End-to-End Neural Approaches for Speech Translation

Manh Ha Nguyen

Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : **25 mai 2016**

Présentée par

Ha NGUYEN

Thèse dirigée par **Yannick ESTÈVE**
et codirigée par **Laurent BESACIER**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG), Laboratoire Informatique d'Avignon (LIA)**
et de l'**École Doctorale MSTII - Mathématiques, Sciences et Technologies de l'Information, Informatique**

End-to-End Neural Approaches for Speech Translation

Approches neurales de bout-en-bout pour la traduction de la parole

Thèse soutenue publiquement le **3 juin 2022**,
devant le jury composé de :

Monsieur Yannick ESTÈVE
Professeur des Universités, Université d'Avignon, Directeur de thèse

Monsieur Laurent BESACIER
Professeur, NAVER LABS Europe, Co-Directeur de thèse

Monsieur Satoshi NAKAMURA
Professeur, Nara Institute of Science and Technology, Rapporteur

Monsieur Alexandre ALLAUZEN
Professeur, ESPCI PARIS, Rapporteur

Madame Catherine BERRUT
Professeur des Universités, Université Grenoble Alpes, Présidente

Monsieur Loïc BARRAULT
Chercheur, Meta AI, Examinateur

Ha Nguyen
@nmh226

I planned to get married today, but I submitted an Interspeech paper instead.
#interspeech2020 #covidyear1

5:52 PM · May 15, 2020 · Twitter Web App

Abstract

The introduction of speech translation corpora, which have speech signals aligned with the corresponding translation texts, coupled with the steady growth in the computational capacity, plays a crucial role in making the training of neural end-to-end speech-to-text translation feasible. The endeavor of this thesis is exploring neural approaches for end-to-end speech-to-text translation, which shall be referred to as Automatic Speech Translation, particularly focusing on two types of end-to-end translation systems: (1) Offline Speech Translation and (2) Online Speech Translation.

With respect to offline speech translation, we build strong end-to-end baselines for two language pairs English-to-Portuguese and English-to-German. They are based on VGG-like Convolutional Neural Network blocks coupled with Long Short-Term Memory layers at the encoder side and a stack of Long Short-Term Memory layers at the decoder side. We investigate different data augmentation techniques as well as different target token units (characters, Byte Pair Encoding of different sizes) and validate those baselines through our participation in international shared tasks on speech translation. Besides, we put Self-Supervised Learning from speech representations, particularly pre-trained English wav2vec, into a comparison with the conventional speech representations including Mel filter-bank and MFCC features, when applied to the speech translation task, specifically in low and medium-resource scenarios, when we have less than 100 hours of training data. We explain through analyses that wav2vec features might be better at discriminating phones, better at aligning source and target sequences, and more robust to speaker variability. Last but not least, we train our own Self-Supervised Learning models from a large amount of unlabelled French speech data, which are then proven effective for a wide range of speech tasks that are included in a reproducible framework for assessing self-supervised representation learning from speech named LeBenchmark.

As regards online speech translation, we adapt wait-k policy, which is originally proposed for online text-to-text translation, for the speech translation task, and advocate for using Unidirectional instead of Bidirectional Long Short-Term Memory speech encoders for online speech translation. We propose a new encoding strategy named Unidirectional Long Short-Term Memory Overlap-and-Compensate, which allows Unidirectional Long Short-Term Memory-based speech encoders to work more effectively in online speech translation. We evaluate both the decoding and encoding strategies firstly on the ability to leverage pre-trained offline end-to-end speech translation models for the online translation task. Furthermore, we propose to fine-tune these

pre-trained models in a training mode more adapted to online translation to further boost the performance of the online translation systems. In addition, other aspects of online speech translation, for instance, the impact of input speech segmentation, the impact of output granularity, and different fine-tuning scenarios, are also investigated.

Keywords: End-to-end automatic speech translation, neural machine translation, low latency speech translation, self-supervised learning for speech translation.

Résumé

La disponibilité de corpus de traduction de la parole dont les signaux de parole sont alignés avec les textes traduits correspondants, couplée à l'augmentation constante de la capacité de calcul, rend désormais possible l'entraînement de systèmes automatiques de traduction parole-texte de bout-en-bout. L'objectif de cette thèse est d'explorer les approches neuronales pour cette tâche de traduction parole-texte, appelée traduction automatique de la parole, en se concentrant particulièrement sur deux types de systèmes de traduction de bout-en-bout: (1) Traduction de la parole hors ligne (offline speech translation) et (2) Traduction de la parole en ligne (online speech translation).

En ce qui concerne la traduction hors ligne, nous développons des baselines solides pour deux paires de langues : anglais-portugais et anglais-allemand. Elles sont fondées sur des blocs de réseaux neuronaux convolutifs couplés à des couches récurrentes de type LSTM (Long Short-Term Memory) côté encodeur et à plusieurs couches LSTM côté décodeur. Nous étudions différentes techniques d'augmentation de données ainsi que différentes unités lexicales cibles (caractères, unités BPE de différentes tailles). Nous validons nos méthodes en participant à des campagnes internationales d'évaluation de traduction de la parole. Nous introduisons aussi, dans cette thèse, l'utilisation de représentations issues de l'apprentissage auto-supervisé (en utilisant un modèle de type wav2vec) et les comparons avec des représentations conventionnelles (dont les coefficients MFCCs et les coefficients en bancs de filtres) pour la tâche de traduction de la parole. Cette comparaison est effectuée en particulier dans des scénarios avec des ressources faibles ou moyennes (moins de 100 heures de données d'entraînement). Nous effectuons des analyses qui montrent que les représentations auto-supervisées améliorent les performances de nos modèles et sont aussi plus efficaces pour discriminer les phonèmes et aligner les séquences source et cible, ainsi que plus robustes à la variabilité des orateurs. Enfin, nous entraînons nos propres modèles d'apprentissage auto-supervisés à partir d'une grande quantité de données brute de parole en français. De tels modèles sont utiles pour un large éventail de tâches concernant la parole. Ces tâches sont incluses dans une suite d'évaluation open-source pour l'apprentissage auto-supervisé, nommée 'LeBenchmark'.

Concernant la traduction de la parole en ligne, nous adaptons la stratégie wait-k, initialement proposée pour la traduction simultanée texte-texte, à la tâche de traduction de la parole. Pour la traduction simultanée de la parole, nous préconisons l'utilisation d'encodeurs LSTM unidirectionnels plutôt que bidirectionnels. Nous proposons une nouvelle stratégie d'encodage nommée 'Unidirectional Long Short-Term Memory Overlap-and-Compensate', qui

permet aux encodeurs de parole LSTM unidirectionnels de fonctionner plus efficacement en ligne. Tout d'abord, nous évaluons nos stratégies de décodage et d'encodage sur la tâche de traduction en ligne. Après, nous proposons d'ajuster ces modèles pré-entraînés (par réglage fin) dans un mode d'apprentissage plus adapté à la traduction en ligne pour encore améliorer les performances. Enfin, d'autres aspects de la traduction en ligne de la parole sont étudiés, tels que l'impact de la segmentation des données en entrée, l'impact de la granularité de sortie ou encore différents scénarios de réglage fin.

Mots clés: Modèles de bout-en-bout pour la traduction automatique de la parole, traduction automatique neuronale, traduction de parole à faible latence, apprentissage auto-supervisé pour la traduction de la parole.

Acknowledgements

This work could not be accomplished without the endless help and encouragement that I have been receiving from those who I would like to express my profound gratitude to.

Firstly, I would like to thank Satoshi Nakamura and Alexandre Allauzen for thoroughly reviewing this dissertation. I also extend my thank to Catherine Berrut and Loïc Barrault for agreeing to be members of the jury.

I would like to thank my co-supervisor Laurent Besacier, who put a start to this journey. Laurent, I owe you a big thank for taking the risk of recruiting me right after I came to France 5 years ago. I was then an insignificant boy from Vietnam equipped with limited knowledge about NLP, although I had my burning enthusiasm which had driven me all the way to GETALP. Thank you for sending that same me with a bit of upgraded knowledge to WIPO for an internship that I would never forget. And thank you for connecting me to Yannick and subsequently to this wonderful PhD that I have been pursuing. I see myself learning a great deal thanks to your close supervision. You might not remember this but your saying *"I only have good students"* several years ago in Graz has been helping me get through ups and downs in this journey, because I have allowed myself to be one of those you mentioned.

I would like to thank my supervisor Yannick Estève for the superb supervision and encouragement that I have been receiving from him. It is a pity that I could not come to LIA more often due to the pandemic. However, you always gave me the warmest welcome every time I had a chance to come to your amazing laboratory in Avignon. I love everything about it. I thank you for making Laurent's leave less dismaying than it should have been. You might well recognize that I had a bit of difficulty at the very beginning, but your encouragement and comprehension adequately padded Laurent's absence. Please do not worry about your Zoom's background not looking so professional with your nice collection of guitars hanging in the back because I believe that everybody loves it. And I am also grateful for your saying *"I am sure that you have all skills and competencies for that"*, which has been immeasurably strengthening my self-confidence.

I am also deeply thankful for being in the same teams with people from both GETALP, LIA, LIUM, and Airbus. Beyond any doubt, these teams, comprising of Marcely Zanon Boito, Solène "Andwine", William Havard, Mahault Garnerin, Hang Le, Maha Elbayad, Marco Dinarelli, Benjamin Lecouteux, Francois Portet, Didier Schwab, Antoine Caubrière, Titouan Parcollet, Natalia Tomashenko, Salima Mdhaffar, Fethi Bougares, Florentin Barbier, Souhir Gahbiche and others, are the strongest and also coolest teams that I

have ever worked with. Thank you for co-writing most of the papers that are listed here in this thesis, and also thank you for enjoying very good coffee breaks with me, which have been transforming very well to the higher working performance of mine.

