510.

Abraham, W. C. and Robins, A. (2005). Memory Retention – The Synaptic Stability versus Plasticity Dilemma. *Trends in Neurosciences*, 28(2):73–78.

Ahluwalia, R., Soni, H., Callow, E., Nascimento, A., and Cock, M. D. (2018). Detecting Hate Speech Against Women in English Tweets. In Caselli, T., Novielli, N., Patti, V., and Rosso, P., editors, *EVALITA Evaluation of NLP and Speech Tools for Italian*, pages 194–199. Accademia University Press.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A Next-Generation Hyperparameter Optimization Framework. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory Aware Synapses: Learning What (not) to Forget. In *European Conference on Computer Vision (ECCV)*.

Anguera, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., and Vinyals, O. (2012). Speaker Diarization: A Review of Recent Research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370.

Arkhipov, M., Trofimova, M., Kuratov, Y., and Sorokin, A. (2019). Tuning Multilingual Transformers for Language-Specific Named Entity Recognition. In *Workshop on Balto-Slavic Natural Language Processing*, pages 89–93.

Arora, G., Rahimi, A., and Baldwin, T. (2019). Does an LSTM Forget More Than a CNN? An Empirical Study of Catastrophic Forgetting in NLP. In *Annual Workshop of the Australasian Language Technology Association*, pages 77–86, Sydney, Australia. Australasian Language Technology Association.

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., and Cheney, N. (2020). Learning to Continually Learn. In *European Conference on Artificial Intelligence (ECAI)*, pages 992–1001. IOS Press.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

Bradley, A. P. (1997). The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition*, 30(7):1145–1159.

Bredin, H. (2017a). pyannote.metrics: A Toolkit for Reproducible Evaluation, Diagnostic, and Error Analysis of Speaker Diarization Systems. In *Interspeech*, pages 3587–3591.

Bredin, H. (2017b). TristouNet: Triplet Loss for Speaker Turn Embedding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5430–5434, New Orleans, LA. IEEE.

Bredin, H. and Laurent, A. (2021). End-to-End Speaker Segmentation for Overlap-Aware Resegmentation. In *Interspeech*.

Bredin, H., Yin, R., Coria, J. M., Gelly, G., Korshunov, P., Lavechin, M., Fustes, D., Titeux, H., Bouaziz, W., and Gill, M.-P. (2020). pyannote.audio: Neural Building Blocks for Speaker Diarization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Bullock, L., Bredin, H., and Garcia-Perera, L. P. (2020). Overlap-Aware Diarization: Resegmentation Using Neural End-to-End Overlapped Speech Detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Carletta, J. (2007). Unleashing the Killer Corpus: Experiences in Creating the Multi-Everything AMI Meeting Corpus. *Language Resources and Evaluation*, 41(2):181–190.

Caselli, T., Novielli, N., Patti, V., and Rosso, P. (2018). Tweetaneuse@ AMI EVALITA2018: Character-Based Models for the Automatic Misogyny Identification Task. In *Proceedings of the Final Workshop*, volume 12, page 13.

Chagas Nunes, J. A., Macêdo, D., and Zanchettin, C. (2019). Additive Margin SincNet for Speaker Recognition. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–5.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations. In III, H. D. and Singh, A., editors, *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Chieu, H. L. and Ng, H. T. (2002). Named Entity Recognition: A Maximum Entropy Approach Using Global Information. In *International Conference on Computational Linguistics (COLING)*.

Chung, J. S., Huh, J., Nagrani, A., Afouras, T., and Zisserman, A. (2020). Spot the Conversation: Speaker Diarisation in the Wild. In *Interspeech*, pages 299–303.

Chung, J. S., Nagrani, A., and Zisserman, A. (2018). VoxCeleb2: Deep Speaker Recognition. In *Interspeech*, pages 1086–1090.

Chung, Y.-A. and Glass, J. (2020). Generative Pre-Training for Speech with Autoregressive Predictive Coding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3497–3501.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2020). Unsupervised Cross-lingual Representation Learning at Scale. In *Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Deng, J., Guo, J., and Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694.