Last but not least, I would like to send the warmest words to my family, who have been incredibly patient and faithful to me and all of my decisions. Thank you, Mom and Grandma, for wholly comprehending that I need to go this far from home in order to define myself and to realize what I want to do in my career. I am also grateful for the endless help and encouragement from my aunts, uncles, cousins, and my amusing little niece throughout the years, even though I have been living a long distance from them. And I would like to say thank to my not-yet-married-due-to-COVID19 wife, who had made a difficult decision on coming to France to stand by my side in this journey.

Contents

List of Publications

The following papers were published as part of this thesis.

International Conference Proceedings

- Evain, S., **Nguyen, H.**, Le, H., Boito, M. Z., Mdhaffar, S., Alisamir, S., et al., ***"Task Agnostic and Task Specific Self-Supervised Learning from Speech with LeBenchmark"***, NeurIPS 2021.
- **H. Nguyen**, Y. Estève, and L. Besacier, ***"Impact of encoding and segmentation strategies on end-to-end simultaneous speech translation"***, in Interspeech 2021, Brno, Czech Republic, 2021.
- S. Evain, **H. Nguyen**, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet, et al., ***"Lebenchmark: A reproducible framework for assessing self-supervised representation-learning from speech"***, in Interspeech 2021, Brno, Czech Republic, 2021.
- **H. Nguyen**, Y. Estève, and L. Besacier, ***"An empirical study of end-to-end simultaneous speech translation decoding strategies"***, in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021.
- **Ha Nguyen**, Fethi Bougares, Natalia Tomashenko, Yannick Estève, Laurent Besacier, ***"Investigating Self-supervised Pre-training for End-to-end Speech Translation"***, Interspeech 2020, Oct 2020, Shangai (Virtual Conf), China.

International Workshops

- Marcely Zanon Boito, John Ortega, Hugo Riguidel, Antoine Laurent, Loïc Barrault, Fethi Bougares, Firas Chaabani, **Ha Nguyen**, Florentin Barbier, Souhir Gahbiche, and Yannick Estève. 2022. ***"ON-TRAC Consortium Systems for the IWSLT 2022 Dialect and Low-resource Speech Translation Tasks"***. In Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022), pages 308–318, Dublin, Ireland (in-person and online). Association for Computational Linguistics.
- Hang Le, Florentin Barbier, **Ha Nguyen**, Natalia Tomashenko, Salima Mdhaffar, et al.. ***"ON-TRAC' systems for the IWSLT 2021 low-resource speech translation and multilingual speech translation shared tasks"***. International Conference on Spoken Language Translation (IWSLT), Aug 2021, Bangkok (virtual), Thailand.

- **Ha Nguyen**, Fethi Bougares, Natalia Tomashenko, Yannick Estève, Laurent Besacier, ***"Investigating Self-supervised Pre-training for End-to-end Speech Translation"***, ICML 2020 workshop on Self-supervision in Audio and Speech (SAS), 2020.

- M. Elbayad, **H. Nguyen**, F. Bougares, N. Tomashenko, A. Caubrière, B. Lecouteux, Y. Estève, and L. Besacier, ***"ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020"***, in The International Conference on Spoken Language Translation ACL - 17th IWSLT, Seattle, WA, United States, Jul. 2020.

- **Nguyen, H.**, Tomashenko, N., Boito, M. Z., Caubrière, A., Bougares, F., Rouvier, M., Besacier, L., and Estève, Y. ***"ON-TRAC consortium end-to-end speech translation systems for the IWSLT 2019 shared task"***, *International Workshop on Spoken Language Translation (IWSLT 2019).*

French Conferences

- Solène Evain, **Ha Nguyen**, Hang Le, Marcely Zanon Boito, Salima Mdhaffar, Sina Alisamir, Ziyi Tong, Natalia Tomashenko, Marco Dinarelli, Titouan Parcollet, Alexandre Allauzen, Yannick Estève, Benjamin Lecouteux, François Portet, Solange Rossato, Fabien Ringeval, Didier Schwab and Laurent Besacier ***"Modèles neuronaux pré-appris par auto-supervision sur des enregistrements de parole en français"***, Les 34e Journées d'Études sur la Parole (JEP2022).

- Hang Le, Sina Alisamir, Marco Dinarelli, Fabien Ringeval, Solène Evain, **Ha Nguyen**, Marcely Zanon Boito, Salima Mdhaffar, Ziyi Tong, Natalia Tomashenko, Titouan Parcollet, Allauzen Alexandre, Yannick Estève, Benjamin Lecouteux, François Portet, Solange Rossato, Didier Schwab and Laurent Besacier, ***"LeBenchmark, un référentiel d'évaluation pour le français oral"***, Les 34e Journées d'Études sur la Parole (JEP2022).

List of Figures

List of Tables

Introduction

Context and motivation

Moving from text-to-text machine translation to speech-to-text machine translation is one-step closer to the age-old dream of humankind, which is easing the language barrier between people of different communities.

Efforts to translate speech automatically dated back to the 1980s, when NEC performed the first proof of concept at the 1983 ITU Telecom World (Nakamura (2009)) and continued flourishing in the 1990s. For decades, the field of speech translation (from speech-to-text specifically) had been witnessing the dominance of two-stage complex cascaded approaches, which couple an automatic speech recognition system followed by a text-to-text machine translation system. For this reason, this field is usually said to be much more challenging than speech recognition and text translation standing alone as it must solve problems coming from the two fronts, which, in their own turns, are far from being solved.

Starting from 2016, end-to-end neural approaches, which sought to tackle cascaded methods' shortcomings, have been emerging and directly challenging the decades-old dominance of cascaded approaches. Pioneers of this research branch (Duong et al. (2016); Bérard et al. (2016); Weiss et al. (2017); Bérard et al. (2018)) argue that two-stage cascaded models are prone to error propagation since the two main components of the system are trained to optimize two separate objective functions. Therefore, they propose to train single speech translation systems which directly predict hypothesis translation from the input sequence, optimizing a single objective function. This way of doing things has been gradually gaining more and more interest from the community, thanks to the proven effectiveness of sequence-to-sequence models for machine translation, and speech recognition tasks (Chorowski et al. (2015); Chan et al. (2016); Zhang et al. (2017); Chorowski and Jaitly (2016)), and the efforts to build speech translation corpora (i.e, parallel data of recorded speech coupled with translation text) which allow training end-to-end speech translation models without using source transcription. Since 2016, this research field has been booming vigorously. However, due to the enormous challenges imposed by this task, it is still nowhere near being solved Sperber and Paulik (2020).

The aforementioned end-to-end speech translation models, which enjoy the advantage of having the whole input sequence available for conditioning the generation of the output translation, are referred to as offline speech trans-

lation. This is said to be less challenging than end-to-end online (sometimes dubbed as *"simultaneous"*) speech translation (Ren et al. (2020); Ma et al. (2020b); Han et al. (2020); Ma et al. (2021)), which has to generate output hypothesis incrementally from partial input speech. This rather newborn research branch is attracting more and more attention from the community by early but encouraging results presented at evaluation campaigns such as the IWSLT 2020 and 2021. However, the problem of end-to-end online translation is also far from being solved, and therefore, attempts to improve the status quo of the field are strongly welcomed.

First and foremost, this thesis is centered on exploring neural methods for end-to-end speech translation. This consists of a wide range of aspects, such as searching for effective end-to-end models, how to deal with the scarceness of the speech translation training data, how to efficiently segment the speech input, which kind of target token units is most beneficial, etc. This also includes answering a research question of how to effectively represent speech input, which is the second focus of this thesis. Particularly, an investigation of the newly proposed self-supervised learning from speech representations (Schneider et al. (2019a); Baevski et al. (2020b))) is conducted, which shall advocate for replacing the conventional approaches of representing speech, such as Mel fitler-bank features and MFCC features, etc., by these self-supervised learning features.

Last but not least, another contribution of this thesis is on online speech translation. Particularly, we aim to adapt wait-k policy (Ma et al. (2019)), a decoding policy proposed for online text-to-text translation, to online speech translation. We couple this adapted decoding strategy with a new Unidirectional Long Short-Term Memory (ULSTM) overlap-and-compensate encoding strategy in order to, firstly, leverage pre-trained offline end-to-end speech translation models in online mode, and secondly, fine-tune these models in online mode to further boost the performance of the online translation systems. Other different aspects of online speech translation, for example, the impact of input speech segmentation and the impact of output granularity, are also studied in this thesis.

Contribution overview

This thesis studies the problem of modeling neural speech translation systems, focusing on the following aspects:

- End-to-end modeling for offline speech translation: we focus on exploring different end-to-end architectures for speech translation, as well as different target token types. We shall argue through experiments that

LSTM-based attentional encoder-decoder architecture is the most beneficial, and that character-based models perform better than their BPE-based counterparts in our settings. This quest for end-to-end models for offline speech translation results in two publications for the IWSLT 2019 and IWSLT 2020 workshop.

- Speech representations for translation: we compare the conventional approaches for speech representation such as Mel filter-bank or MFCC features, with a new kind of speech representation based on self-supervised learning. We show that wav2vec features, which are based on contrastive predictive coding, outperform conventional features by a large margin in low-resource conditions where speech translation training data is not sufficiently available. This investigation of self-supervised learning from speech features results in several publications to several prestigious international conferences such as the Interspeech 2020, the Interspeech 2021, and the NeurIPS 2021.

- End-to-end online translation system: we concentrate on balancing between translation quality and latency of end-to-end translation systems. A wait-k like decoding strategy is proposed, and proven to be sufficient for leveraging pre-trained offline end-to-end models for online speech translation. In order to further improve the performance of the online translation system, we also propose an encoding strategy namely ULSTM overlap-and-compensate, which allows VGG-like speech encoders with ULSTM layers to encode the partial speech input more efficiently. In addition, we also advocate for fine-tuning offline models in a training that is more adapted to online translation, and show that it can help improve the performance of the online translation system while keeping a reasonable developing cost. These works result in 2 publications to 2 high-ranking international conferences including the ICASSP 2021 and the Interspeech 2021.

Thesis outline

This dissertation is organized into two main parts. In the first part of the dissertation, we discuss the background knowledge that is closely related to the work carried out in this thesis. This consists of:

- **Chapter 1: Neural Machine Translation.** We present in this chapter an overview of the older methods that had been dominant before the era of end-to-end neural machine translation, for example, the word-based and

phrase-based statistical machine translation. This will be followed by the state-of-the-art of neural methods, specifically end-to-end machine translation. Besides, this chapter also describes commonly used metrics for evaluating a machine translation system.

- **Chapter 2: Neural Speech Translation.** This chapter is dedicated to presenting the state-of-the-art of neural end-to-end speech translation. Before doing that, we also discuss briefly cascaded systems, which remain strong baselines that end-to-end models need to surpass. After this, we emphasize some important end-to-end architectures widely used in speech translation, as well as the challenges and their corresponding solutions. This chapter ends with an overview of the speech translation corpora.

- **Chapter 3: Online Neural Machine Translation.** This chapter aims to present the state-of-the-art of online neural machine translation. In this chapter, our focus is on the discussion about different online decoding strategies of both deterministic and dynamic nature. Besides, we also mention how an online machine translation system can be automatically evaluated.

- **Chapter 4: Online Neural Speech Translation.** In this chapter, we discuss briefly early attempts on online speech translation systems, before giving an overview of cascaded online speech translation models. This will be followed by the discussion about end-to-end models for online speech translation, which are closely related to what we are aiming to do in this thesis.

- **Chapter 5: Self-supervised Learning Speech Representation**. In this chapter, we first give an overview of the conventional approaches of representing speech data. After that, a detailed discussion centered on self-supervised learning from speech methods, which are promising to replace the conventional approaches for speech representations, shall be given.

The second part of the thesis aims to detail the scientific contributions of this thesis, particularly in the following aspects:

- **Chapter 6: Offline Neural Speech Translation.** We explore different end-to-end architectures for automatic speech translation, The discussion will be confined in the context of our participation in the IWSLT 2019 and IWSLT 2020 evaluation campaigns, where automatic translations generated by these end-to-end models are evaluated and compared with methods from other research groups.

- **Chapter 7: Self-supervised Learning Speech Representation.** This chapter aims to discuss our contributions in leveraging the pre-trained self-supervised learning English model to generate speech features for the speech translation task, as well as our efforts to train our own SSL models from French speech. We show in this chapter results that are in favor of using self-supervised learning speech representations instead of the conventional ones.

- **Chapter 8: Online Neural Speech Translation.** We discuss in this chapter our contributions in terms of both decoding and encoding approaches for online speech translation, which make effective use of pre-trained offline speech translation models in the online translation task. We also show in this chapter how these pre-trained models can be fine-tuned in a training more adapted to online translation to further improve the performance of the online translation system.

We conclude this thesis in **Conclusion**, where we summarize our contributions as well as discuss our perspective on the future work.

Part I

State of the Art

Chapter 1

Neural Machine Translation

This chapter lays out the background for our works, giving an overview of the state-of-the-art methods for Neural Machine Translation.

1.1 Definition

Human beings have dreamed to be able to seamlessly communicate between different languages for ages. Efforts to realize this dream by using machines to translate across languages dated back almost as far as electronic computers came into existence. During World War II, computers were used in Britain to crack the German Enigma code. This, to Warren Weaver - one of the pioneers in machine translation, seemed like an appropriate metaphor for machine translation. In his 1949 memorandum, he wrote:

> *When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode'.* [Weaver, 1949]

Machine Translation, as the name might reveal, is all about using machines (mostly computer software) to automatically translate text or speech from one language to another. However, in literature, this term is used somewhat abusively to refer to text-to-text machine translation (**MT**) [1], while speech-to-text machine translation is often referred as Automatic Speech Translation (**AST**). Even though text-to-text translation is not the main concern of this thesis, this chapter is dedicated to discussing fundamental aspects of text-to-text translation, which are well transferred to the works in speech-to-text translation.

Early efforts in the field revolved around *Rule-base* methods (*RBMT*), which was based essentially on several linguistic rules, for example, rules for syntactic analysis, rules for morphology, lexical rules, etc. Three representative methods of RBMT are:

[1] More specifically, the term Machine Translation (MT) is commonly used to refer to Offline text-to-text machine translation. The same for Automatic Speech Translation (AST), which usually refers to offline speech-to-text machine translation.

- *Direct translation*: uses large bilingual dictionaries to translate word-by-word a source sentence to a target sentence. This is done by mapping each source word to its target word, without complicated analysis and syntactic reorganisation.
- *Transfer-based translation*: is an indirect approach, which translates in multiple stages, for example, the translation system first *analyzes* the source sentence to determine its structure, and then *transfers* the resulting structure to the target language, and finally, *generates* the target language sentence based on the transferred knowledge.
- *Interlingual translation*: is another indirect approach, which is based on abstract language representations independent from both source and target language. In this approach, the translation process is done in two stages: (1) encoding the source sentence onto an *interlingua*, and then (2) decoding the target from interlingua.

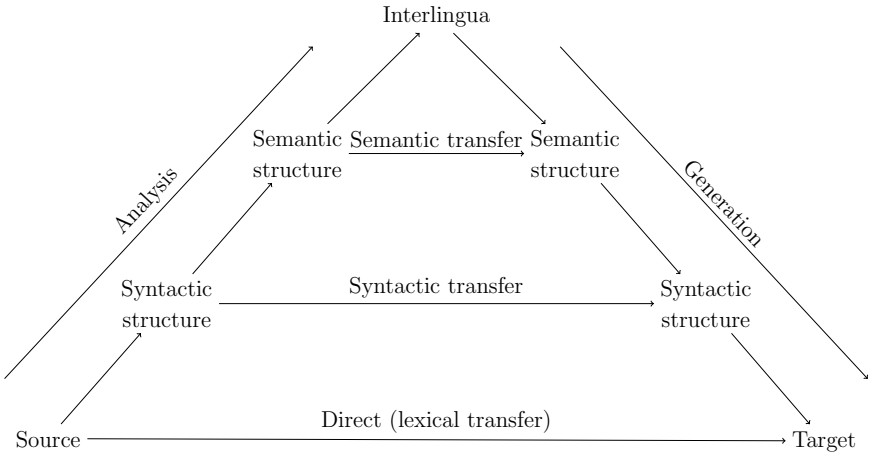


Figure 1.1: The Vauquois triangle (Vauquois (1968)).

Up until the late 1980s, MT systems were mostly RBMT, whose differences (in terms of the depth of analysis and abstraction of the language) can be illustrated by the famous Vauquois triangle (Vauquois (1968)) (Figure 1.1). Since 1989, RBMT's dominance has been challenged by the emergence of the new *data-driven* methods, most aggressively by *Statistical MT (SMT)*, whose dominance has been broken by yet another branch of research called *Neural MT (NMT)* since the 2010s. The remaining of this chapter is dedicated to discussing different aspects of both SMT and NMT methods.

1.2 Statistical Machine Translation

One of the groundbreaking works that challenged the dominance of rule-based MT was done by a group at IBM (Brown et al. (1993)). The distinctive feature of their works was to solely use statistical methods as the means of analysis and generation without caring about linguistic rules (Hutchins (2007)). Using Bayes's rule, Brown et al. (1993) formulate the problem of translating a French sentence to an English sentence as the following:

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)} \tag{1.1}$$

with $P(e|f)$ being a probability, which can be interpreted as the probability that e is produced as a translation by a translator presented with f. Since $P(f)$ is independent of e, finding the translation $\hat{e}$ is equivalent to maximizing the product $P(e)P(f|e)$. We thus arrive at what is called by Brown et al. (1993) the *"Fundamental Equation of Machine Translation"*:

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, P(e)P(f|e) \tag{1.2}$$

Note that, mathematically speaking, the problem is now reversed: instead of modeling $P(e|f)$ (the probability that an English translation e is derived from a French sentence f) directly, we model $P(f|e)$ (the likelihood that a translator produces the French translation f given the English sentence e). This way of doing things is called a *noisy-channel model*, a concept borrowed from information theory (Shannon (1948)). When applying the noisy-channel model to the translation task, we somewhat assume that the foreign speaker actually wanted to utter an English sentence, but through a noisy channel, things got distorted and the speaker ended up speaking a French sentence (Koehn (2009)).

Equation 1.2 implies two core components of a SMT system:

- The *language model* presented by $P(e)$ guarantees the fluency of the English translation.
- The *translation model* presented by $P(f|e)$ takes charge of finding an adequate translation regardless of its fluency.

1.2.1 Language model

As stated earlier, the role of a language model is to assess the plausibility or fluency of a sentence, for instance, an English sentence e as the above example. Let $e = (e_1, e_2, ..., e_{|e|})$ be a sequence of English words. A common practice

to estimate the joint probability $P(e)$ is to decompose the whole-sentence probability into single-word probabilities, using the Markov chain:

$$\begin{aligned} P(e) = P(e_1, e_2, ..., e_{|e|}) &= P(e_1) \times P(e_2|e_1) \times ... \times P(e_{|e|}|e_1, e_2, ..., e_{|e|-1}) \\ &= \prod_{i=1}^{|e|} P(e_i|e_{<i}) \end{aligned} \tag{1.3}$$

assuming that the distribution of a given word e_i is conditioned only on its preceding words $e_{<i} = (e_1, e_2, ..., e_{i-1})$. The major disadvantage of conditioning on all previous tokens is that the computation complexity grows exponentially with respect to the length of the sequence. To deal with this problem, which is referred to as the curse of dimensionality, the most common method for language modeling in the early days was to use $n-gram$ language models. These models condition each word's probability on the history of the most recent $(n-1)$ words, instead of considering the whole history:

$$P(e_1, e_2, ..., e_{|e|}) = \prod_{i=1}^{|e|} P(e_i|e_{i-n+1}, ..., e_{i-1}) \tag{1.4}$$

For instance, when using a $trigram$ language model, one can estimate $P(e_i|e_{i-2}, e_{i-1})$ as the following [2]:

$$P(e_i|e_{i-2}, e_{i-1}) = \frac{count(e_{i-2}, e_{i-1}, e_i)}{count(e_{i-2}, e_{i-1})} \tag{1.5}$$

with $count(e_i, ..., e_k)$ is the number of occurrences of the sequence $e_i, ..., e_k$ in the training data.