Desplanques, B., Thienpondt, J., and Demuynck, K. (2020). ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification. In Meng, H., Xu, B., and Zheng, T. F., editors, *Interspeech*, pages 3830–3834. ISCA.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Diez, M., Landini, F., Burget, L., Rohdin, J., Silnova, A., Žmolíková, K., Novotný, O., Veselý, K., Glembek, O., Plchot, O., Mošner, L., and Matějka, P. (2018). BUT System for DIHARD Speech Diarization Challenge 2018. In *Interspeech*, pages 2798–2802.

Do, Q., Gaspers, J., Roeding, T., and Bradford, M. (2020). To What Degree Can Language Borders Be Blurred In BERT-Based Multilingual Spoken Language Understanding? In *International Conference on Computational Linguistics (COLING)*, pages 2699–2709, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Draelos, T. J., Miner, N. E., Lamb, C. C., Cox, J. A., Vineyard, C. M., Carlson, K. D., Severa, W. M., James, C. D., and Aimone, J. B. (2017). Neurogenesis Deep Learning: Extending Deep Networks to Accommodate New Classes. In *International Joint Conference on Neural Networks (IJCNN)*, pages 526–533.

Fersini, E., Nozza, D., and Rosso, P. (2018a). Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI). *EVALITA Evaluation of NLP and Speech Tools for Italian*, 12:59.

Fersini, E., Rosso, P., and Anzovino, M. (2018b). Overview of the Task on Automatic Misogyny Identification at IberEval 2018. In *IberEval@ SEPLN*, pages 214–228.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Precup, D. and Teh, Y. W., editors, *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named Entity Recognition through Classifier Combination. In *Conference on Natural Language Learning at HLT-NAACL*, pages 168–171.

French, R. M. (1991). Using Semi-Distributed Representations to Overcome Catastrophic Forgetting in Connectionist Networks. In *Annual Cognitive Science Society Conference*, volume 1, pages 173–178.

French, R. M. (1999). Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.

Fujita, Y., Kanda, N., Horiguchi, S., Nagamatsu, K., and Watanabe, S. (2019). End-to-End Neural Speaker Diarization with Permutation-Free Objectives. In *Interspeech*, pages 4300–4304.

Fujita, Y., Kanda, N., Horiguchi, S., Xue, Y., Nagamatsu, K., and Watanabe, S. (2019). End-to-End Neural Speaker Diarization with Self-Attention. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 296–303.

Garcia-Romero, D., McCree, A., Snyder, D., and Sell, G. (2019). JHU-HLTCOE System for VoxSRC 2019.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (1993). DARPA TIMIT Acoustic-Phonetic Continous Speech Corpus CD-ROM. NIST Speech Disc 1-1.1. *NASA STI/Recon Technical Report n*, 93:27403.

Gelly, G. and Gauvain, J. (2017). Spoken Language Identification Using LSTM-Based Angular Proximity. In *Interspeech*, pages 2566–2570. ISCA.

Gittens, A., Achlioptas, D., and Mahoney, M. W. (2017). Skip-Gram - Zipf + Uniform = Vector Additivity. In *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–76, Vancouver, Canada. Association for Computational Linguistics.

Grossberg, S. (1987). Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, 11(1):23–63.

Haasnoot, E., Khodabakhsh, A., Zeinstra, C., Spreeuwers, L., and Veldhuis, R. (2018). FEERCI: A Package for Fast Non-Parametric Confidence Intervals for Equal Error Rates in Amortized O(m log n). In *International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5.

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality Reduction by Learning an Invariant Mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1735–1742.

Hadsell, R., Rao, D., Rusu, A. A., and Pascanu, R. (2020). Embracing Change: Continual Learning in Deep Neural Networks. *Trends in Cognitive Sciences*, 24:1028–1040.

Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990). The ATIS Spoken Language Systems Pilot Corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Hermans, A., Beyer, L., and Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Horiguchi, S., Fujita, Y., Watanabe, S., Xue, Y., and Nagamatsu, K. (2020). End-to-End Speaker Diarization for an Unknown Number of Speakers with Encoder-Decoder Based Attractors. In *Interspeech*, pages 269–273.

Horiguchi, S., Garcia, P., Fujita, Y., Watanabe, S., and Nagamatsu, K. (2021). End-to-End Speaker Diarization as Post-Processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Ioffe, S. (2006). Probabilistic Linear Discriminant Analysis. In Leonardis, A., Bischof, H., and Pinz, A., editors, *European Conference on Computer Vision (ECCV)*, pages 531–542, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jatnika, D., Bijaksana, M. A., and Suryani, A. A. (2019). Word2Vec Model Analysis for Semantic Similarities in English Words. *Procedia Computer Science*, 157:160–167. International Conference on Computer Science and Computational Intelligence (ICCSCI): Enabling Collaboration to Escalate Impact of Research Results for Society.