Since it is completely independent of the source language, the training of the target language model is done solely using monolingual data in the target language, which is much more common than bilingual data. In practice, with the help of neural networks, much more complex methods are used for language modeling, for example, Neural Probabilistic Language Model (Bengio et al. (2003)), Recurrent Language Model Mikolov et al. (2010), Convolutional Language Model (Kalchbrenner et al. (2016); Gehring et al. (2017)).

1.2.2 Translation model

Early efforts in building SMT systems revolved around **word-based** methods, whose foundation was laid by the five *IBM models* (Brown et al. (1993)),

[2] In practice, however, some smoothing techniques (Chen and Goodman (1999)) are more frequently applied to estimate $P(e_i|e_{i-2}, e_{i-1})$ rather than the direct use of Equation 1.5.

which consider the translation problem as word-level alignment of the source and target sentence. These models restrict themselves to the one-to-many alignment, in which, one source word may align to several target words but not the other way around. Recall that our translation model is a noisy channel which reverses the translation direction, we consider the following example of word-alignment from English to French (where the original task is to translate from French to English):

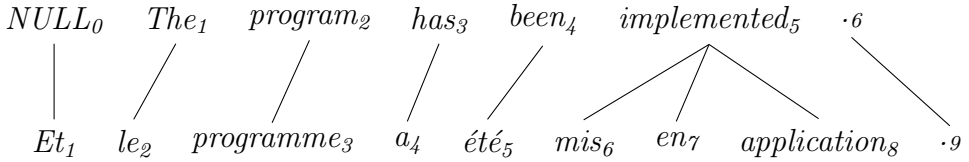


Figure 1.2: Example of word-aligment from English to French

In the above example (Figure 1.2), for French word f_j at position j, there exists an alignment a_j: $j \rightarrow i$, that connects this word with an English word at position i. In case the French word is not aligned with any English word, a special *NULL* (position $0th$ in Figure 1.2) token is introduced that is treated just like another input word. Brown et al. (1993) consider every alignment to be correct with some probability, and the job of a training process is to learn from the training data how to put higher probabilities on highly probable alignments and vice-versa. They formalize their translation models as the following:

$$P(f|e) = \sum_{a} P(f, a|e) \tag{1.6}$$

The probability of a translation is the sum over all the probabilities of all possible alignments. The way we compute the probability of an alignment ($P(f, a|e)$) depends on the IBM model being used. Brown et al. (1993) detail their five models of increasing complexity, whose advances are summarized as the following (Koehn (2009)):

- IBM Model 1: lexical translation (the translation of words in isolation);
- IBM Model 2: includes the absolute alignment model, which takes into account the word positions;
- IBM Model 3: includes the fertility model, which models the number of French words each English word is aligned to;
- IBM Model 4: includes the relative alignment model;

- IBM Model 5: fixes deficiency.

In the original work (Brown et al. (1993)), the training of these IBM models using the *Expectation Maximization (EM)* algorithm is also discussed. The following is a quick summarization of how the EM algorithm works (Koehn (2009)):

- Step 1: Initialize the model. A common practice is to initialize the model using uniform distributions.
- Step 2: Apply the model to the data (expectation step).
- Step 3: Learn the model from the data (maximization step).
- Step 4: Repeat steps 2 and 3 until convergence.

Word-based models suffer from a major flaw: the reliance on one-to-many alignments. That is to say, considering the translation model of direction English to France, the presented IBM models cannot deal with the case where multiple English words can be aligned with one French word (many-to-many alignment). One solution for this problem is to use phrases (small sequences of words), instead of words as translation units.

Phrase-based models (Koehn et al. (2003)): were born to tackle word-based translation's shortcomings. As stated earlier, instead of using words as the atomic unit of translation, phrase-based translation uses phrases (small sequences of consecutive words). Similar to word-based translation, phrase-based translation is defined as the following:

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, P(e)P(f|e) \tag{1.7}$$

Bayes rule is used to invert the translation direction and a language model $P(e)$ is also integrated. The major difference comes from the way the translation model $P(f|e)$ is decomposed:

$$P(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(start_i - end_{i-1} - 1) \tag{1.8}$$

In Equation 1.8, the English sentence e is decomposed into I phrases $\bar{e}_i$, which are translated into foreign phrases $\bar{f}_i$ using the phrase translation probability $\phi(\bar{f}_i|\bar{e}_i)$. $d(start_i - end_{i-1} - 1)$ is the relative distortion probability distribution (*distance-based reordering model*), which handles reordering. Reordering distance is measured on the foreign input side. In particular, $start_i$ is defined as the position of the first word of the foreign input phrase that

translates to the ith English phrase, and end_{i-1} denotes the position of the last word of the foreign phrase translated into the previous $(i-1)th$ English phrase. Koehn et al. (2003) propose to use an exponentially decaying cost function for estimating d, instead of estimating it from the data: $d(start_i - end_{i-1} - 1) = \alpha^{|start_i - end_{i-1} - 1|}$, with $\alpha \in [0, 1]$ is chosen so that d is a proper probability distribution. The whole idea is to penalize reordering of phrases over large distances. The state-of-the-art phrase-based translation (Koehn (2010)) puts different weights $(\lambda_\phi, \lambda_d, \lambda_{LM})$ upon the three main components of the model (the phrase translation table $\phi(\bar{f}_i|\bar{e}_i)$, the reordering model d and the language model $P(e)$). Furthermore, a log-linear model is used to improve the performance:

$$p(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x) \tag{1.9}$$

h_i is a feature (e.g., phrase model, language model, reordering model, etc.) accompanied by a weight λ_i.

1.3 Neural Machine Translation

Statistical Machine Translation had been standing as state-of-the-art in Machine Translation for quite some time, until new methods categorized as **"Neural Machine Translation"** appeared and progressively challenged its dominance. While SMT, with its many components, has an expensive development cost of optimizing each component separately before combining all of them together, these neural methods enjoy the advantage of being able to be trained directly *end-to-end*.

The idea of applying neural methods to MT started with the effort to use neural networks for language modeling (Schwenk et al. (2006)). In this work, a neural network is trained to project words in the vocabulary onto a continuous space. This new representation of words delivers consistent improvements in terms of BLEU scores on their test and development set. Neural networks are then gradually exploited to model other components of the traditional SMT. Early neural methods to model translation model of SMT are Schwenk (2012); Cho et al. (2014b). Whereas Schwenk (2012) uses a Feed-Forward Neural network (FNN) to compute the continuous phrase representation, Cho et al. (2014b) do the same job with Recurrent Neural Networks (RNNs). Both works use their new neural methods in order to rescore the phrase pairs in the phrase table, and use this additional score as a feature in the log-linear model.

Moving beyond the efforts to model separate components of traditional SMT translation, Sutskever et al. (2014) ambitiously train their *sequence-to-sequence networks* in an *end-to-end* fashion. This sequence-to-sequence model (also known as *encoder-decoder architecture*) consists of an *encoder*, which encodes the input sequence into a fixed-size vector, and a *decoder* (also modeled by an RNN), which predicts target words conditioned on the encoder's output (Figure 1.3). This model is trained end-to-end on parallel data, maximizing a single objective function.

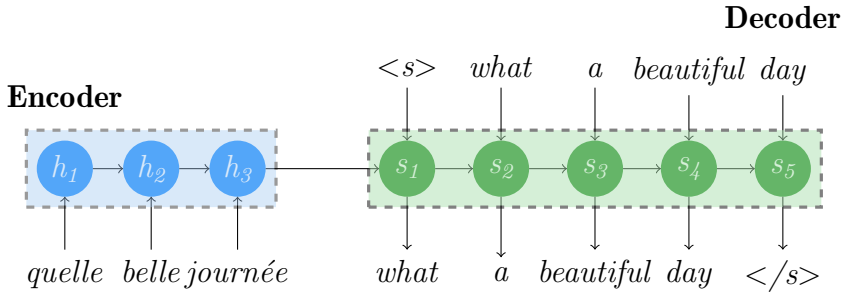


Figure 1.3: A basic sequence-to-sequence (encoder-decoder) model Sutskever et al. (2014). An RNN encoder encodes the input sequence (*"quelle belle journée"*) into a fixed-size representation, which serves as the input state of the decoder. The decoder (also an RNN) decodes the output sequence conditioned on the encoder's output representation and the previous target word until an end of sequence </s> is predicted.

1.3.1 Sequence-to-sequence modeling

To formalize this model, assuming that we have a variable-length input sequence $X = (x_1, x_2, ..., x_T)$ of length T whose ground-truth translation is a sequence $Y = (y_1, y_2, ..., y_{T'})$ of length T'.

1.3.1.1 Encoder

The encoder, parameterized by an RNN, encodes the input sequence X and outputs a sequence of abstract representation $h = (h_1, h_2, ..., h_T)$ of X:

$$h_i = encode(h_{i-1}, x_i) \tag{1.10}$$

h_i is the state of the RNN cell at time step i. The initial state h_0 can be set at random and trained with the model. $encode()$ is the transition function of the RNN. In practice, *Long Short-Term Memory (LSTM)* (Hochreiter and

Schmidhuber (1997)) or its variant *Gated Recurrent Units (GRU)* (Cho et al. (2014a)) are used for modeling this function. In this primitive form, the fixed-size last hidden state h_T is expected to fully capture useful information of the input sequence X in order to predict the output sequence $\hat{Y}$.