Javed, K. and White, M. (2019). Meta-Learning Representations for Continual Learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

K, K., Wang, Z., Mayhew, S., and Roth, D. (2020). Cross-Lingual Ability of Multilingual BERT: An Empirical Study. In *International Conference on Learning Representations (ICLR)*.

Kahn, J., Lee, A., and Hannun, A. (2020). Self-Training for End-to-End Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7084–7088.

Kaya, M. and Bilge, H. c. (2019). Deep Metric Learning: A Survey. *Symmetry*, 11(9).

Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Kinoshita, K., Delcroix, M., and Tawara, N. (2021). Integrating End-to-End Neural and Clustering-Based Diarization: Getting the Best of Both Worlds. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7198–7202.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kolbæk, M., Yu, D., Tan, Z.-H., and Jensen, J. (2017). Multitalker Speech Separation With Utterance-Level Permutation Invariant Training of Deep Recurrent Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(10):1901–1913.

Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Landini, F., Profant, J., Diez, M., and Burget, L. (2022). Bayesian HMM Clustering of x-vector Sequences (VBx) in Speaker Diarization: Theory, Implementation and Analysis on Standard Tasks. *Computer Speech & Language*, 71:101254.

Landini, F., Wang, S., Diez, M., Burget, L., Matějka, P., Žmolíková, K., Mošner, L., Silnova, A., Plchot, O., Novotný, O., Zeinali, H., and Rohdin, J. (2020). BUT System for the Second DIHARD Speech Diarization Challenge. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6529–6533.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521(7553):436–444.

Lee, D.-H. et al. (2013). Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *ICML Workshop on Challenges in Representation Learning*, volume 3, page 896.

Lee, S. (2017). Toward Continual Learning for Conversational Agents. *arXiv preprint arXiv:1712.09943*.

Li, Y., Gao, F., Ou, Z., and Sun, J. (2018). Angular Softmax Loss for End-to-End Speaker Verification. In *International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 190–194.

Li, Z. and Hoiem, D. (2016). Learning Without Forgetting. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *European Conference on Computer Vision (ECCV)*, pages 614–629, Cham. Springer International Publishing.

Lin, Q., Yin, R., Li, M., Bredin, H., and Barras, C. (2019). LSTM Based Similarity Measurement with Spectral Clustering for Speaker Diarization. In *Interspeech*, pages 366–370.

Liu, Y., Li, H., and Wang, X. (2017). Rethinking Feature Discrimination and Polymerization for Large-Scale Recognition. *arXiv preprint arXiv:1710.00870*.

Liu, Z., Winata, G. I., Madotto, A., and Fung, P. (2021). Preserving Cross-Linguality of Pre-trained Models via Continual Learning. In *Workshop on Representation Learning for NLP (RepL4NLP)*, pages 64–71, Online. Association for Computational Linguistics.

Lopez-Paz, D. and Ranzato, M. A. (2017). Gradient Episodic Memory for Continual Learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Madikeri, S., Himawan, I., Motlicek, P., and Ferras, M. (2015). Integrating Online i-vector Extractor with Information Bottleneck Based Speaker Diarization System. In *Interspeech*.

Madotto, A., Lin, Z., Zhou, Z., Moon, S., Crook, P., Liu, B., Yu, Z., Cho, E., Fung, P., and Wang, Z. (2021). Continual Learning in Task-Oriented Dialogue Systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7452–7467, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Malmasi, S., Fang, A., Fetahu, B., Kar, S., and Rokhlenko, O. (2022a). MultiCoNER: a Large-Scale Multilingual Dataset for Complex Named Entity Recognition. *arXiv preprint arXiv:2208.14536*.

Malmasi, S., Fang, A., Fetahu, B., Kar, S., and Rokhlenko, O. (2022b). SemEval-2022 Task 11: Multilingual Complex Named Entity Recognition (MultiCoNER). In *International Workshop on Semantic Evaluation (SemEval)*. Association for Computational Linguistics.

Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The DET Curve in Assessment of Detection Task Performance. Technical report, National Inst of Standards and Technology Gaithersburg MD.

Matejka, P., Novotnỳ, O., Plchot, O., Burget, L., Sánchez, M. D., and Cernockỳ, J. (2017). Analysis of Score Normalization in Multilingual Speaker Recognition. In *Interspeech*, pages 1567–1571.

Mclaren, M., Castán, D., Nandwana, M. K., Ferrer, L., and Yilmaz, E. (2018). How to Train Your Speaker Embeddings Extractor. In *Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 327–334. ISCA.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Mingote, V., Miguel, A., Ortega, A., and Lleida, E. (2020). Optimization of the Area Under the ROC Curve using Neural Network Supervectors for Text-Dependent Speaker Verification. *Computer Speech & Language*, 63:101078.

Monaikul, N., Castellucci, G., Filice, S., and Rokhlenko, O. (2021). Continual Learning for Named Entity Recognition. In *AAAI Conference on Artificial Intelligence*.

Mueller, D., Andrews, N., and Dredze, M. (2020). Sources of Transfer in Multilingual Named Entity Recognition. In *Annual Meeting of the Association for Computational Linguistics*, pages 8093–8104, Online. Association for Computational Linguistics.

Musgrave, K., Belongie, S., and Lim, S.-N. (2020). A Metric Learning Reality Check. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *European Conference on Computer Vision (ECCV)*, pages 681–699, Cham. Springer International Publishing.

Nagrani, A., Chung, J. S., and Zisserman, A. (2017). VoxCeleb: A Large-Scale Speaker Identification Dataset. In *Interspeech*, pages 2616–2620.

Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *International Conference on Machine Learning (ICML)*.

Nakayama, H. (2018). seqeval: A Python Framework for Sequence Labeling Evaluation. Software available at github.com/chakki-works/seqeval.

Okafor, E., Pawara, P., Karaaba, F., Surinta, O., Codreanu, V., Schomaker, L., and Wiering, M. (2016). Comparative Study Between Deep Learning and Bag of Visual Words for Wild-Animal Recognition. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8.

Otterson, S. and Ostendorf, M. (2007). Efficient Use of Overlap Information in Speaker Diarization. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 683–686. IEEE.

Paik, I., Oh, S., Kwak, T., and Kim, I. (2020). Overcoming Catastrophic Forgetting by Neuron-Level Plasticity Control. *AAAI Conference on Artificial Intelligence*, 34(04):5339–5346.

Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Park, T. J., Han, K. J., Kumar, M., and Narayanan, S. (2019). Auto-Tuning Spectral Clustering for Speaker Diarization Using Normalized Maximum Eigengap. *IEEE Signal Processing Letters*.

Park, T. J., Kanda, N., Dimitriadis, D., Han, K. J., Watanabe, S., and Narayanan, S. (2022). A Review of Speaker Diarization: Recent Advances with Deep Learning. *Computer Speech & Language*, 72:101317.

Pascual, S., Ravanelli, M., Serrà, J., Bonafonte, A., and Bengio, Y. (2019). Learning Problem-Agnostic Speech Representations from Multiple Self-Supervised Tasks. In *Interspeech*, pages 161–165.

Pellegrini, L., Graffieti, G., Lomonaco, V., and Maltoni, D. (2020). Latent Replay for Real-Time Continual Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep Contextualized Word Representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pomponi, J., Scardapane, S., Lomonaco, V., and Uncini, A. (2020). Efficient Continual Learning in Neural Networks with Embedding Regularization. *Neurocomputing*, 397:139–148.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ramshaw, L. and Marcus, M. (1995). Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. (2019). Continual Unsupervised Representation Learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, pages 7645–7655. Curran Associates, Inc.

Ravanelli, M. and Bengio, Y. (2018). Speaker Recognition from Raw Waveform with SincNet. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028.

Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., and Bengio, Y. (2020). Multi-Task Self-Supervised Learning for Robust Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6989–6993.

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., and Kim, B. (2019). Visualizing and Measuring the Geometry of BERT. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Robins, A. (1995). Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2):123–146.

Rosenberg, A. E., DeLong, J., Lee, C.-H., Juang, B.-H., and Soong, F. K. (1992). The Use of Cohort Normalized Scores for Speaker Verification. In *International Conference on Spoken Language Processing (ICSLP 1992)*, pages 599–602.

Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*. PhD thesis, NUI Galway.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323(6088):533–536.