1.3.1.2 Decoder

The decoder, which is also an RNN, predicts a variable-length output sequence $\hat{Y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_{T''})$, with $\hat{y}_i$ chosen among all elements of a vocabulary V'. In order to do this, the decoder needs to carry the following computation:

$$s_t = decode(s_{t-1}, \hat{y}_{t-1}) \tag{1.11}$$

$$z_t = generate(s_t, \hat{y}_{t-1}) \tag{1.12}$$

$$\hat{y}_t = \underset{i \in |V'|}{\text{argmax}}\, z_{ti} \tag{1.13}$$

with s_t is the hidden state of the decoder at each time step t, which is computed by $decode()$ (an LSTM or GRU). If the encoder and the decoder have the same cell size, we can assign directly $s_0 = h_T$, otherwise, there should be a projection layer in between for matching the size of h_T with that of s_0, for instance, $s_0 = tanh(W_p h_T + b_p)$ (W_p and b_p are the parameters of the projection layer). $generate()$ takes as input the current decoder's state s_t and the previous target token $\hat{y}_{t-1}$ to calculate a vector z_t of size $|V'|$. The goal of computing z_t is to score each symbol in the target vocabulary. In practice, this can be done by using a $softmax$ function. The decoder then outputs the symbol $\hat{y}_t$, which has the highest score. At the beginning of the process, a special symbol called *beginning of sequence (<s>)* is fed in the decoder, serving as $\hat{y}_0$. During training time, when we know in advance the target length, output length T'' is set to be equal to the ground-truth length T'. In inference time, when T' is unknown, the model learns to stop decoding by producing another special symbol named *end of sequence (</s>)*. Moreover, during training, a common practice is to use a technique called *teacher forcing* (Williams and Zipser (1989)), which feeds the decoder (Equation 1.11 and 1.12) with previous ground-truth symbol y_{t-1} instead of the previous predicted output $\hat{y}_{t-1}$.

Despite being very promising, this method, as pointed by Cho et al. (2014a), does not perform well when the length of the sentence and the number of unknown words increase. The culprit is the fixed-size representation output of the encoder, which is forced to represent any arbitrarily long sequences. Bahdanau et al. (2015) fix this by introducing an *attention mechanism*, which

allows the decoder to attend at any positions of the input sequence at each time step (Figure 1.4).

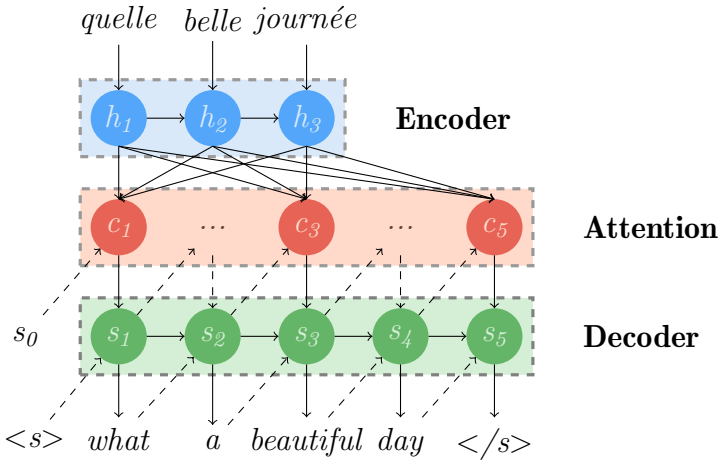


Figure 1.4: Illustration of a sequence-to-sequence model with attention. Instead of taking the last hidden state of the encoder as the initial hidden state of the decoder, the attention mechanism allows the decoder to look anywhere in the entire sequence of hidden states generated by the encoder. At each time step, a context vector is generated by the attention mechanism, summarizing the input sequence conditioned on the current hidden state of the decoder. This context vector is used to update the decoder's hidden state and to generate the target symbol.

1.3.1.3 Attention-based encoder-decoder

Attention mechanism keeps the same encoder, whereas making some significant modifications at the decoder:

$$c_t = attention(s_{t-1}, h) \tag{1.14}$$

$$s_t = decode(s_{t-1}, \hat{y}_{t-1}, c_t) \tag{1.15}$$

$$z_t = generate(s_t, \hat{y}_{t-1}, c_t) \tag{1.16}$$

$$\hat{y}_t = \underset{i \in |V'|}{\text{argmax}}\, z_{ti} \tag{1.17}$$

The significant modification is that instead of relying on a fixed-size representation h of the whole input sequence, it computes at each time step t a context c_t based on a mechanism called *Attention* (Equation 1.14). This context is then used by the decoder to calculate its current state s_t (Equation 1.15) or to generate its next output (Equation 1.16).

Attention function: there are several ways to implement the *attention()* mechanism in Equation 1.14. In their original work, Bahdanau et al. (2015) formalize this mechanism as the following:

$$c_t = \sum_{i=1}^{T} \alpha_{ti} h_i \tag{1.18}$$

$$\alpha_{ti} = softmax(e_{ti}) = \frac{\exp(e_{ti})}{\sum_{k=1}^{T} \exp(e_{tk})} \tag{1.19}$$

$$e_{ti} = v_a^\top tanh(W_a s_{t-1} + U_a h_i) \tag{1.20}$$

The weighted sum in Equation 1.18 can be understood as the *expected annotation* over all the annotations $h = (h_1, h_2, ..., h_T)$ (the annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence) with probabilities α_{ti}. These probabilities are computed by a $softmax()$ function in Equation 1.19, whose input is energy scores generated by an *alignment model*. Alignment model puts a score on the matching of inputs around position i and the output at position t. Variants of attention mechanism, for instance Luong et al. (2015a), differ from each other on how they compute the alignment model, whose original form is depicted in Equation 1.20.

1.3.1.4 Bidirectional RNN encoder

Bahdanau et al. (2015) also propose to use a *bidirectional RNN* (*BiRNN*, Schuster and Paliwal (1997)) instead of the presented *unidirectional* RNN (*UniRNN*) one (Figure 1.5). Consisting of two RNNs, BiRNN reads the input sequence from *left-to-right* (from x_1 to x_T) and calculates a sequence of forward hidden states $(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_T})$ by its forward RNN $\overrightarrow{f}$, and its backward RNN $\overleftarrow{f}$ does everything in the reversed order, giving the sequence of backward hidden states $(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_T})$. The hidden state h_j of each input word x_j is obtained by concatenating the corresponding forward state $\overrightarrow{h_j}$ and the backward state $\overleftarrow{h_j}$ (i.e, $h_j = [\overrightarrow{h_j}^\top, \overleftarrow{h_j}^\top]^\top$). Bahdanau et al. (2015) show that this better represents the word x_j by relating it to both its preceding and following words.

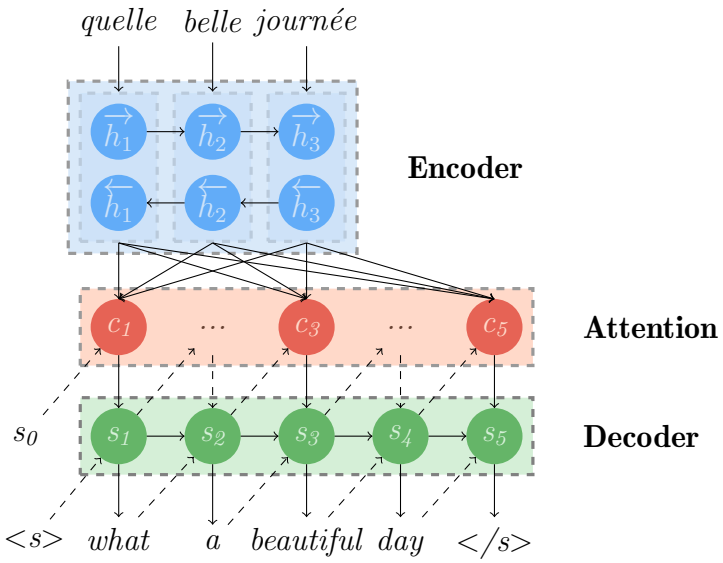


Figure 1.5: Illustration of an attentional encoder-decoder architecture with Bidirectional RNN.

1.3.1.5 Training sequence-to-sequence model

As stated earlier, one of the appealingness of the whole family of sequence-to-sequence models is that they allow the training to be done in an *end-to-end* fashion, in which all components of the model are jointly trained to optimize a single objective function. Let assume that we have a training set $\mathcal{D}$ that contains $|\mathcal{D}|$ sentence pairs (X_i, Y_i). Usually, the NMT model is trained by estimating its parameters θ using *Maximum Likelihood Estimation (MLE)*. This term θ includes all the parameters of the whole sequence-to-sequence model. Finding a set of good $\hat{\theta}$ for our problem is equivalent to:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \sum_{i=1}^{|\mathcal{D}|} \log p(Y_i|X_i;\theta) + \Omega(\theta) \tag{1.21}$$

with $\Omega()$ is a regularization function, for instance, *L2-regularization* (also known as *weight decay*): $\Omega(\theta) = \|\theta\|^2$. We can use back-propagation and gradient-based optimization to solve this problem (Equation 1.21). In practice, $\mathcal{D}$ is randomly split into smaller sets of examples, and the training iterates through these sets to gradually update the network's parameters θ.

These breakthroughs, which are fueled by the substantial increase of both computational capacity and parallel training data size, encourage more and more research diving into further improving the performance of neutral systems, which have consequently overthrown SMT to stand as the state-of-the-art in MT in the majority of settings.

1.3.2 Improvements

1.3.2.1 Out-of-vocabulary (OOV) problem

So far in this thesis, NMT models have been described as word-based models, which translate a sequence of words into another sequence of words. However, the underlying problem with such a technique is that due to computational challenges, NMT models typically restrict the vocabulary to a shortlist of several tens of thousands of most frequent words, for example, a shortlist of $30,000$ words is used in Bahdanau et al. (2015), and use an unknown token *UNK* to represent the rest. This, as shown in Bahdanau et al. (2015), makes NMT models much more sensitive to rare words, that are excluded from the shortlist, in comparison with SMT. Early NMT works, either attempt to modify the network to allow training with very large vocabularies (Jean et al. (2015a)) or, while still confining their vocabularies to the size of $30,000$ to $80,000$ most common words, attempt to identify the unknown words and replace them in post-processing (Luong et al. (2015b); Jean et al. (2015b)). Later on, a family of more effective solutions has arisen, whose representatives are *Byte Pair Encoding (BPE)* (Sennrich et al. (2016)) and its variant *SentencePiece* (Kudo and Richardson (2018)). The main idea of these approaches is to break rare words up into *subword units*, which can be anything from a single character to an entire word. A vocabulary of the most frequent subwords is constructed by a training process that iterates through the training set, starting with identifying a list of all single characters appearing in the training data. Then, the most frequent pair of characters are merged into character n-grams. This merge *operation* is repeated until a fixed number of character n-grams is reached, or when no new pair can be found. Chung et al. (2016); Luong and Manning (2016) argue that *character-level* translation is more desirable for several reasons such as (1) it is naturally immune to the OOV issue, (2) it is capable of modeling different, rare morphological variants of a word, and (3) it does not require a perfect word segmentation, which is a non-trivial problem itself. Luong and Manning (2016) propose to use hybrid systems that generally work at the word-level but switch to the character-level whenever the word-level NMT produces an *UNK*. Chung et al. (2016) advocate for using characters as the target side's units, while the source side's remains subword units. These efforts have been realized by Lee et al. (2017), who attempt to train full character-level NMT models (on both source and target side). Recent works on *character-based* translation are Kreutzer and Sokolov (2018); Cherry et al. (2018); Ataman et al. (2019), which show that character-based NMT are preferable to word-based NMT in terms of translation quality. However, they also underline that character-level NMT models are more costly to train because the character sequence is longer, which in-

curs expensive calculations on the decoder side, especially on the attention mechanism.

1.3.2.2 Beam search

During inference (decoding) time, the presented sequence-to-sequence models generate a translation sequence $\hat{Y}$, also called a *hypothesis*, one step at a time. Firstly, they encode the source sequence X and initiate the translation prefix with $\hat{y_0} = <s>$. These are inputs for estimating the output distribution of the next output token $p(\hat{y_1}|\hat{y_0}, X)$. At this step, what happens is that we sample the word that receives the highest probability, and use it as conditioning context for computing the probability distribution for the next output word. We continue sampling greedily output tokens like that until the end-of-sequence token $</s>$ is chosen or a limited number of tokens has been reached. Unfortunately, there is a problem with this so-called *greedy decoding* approach. Sometimes, the best hypothesis might not be the one that contains all the tokens with the highest probabilities. For example, when generating an idiomatic phrase like *"piece of cake"*[3], one can imagine that the first word *"piece"* might have a very low probability of being chosen. For this reason, the same as SMT, *beam search* is widely used in NMT as an improvement of greedy decoding. Beam search keeps track of b, also called the *beam width*, best candidates, also called *partial hypotheses*. When predicting the first word of the output sequence, we sample the top b words scored by their translation probabilities. After that, each of these words is used as the conditioning context for predicting the next word. Word translation probabilities are accumulated by adding the current predicted word's log-probability with the sum of log-probabilities of its preceding words (at this point the log-probability of the first word). Only b partial hypotheses that receive the best scores are kept in the beam. If the $</s>$ token is generated in a hypothesis, we consider the hypothesis to be complete, we strip it from the beam, and reduce b by 1. We repeat this process until no hypothesis remains in the beam (Figure 1.6). However, beam search favors short sequences as a negative log-probability is added with each new token, making the scores more negative for longer sentences. In order to tackle this problem, Wu et al. (2016) propose to normalize the scores of the complete hypotheses by their lengths:

$$\text{score}(\hat{Y}) = \frac{\log p(\hat{Y}|X)}{\text{lp}(\hat{Y})} \tag{1.22}$$

[3]This example, which means *"easy"*, is taken from Koehn (2020).

$$\text{lp}(\hat{Y}) = \left(\frac{\mu + |\hat{Y}|}{\mu + 1} \right)^{\alpha} \tag{1.23}$$

with μ (often set to 0) and $0 < \alpha < 1$ (referred to as the *length penalty*) are hyperparameters of the model. One can optimize α on the development set ($\alpha \in [0.6, 0.7]$ is recommended by Wu et al. (2016)).

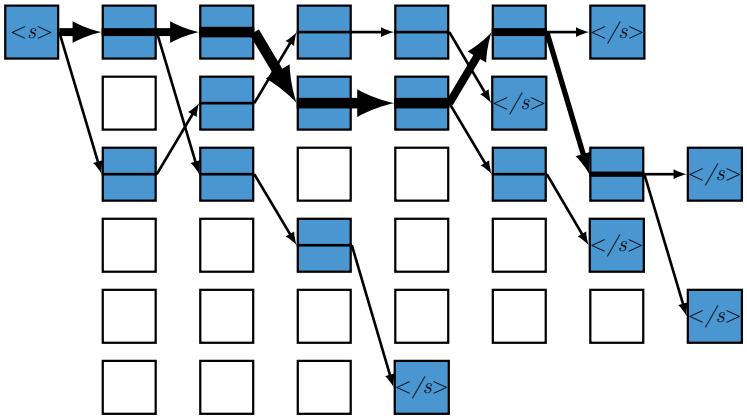


Figure 1.6: Illustration of *beam search* with beam width $b = 6$ (Koehn (2020)).

1.3.2.3 Ensemble

Ensemble is a technique in machine learning that aims to build multiple (instead of just one) systems and then combine them. When apply to NMT, this often means to train several instances of the same model, and then average their outputs (log-probability). Intuitively, this can work because different systems make different errors, and therefore, averaging them will help average out the errors of individual models. Early works, such as Sutskever et al. (2014); Luong et al. (2015b) show improvement by combining the output of individual models as the following:

$$\log p(w) = \sum_{i}^{N} \lambda_i \log p_i(w) \tag{1.24}$$

with N is the number of individual models, $p_i(w)$ is the output of model i, which is a softmax probability of a word w, and λ_i is a weight associated with this model (usually, $\lambda_i = 1/N$). Different instances of a model can be achieved either by running completely different training runs (using different random initialization, or different random shuffle of the training data), or ensembling different checkpoints of the same training run. A slightly different and rather

cheaper approach is to average all parameters element-wise of different checkpoints of the same models, and then use the obtained single averaged model for decoding (Vaswani et al. (2017)).

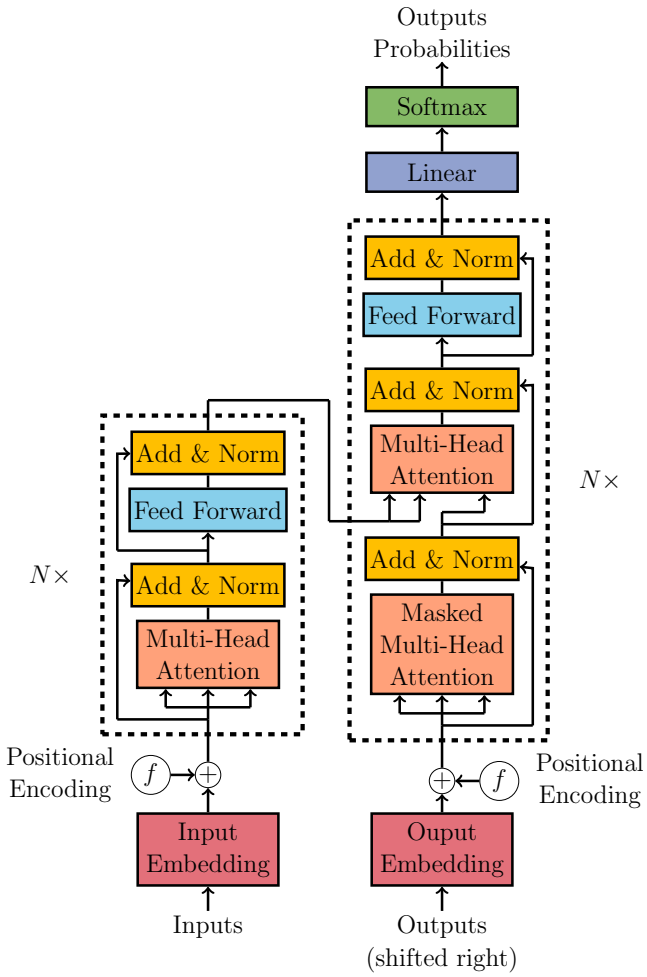


Figure 1.7: Transformer model (Vaswani et al. (2017).)

1.3.2.4 Network extension

Beside the efforts to improve decoding, the successors of Sutskever et al. (2014) and Bahdanau et al. (2015) expand their networks by going deeper (i.e, having more layers) and bigger on both encoder and decoder side. Zhou et al. (2016) stack multiple LSTM layers at both encoder and decoder side (in total 16 LTSM layers) coupling with residual connections, which give them the lead by +3 BLEU score over a rather quite deep network of the day with 6 LSTM layers proposed by Luong et al. (2015b) on WMT14 En→Fr. Wu et al. (2016) achieve +0.8 BLEU improvement over Zhou et al. (2016) with a similar sized

network, but using subword units instead of words.

Different from recurrent networks of Sutskever et al. (2014) and Bahdanau et al. (2015), Vaswani et al. (2017) propose a new architecture called *Transformer* architecture, which inherits the *encoder-decoder* concept of sequence-to-sequence modeling, but entirely replaces recurrence by an attention mechanism. This not only allows them to better parallelize their training, but also helps them achieve the state-of-the-art performance on WMT2014 En→De translation task. Figure 1.7 illustrates the original Transformer model, which consists of an encoder (on the left side) and a decoder (on the right side). The encoder is composed of a stack of N identical layers ($N = 6$ in the original proposal (Vaswani et al. (2017))). Each layer has two sub-layers: (1) a *Multi-Head Self-Attention* mechanism (Cheng et al. (2016)), and (2) a position-wise fully connected feed-forward network. The decoder has almost identical design as the encoder, except that an additional sub-layer called *Masked Multi-Head Attention* is stacked on the bottom of each layer. On both sides, *residual connection* (He et al. (2016)) is applied around each sub-layer, followed by *layer normalization* (Ba et al. (2016)). Attention plays a crucial role in Transformer models, so important that Vaswani et al. (2017) claim that *"Attention is all you need"*. Attention is referred as a mapping of a query Q (of dimension d_k) and a set of key-value pairs (K, V) (of dimension d_k and d_v respectively) to an output (of dimension d_{model}) by a *Scaled Dot-Product Attention* (Equation 1.25):

$$\text{Attention}(V, K, Q) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V \tag{1.25}$$

In fact, Vaswani et al. (2017) propose to apply Equation 1.25 multiple times. In this practice, which they name the *Multi-Head Attention* mechanism, a set of (V, K, Q) is projected h times onto h different spaces, on which the scaled dot-product attention is computed in parallel. The outputs are then concatenated and once again projected to the final layer's output (Equation 1.26):

$$\text{MultiHeadAttention}(V, K, Q) = \text{Concatenate}(head_1, ..., head_h)W^O \tag{1.26}$$

with $head_i = Attention(W_i^Q Q, W_i^K K, W_i^V V)$ is the computation of attention head i-th. $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$,$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are projection matrices. Vaswani et al. (2017) obtain 28.4 BLEU on the WMT2014 En→De translation task by using 8 attention heads ($h = 8$), and for each head, they set $d_k = d_v = d_{model}/h = 64$.