Ryant, N., Church, K., Cieri, C., Cristia, A., Du, J., Ganapathy, S., and Liberman, M. (2019). The Second DIHARD Diarization Challenge: Dataset, Task, and Baselines. In *Interspeech*, pages 978–982.

Ryant, N., Church, K., Cieri, C., Du, J., Ganapathy, S., and Liberman, M. (2020a). Third DIHARD Challenge Evaluation Plan. *arXiv preprint arXiv:2006.05815*.

Ryant, N., Singh, P., Krishnamohan, V., Varma, R., Church, K., Cieri, C., Du, J., Ganapathy, S., and Liberman, M. (2020b). The Third DIHARD Diarization Challenge. *arXiv preprint arXiv:2012.01477*.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823. arXiv: 1503.03832.

Schuster, S., Gupta, S., Shah, R., and Lewis, M. (2019). Cross-lingual Transfer Learning for Multilingual Task Oriented Dialog. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3795–3805, Minneapolis, Minnesota. Association for Computational Linguistics.

Settles, B. (2009). Active Learning Literature Survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual Learning with Deep Generative Replay. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, pages 2990–2999. Curran Associates, Inc.

Snyder, D., Chen, G., and Povey, D. (2015). MUSAN: A Music, Speech, and Noise Corpus. *arXiv preprint arXiv:1510.08484*.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018). x-vectors: Robust DNN Embeddings for Speaker Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333.

Srivastava, Y., Murali, V., and Dubey, S. R. (2020). A Performance Evaluation of Loss Functions for Deep Face Recognition. In Babu, R. V., Prasanna, M., and Namboodiri, V. P., editors, *Computer Vision, Pattern Recognition, Image Processing, and Graphics*, pages 322–332, Singapore. Springer Singapore.

Sun, F.-K., Ho, C.-H., and Lee, H.-Y. (2020). LAMAL: LAnguage Modeling Is All You Need for Lifelong Language Learning. In *International Conference on Learning Representations (ICLR)*.

Takashima, Y., Fujita, Y., Horiguchi, S., Watanabe, S., Garc'ia, P., and Nagamatsu, K. (2021). Semi-Supervised Training with Pseudo-Labeling for End-to-End Neural Diarization. In *Interspeech 2021*.

Terekhov, A. V., Montone, G., and O'Regan, J. K. (2015). Knowledge Transfer in Deep Block-Modular Neural Networks. In *Conference on Biomimetic and Biohybrid Systems*, pages 268–279. Springer.

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the CoNLL-2000 Shared Task Chunking. In *Conference on Computational Natural Language Learning, Learning Language in Logic Workshop*.

Traer, J. and McDermott, J. H. (2016). Statistics of Natural Reverberation Enable Perceptual Separation of Sound and Space. *Proceedings of the National Academy of Sciences*, 113(48):E7856–E7865.

Tzinis, E., Adi, Y., Ithapu, V. K., Xu, B., and Kumar, A. (2022). Continual Self-Training with Bootstrapped Remixing for Speech Enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6947–6951.

Upadhyay, S., Faruqui, M., Tür, G., Dilek, H.-T., and Heck, L. (2018). (Almost) Zero-Shot Cross-Lingual Spoken Language Understanding. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038.

Vanschoren, J. (2019). Meta-Learning. In *Automated Machine Learning*, pages 35–61. Springer, Cham.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is All You Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wan, L., Wang, Q., Papir, A., and Moreno, I. L. (2018). Generalized End-to-End Loss for Speaker Verification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883.

Wang, Z., K, K., Mayhew, S., and Roth, D. (2020). Extending Multilingual BERT to Low-Resource Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Warren, C. (2021). EchoThief Impulse Response Library. Available at www.echothief.com/downloads.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A Discriminative Feature Learning Approach for Deep Face Recognition. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *European Conference on Computer Vision (ECCV)*, volume 9911, pages 499–515. Springer International Publishing, Cham.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). HuggingFace's Transformers: State-of-the-Art Natural Language Processing. *arXiv preprint arXiv:1910.03771*.

Xia, Y., Wang, Q., Lyu, Y., Zhu, Y., Wu, W., Li, S., and Dai, D. (2022). Learn and Review: Enhancing Continual Named Entity Recognition via Reviewing Synthetic Samples. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2291–2300, Dublin, Ireland. Association for Computational Linguistics.