Apart from RNN and Transformer architectures, Kalchbrenner et al. (2016); Gehring et al. (2017); Elbayad et al. (2018) propose to fully use convolutional

neural networks, whose processing speed is so much faster than RNN-based approaches, for translation models. While the *ByteNet* proposed by Kalchbrenner et al. (2016) catches up on its RNN-based counterparts' performance in terms of BLEU scores, Gehring et al. (2017)'s *ConvS2S* and Elbayad et al. (2018)'s *Pervasive Attention* model even beat RNN-based models on the same WMT2014 En→De translation task.

1.4 Evaluation

Evaluation is as important for Machine Translation as for any other development tasks. *"How good the machine translation system is?"*, we might well ask ourselves when confronted with one. This question is of great importance not only during the testing time but also during the development process when one needs to decide which model one should choose. Due to the inherent ambiguity of the translation task (there is no absolute translation to one sentence, in other words, there might exist many reasonable translations to one sentence), this simple question is harder to answer than it might seem.

In Koehn (2009), Philipp Koehn lays out several goals an evaluation metric should have, which include:

- *low-cost*: the evaluation of a system should be done as quickly and cheaply as possible.
- *tunable*: the metric should be able to be used directly for system optimization.
- *meaningful*: the metric should help answer the question *"How good the machine translation system is?"*
- *consistent/stable*: the metric should allow the evaluation on one part of the test corpus to be consistent with that on another part.
- *inter-annotator agreement*: the same conclusions should be drawn by different evaluators using the same metric.
- *correct judgment*: the metric should come up with a correct judgment about the translation.

1.4.1 Human judgment

Taking into account that human beings are main users of translation products, using human judgment (sometimes referred as Manual Evaluation) as

an evaluation metric is a natural option. Judgments about the quality of the translation system can be gathered either from professional evaluators (translators or linguists) or from crowd-source platforms. Judgments can be given out in different formats:

Direct assessment: evaluators score a single translated sentence at a time based on a given scale. A rather common practice is to ask evaluators to judge the *Adequacy* and *Fluency* of each translated sentence based on a discrete scale from 1 to 5.

Ranking: human judges are asked to rank translation candidates of two or more translation systems on a ranking, instead of giving absolute adequacy or fluency scores, for example, they could usually find themselves answer the question *"Is the output translation of system A better than that of B, or worse, or indistinguishable?"*

Human Translation Edit Rate (Post-editing effort): this evaluation method measures the efforts (time or other criteria) that a human evaluator needs to take in order to post-edit a translation candidate into a correct translation.

These methods are suitable for comparing several models in-house or different models in evaluation campaigns such as the WMT (Workshop on Statistical Machine Translation) or the IWSLT (International Workshop on Spoken Language Translation).

1.4.2 Automatic metrics

Human judgment suffers from one major disadvantage: it is not *low-cost* in terms of both time and money as translators need to be paid a substantial amount of money for a great deal of time they spend manually judging the translation output sentence by sentence. For this reason, we would like to automate the evaluation process ideally by using computer programs to know quickly whether our system shows improvements after a change or not. Automatic machine translation metrics have been developed to serve this objective. In this section, we review several automatic metrics that are commonly used in practice, with a major press on BLEU score, which, in the context of this thesis, is used as the main evaluation metric for assessing the quality of the translated text.

BLEU (Papineni et al. (2002)): is arguably the most commonly used evaluation metric for MT. It is centered on the idea that *"The closer a machine translation is to a professional human translation, the better it is"* (Papineni et al. (2002)). Here, in order to assess the quality of a machine translation, they propose to measure its closeness to one or more reference human translations according to a numerical metric. In detail, they take the geometric

average of the modified n-gram [4] precision scores (of the whole test corpus) and then multiply the result by an exponential brevity penalty factor.

At the sentence-level, the modified n-gram precision is counted by firstly computing the maximum number of correct n-gram (the times an n-gram appears in any single reference translation). This number is then clipped by the corresponding n-gram maximum reference count:

$$Count_{clip} = \min(Count, Max\_ref\_Count) \tag{1.27}$$

Finally, we add these clipped counts up, and divide by the total (unclipped) number of candidate n-gram in the hypothesis. At corpus-level, this computation is done as the following:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram' \in C'} Count(n-gram')} \tag{1.28}$$

$Count_{clip}$ operation helps punish very long hypotheses which create some n-grams repeatedly. A good thing about BLEU is that it also penalizes very short sentences by applying sentence brevity penalty:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases} \tag{1.29}$$

with c and r are the total length of the candidate translation corpus, and the test corpus' effective reference length respectively. The final BLEU metric is computed as geometric average of the modified n-gram precisions, p_n, using n-grams up to length N and positive weights w_n summing up to 1.

$$BLEU = BP \times exp \sum_{n=1}^{N} w_n \log p_n \tag{1.30}$$

A common practice is to choose $N = 4$, and $w_n = 1/N$.

BLEU has been the most favorable evaluation metric in most translation tasks due to its attractiveness including the use of multiple reference translations. Moreover, its geometric mean of the modified n-gram precision not only counts the number of correct words but also rewards correct word order, while the brevity penalty penalizes too short translation. However, some critical points of BLEU can be found in Koehn (2009), for example, the score itself does not have an intuitive interpretation (i.e, nobody knows what a BLEU score of 20% means), etc.

[4] An n-gram is a contiguous sequence of n tokens, with a token is anything delimited by whitespaces, after tokenization, for example, a word, a punctuation symbol, etc.

METEOR (Banerjee and Lavie (2005)): is a more recent metric than BLEU, that computes a score for unigram matching by combining both unigram-precision and unigram-recall. Moreover, this metric is designed to counter several BLEU's drawbacks. By incorporating the use of stemming, METEOR allows different forms of the same word to be matched, while BLEU would punish them strongly. For example, the noun *attractiveness* and the adjective *attractive* carry the same meaning, but are considered mismatched by BLEU. Furthermore, METEOR also incorporates the use of synonyms, allowing, for example, *appealing* and *attractive* to be equally considered. Although METEOR correlates better with human judgment than BLEU, its critical weakness is that it is computed in a much more complicated manner. Moreover, it requires access to linguistic resources such as morphological stemmers and synonym databases that are not always available for every language.

Word Error Rate (WER): is borrowed from Automatic Speech Recognition (ASR). This metric is derived from the Levenshtein distance (Levenshtein et al. (1966)), which is defined as the minimum number of editing steps *insertions/deletions/substitutions* (at word level) needed to match two sequences. Even though WER is one of the most commonly used metrics in ASR, where the word order of the hypothesis is expected to be strictly matched that of the reference, when applied to MT, this metric shows its disadvantages. The one big problem is that with translation, one can have two perfectly fine translated sentences with completely different word orders. This metric, therefore, penalizes severely the translation that should not be penalized.

Translation Edit Rate (TER) (Snover et al. (2006)): measures the minimum amount of editing that a human would have to perform to match the translation output with the reference. This fixes the aforementioned drawback of WER by introducing a *shift* operation to the combination of *addition/deletion/substitution* of words. With this additional *shift* operation in hand, TER allows to move a contiguous sequence of words to another location within the same hypothesis. This edition of several words has the cost as other operations (*addition/deletion/substitution*) which can be done only on a single word basis. Snover et al. (2006) show that TER not only correlates reasonably well with human judgment but it is also more intuitive than BLEU (i.e., the score indicates the amount of work needed to correct the translations). However, this also comes with a price: TER computation is NP-complete (Shapira and Storer (2002)). Snover et al. (2006) propose an approximation method to compute TER which, unfortunately, does not guarantee to find the optimal match and is still fairly expensive to be carried out.

Besides, we also have more recently proposed automatic evaluation metrics such as *BERTScore* (Zhang* et al. (2020)) which computes a similarity score between each token in the hypothesis sentence with each token in the

reference sentence. However, instead of exact string matching (e.g., in BLEU) or heuristics matching (e.g., in METEOR), they compute the similarity between contextualized token embeddings (BERT) of the hypothesis and the reference. Their reasons are: (1) contextualized embeddings are more robust to paraphrase matching and (2) they capture more effectively distant dependencies and ordering. Another recent automatic metric for MT evaluation is *COMET* (Rei et al. (2020)), which is a learned framework trained to optimize the correlation with human judgments or to minimize the distance between the "better" hypothesis and the anchors (source or reference) on the embedding space.

1.5 Conclusion

In this chapter, we briefly present traditional methods for machine translation including Rule-based Machine Translation, which is based essentially on linguistic rules, and Statistical Machine Translation, which exploits statistical methods without caring about linguistic rules. Statistical Machine Translation, which is composed of a language model and a translation model, had been standing as state-of-the-art in the field for decades, until the emergence of Neural Machine Translation, which is presented in a greater volume of this chapter. The dominant method in Neural Machine Translation is end-to-end sequence-to-sequence modeling. The outstanding representative of this kind of models consists of an encoder and a decoder, which are bridged by an attention mechanism. Beside discussing several network extensions for Neural Machine Translation, for instance, Bidirectional Long Short-Term Memory encoders, and the Transformer architecture, we also review different approaches for improving the quality of the automatic translation, for example, the use of beam search, ensemble, and subword units. Translation quality is most often measured by BLEU, an automatic metric, which is presented in this chapter along with METEOR, Word Error Rate, and Translation Edit Rate.