Xu, W., Haider, B., and Mansour, S. (2020). End-to-End Slot Alignment and Recognition for Cross-Lingual NLU. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5052–5063, Online. Association for Computational Linguistics.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. (2021a). mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Xue, Y., Horiguchi, S., Fujita, Y., Takashima, Y., Watanabe, S., Perera, L. P. G., and Nagamatsu, K. (2021b). Online Streaming End-to-End Neural Diarization Handling Overlapping Speech and Flexible Numbers of Speakers. In *Interspeech*, pages 3116–3120.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2018). Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations (ICLR)*.

Yoshioka, T., Erdogan, H., Chen, Z., and Alleva, F. (2018). Multi-Microphone Neural Speech Separation for Far-Field Multi-Talker Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5739–5743.

Yu, D., Kolbæk, M., Tan, Z.-H., and Jensen, J. (2017). Permutation Invariant Training of Deep Models for Speaker-Independent Multi-Talker Speech Separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual Learning Through Synaptic Intelligence. In Precup, D. and Teh, Y. W., editors, *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.

Zhang, C. and Koishida, K. (2017). End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances. In *Interspeech*, pages 1487–1491.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *IEEE International Conference on Computer Vision (ICCV)*.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage de représentation en continu pour la langue écrite et parlée

Mots clés: 3 à 6 mots clefs (version en français)

Résumé: L'apprentissage automatique a récemment connu des avancées majeures, mais les modèles actuels sont généralement entraînés une fois sur une tâche cible et leurs paramètres sont rarement révisés. Ce problème affecte les performances après la mise en production car les spécifications des tâches et les données peuvent évoluer avec le temps. Pour résoudre ce problème, l'apprentissage continu propose un entraînement au fil du temps, à mesure que de nouvelles données sont disponibles. Cependant, les modèles entraînés de cette manière souffrent d'une perte de performance sur les exemples déjà vus, un phénomène appelé *oubli catastrophique*. De nombreuses études ont proposé différentes stratégies pour prévenir l'oubli, mais elles s'appuient souvent sur des données étiquetées rarement disponibles en pratique.

Dans cette thèse, nous étudions l'apprentissage continu pour la langue écrite et parlée. Notre objectif est de concevoir des systèmes autonomes et auto-apprenants capables d'exploiter les données disponibles sur le terrain pour s'adapter aux nouveaux environnements. Contrairement aux travaux récents sur l'apprentissage de représentations à usage général, nous proposons d'exploiter des représentations adaptées à une tâche cible. En effet, ces dernières pourraient être plus faciles à interpréter et à exploiter par des méthodes non supervisés et plus robustes à l'oubli, comme le clustering.

Dans ce travail, nous améliorons notre compréhension de l'apprentissage continu dans plusieurs contextes. Nous montrons que les représentations spécifiques à une tâche permettent un apprentissage continu efficace à faibles ressources, et que les prédictions d'un modèle peuvent être exploitées pour l'auto-apprentissage.

Title: Continual Representation Learning in Written and Spoken Language

Keywords: continual learning, representation learning, metric learning, speaker diarization, sequence labeling

Abstract: Although machine learning has recently witnessed major breakthroughs, today's models are mostly trained once on a target task and then deployed, rarely (if ever) revisiting their parameters. This problem affects performance after deployment, as task specifications and data may evolve with user needs and distribution shifts. To solve this, *continual learning* proposes to train models over time as new data becomes available. However, models trained in this way suffer from significant performance loss on previously seen examples, a phenomenon called *catastrophic forgetting*. Although many studies have proposed different strategies to prevent forgetting, they often rely on labeled data, which is rarely available in practice.

In this thesis, we study continual learning for written and spoken language. Our main goal is to design autonomous and self-learning systems able to leverage scarce on-the-job data to adapt to the new environments they are deployed in. Contrary to recent work on learning general-purpose representations (or *embeddings*), we propose to leverage representations that are tailored to a downstream task. We believe the latter may be easier to interpret and exploit by unsupervised training algorithms like clustering, that are less prone to forgetting.

Throughout our work, we improve our understanding of continual learning in a variety of settings, such as the adaptation of a language model to new languages for sequence labeling tasks, or even the adaptation to a live conversation in the context of speaker diarization. We show that task-specific representations allow for effective low-resource continual learning, and that a model's own predictions can be exploited for full self-learning.