Chapter 2

Neural Speech Translation

2.1 Definition

Automatic Speech Translation (AST), or sometimes addressed by the term *Spoken Language Translation (SLT)*, refers to automatic processes of translating speech from one language into either speech or text of another. This thesis focuses on the latter, investigating *Speech-to-Text (S2T)* translation.

Early attempts in AST dated back to the 1980s when NEC performed the first proof of concept at the 1983 ITU Telecom World (Nakamura (2009)). Research interest in the field continued raising in the 1990s when ASR proved itself promising. As time goes on, this field flourishes, and constraints in terms of domain, vocabulary, speaking style, etc. have been gradually loosened up (Niehues et al. (2021)). For the first several decades of development, AST systems had been implemented as cascades of ASR systems followed by MT systems (Figure 2.1). AST, therefore, has been considered a much harder task than text translation as it requires solving problems that come from both sides, ASR and MT, which are far from being solved. Furthermore, as speech signal does not explicitly contains syntactic and semantic clues of written language, for instance, paragraph and sentence delimiters, punctuation marks, and capitalized words, etc., it piles up difficulties for AST to solve. For these reasons, until recently, progress in ST had been focusing either on improving ASR or MT models, or on optimizing the coupling of ASR and MT. However, since 2016, with the introduction of end-to-end methods (Duong et al. (2016); Bérard et al. (2016); Weiss et al. (2017); Bérard et al. (2018)), research interest in AST has rapidly navigated from the aforementioned two fronts of improvements to improving a single model: the end-to-end AST model (Figure 2.2).

We discuss different important aspects of AST in this chapter, reviewing both cascaded systems and end-to-end systems. Before doing so, let us formalize the AST task by assuming that our AST system takes as input speech feature sequence $X = (x_1, x_2, ..., x_T)$, and produces the best translation $\hat{Y} = (\hat{y_1}, \hat{y_2}, ..., \hat{y_S}) \in \mathcal{T}$ from the MT hypothesis space. Speech features sequence X can be *Mel Frequency Cepstral Coefficient (MFCC)*, *Log Mel Filterbank (MFB)*, or *Self-Supervised Learning features*, etc. (details of these features are discussed in Chapter 5), while output Y can be a sequence of any

subword units (e.g, characters, BPEs, etc.) similar to NMT's outputs. In case where an ASR system is needed, let $S \in \mathcal{H}$ be a possible transcript from the ASR hypothesis space. The speech translation problem is formalized as the following:

$$\hat{Y} = \underset{Y \in \mathcal{T}}{\operatorname{argmax}} P(Y|X) \tag{2.1}$$

Depending on the methods being used, cascaded or end-to-end models, Equation 2.1 is decomposed differently.

2.2 Cascaded systems

As mentioned earlier, before the booming of end-to-end approaches, the only feasible solution for implementing an AST system had been to stack an ASR system that processes the input speech signal to produce *intermediate representations*, before an MT system that absorbs the ASR's output for emitting the final textual translation (Figure 2.1). In early AST systems, where ASR and MT systems are trained independently, intermediate representations are source language transcription. These systems, which are referred to as the *loosely coupled cascade* by Sperber and Paulik (2020), decompose Equation 2.1 as in Equation 2.2, 2.3 and 2.4. More recent systems might use different intermediate representations, for example, the hidden states of the ASR's decoder.

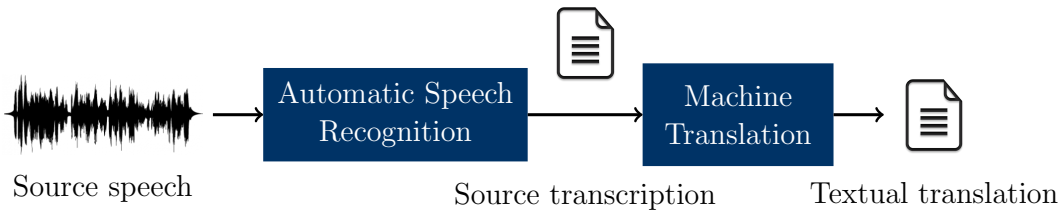


Figure 2.1: A traditional cascaded AST system.

$$\hat{Y} = \underset{Y \in \mathcal{T}}{\operatorname{argmax}} \sum_{S \in \mathcal{H}} P(Y|S, X) P(S|X) \tag{2.2}$$

$$\approx \underset{Y \in \mathcal{T}}{\operatorname{argmax}} \sum_{S \in \mathcal{H}} P_{MT}(Y|S) P_{ASR}(S|X) \tag{2.3}$$

$$\approx \underset{Y \in \mathcal{T}}{\operatorname{argmax}} \sum_{S \in \mathcal{H}'} P_{MT}(Y|S) P_{ASR}(S|X) \tag{2.4}$$

with $\mathcal{H}'$ contains only one single entry, the *1-best* ASR output.

Several advantages of cascaded AST systems are discussed in Niehues (2019). The most important of all is that data for training such cascaded models seems to be more abundant than that for training end-to-end systems. This comes from the fact that (1) both research fields that fuel the development of cascaded AST have longer histories of development, through which training data for both ASR and MT have gradually been collected; and (2) from the nature of the data itself, ASR and MT corpora are easier and cheaper to build. Parallel data of translated text aligned with speech signals, which is needed for training end-to-end models, on the other hand, is quite scarce.

The advantages of cascaded systems seem to be outweighed by a wide range of essential problems against which so much effort has been devoted to tackling. The first and arguably the most crucial drawback of cascaded methods is *error propagation*. Early loosely coupled cascade AST systems seem to suffer from this problem more severely. This is because ASR and MT systems are built separately and the best hypothesis of the former is used as input to the latter, and therefore, the MT module is often unprepared for the ill-formed inputs propagated from the ASR module. One of the solutions for this problem is to use the *N-best* translation approach (Woszczyna et al. (1993); Lavie et al. (1996)) which uses a list of the N best ASR's hypotheses instead of *1-best* ASR's output (i.e, increasing $\mathcal{H}'$ in Equation 2.4) to select and analyze the most plausible sentence hypothesis for producing an accurate and meaningful translation. Alternatives to *N-best* list are *word lattices* (Schultz et al. (2004); Zhang et al. (2005); Matusov et al. (2005)) and *confusion nets* (Bertoldi and Federico (2005)), which also attempt to build a *tighter coupling* of more closely interacting ASR and MT systems. Apart from marginalizing ASR's outputs, another idea to mitigate error propagation is augmenting MT's training data by injecting synthetic ASR errors in order to train a more robust MT system (Tsvetkov et al. (2014); Ruiz et al. (2015); Sperber et al. (2017)). Another critical disadvantage of cascaded systems comes from the mismatch between the data training types of ASR and MT systems, particularly, ASR often outputs and, therefore, is trained on unpunctuated transcripts, while punctuation is much needed for translation. Solutions for this problem include additional modules for segmenting ASR's output, predicting and inserting punctuation (Matusov et al. (2006)) to improve MT text inputs, disfluency removal (Fitzgerald et al. (2009)) to avoid translation errors caused by disfluencies rooted by spoken language. Moreover, Sperber and Paulik (2020) also point out that cascaded systems suffer from the *information loss* problem. Particularly, the input text of the MT system does not convey some important characteristics of the speech signal, for example, *prosody*, which might be crucially important for conditioning a good translation. Solutions are yet to continue stacking up additional compo-

nents that compensate for the lost information (Aguero et al. (2006)). Piling up additional components means escalating development cost, which is arguably another critical weakness of cascaded AST. Besides, training cascaded models also requires source transcriptions, which might be abundant in some languages, but scarce, or worse, not available at all in others, especially in low-resource languages.

2.3 End-to-end Automatic Speech Translation

End-to-end Automatic Speech Translation models, as defined by the IWSLT 2019 evaluation campaign (Jan et al. (2019)), are models: (1) trained without exploiting intermediate discrete representations (e.g., source language transcription or hypotheses fusion in the target language) [1], and (2) whose parameters that are used during decoding must be all jointly trained on the end-to-end task. Duong et al. (2016); Bérard et al. (2016); Weiss et al. (2017); Bérard et al. (2018) are pioneers for end-to-end methods, promising to get rid of Equation 2.2-2.4 completely.

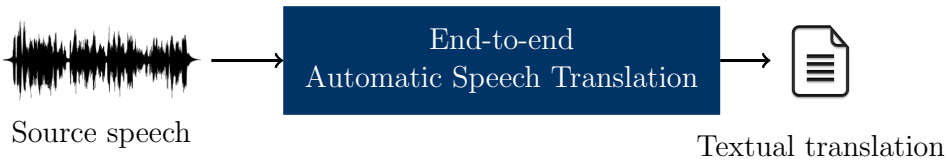


Figure 2.2: An end-to-end AST model.

The development of such systems is owed to the fact that end-to-end sequence-to-sequence models have been proven effective for MT, and ASR task also achieves promising results when exploiting this kind of models (Chorowski et al. (2015); Chan et al. (2016); Zhang et al. (2017); Chorowski and Jaitly (2016)). More importantly, the efforts to make speech translation corpora (i.e, parallel data of recorded speech coupled with translation text, details can be found in Section 2.4) available directly encourage the training AST models without using source transcription.

The first attempt to use end-to-end models for translation is made by Duong et al. (2016). However, this work only focuses on the alignment between source speech and its corresponding text translation without proposing a complete end-to-end translation system. The first end-to-end speech-to-text

[1]This definition might not be relevant in some cases, for instance, when the encoder is pre-trained on an ASR task, which, as we shall see, is a fairly common strategy for developing an end-to-end AST system.

translation system is trained by Bérard et al. (2016) on a synthetic (Text-To-Speech) speech corpus rather than the real parallel speech translation corpus. Real end-to-end AST models built on real speech translation corpora are proposed by Weiss et al. (2017) and Bérard et al. (2018).

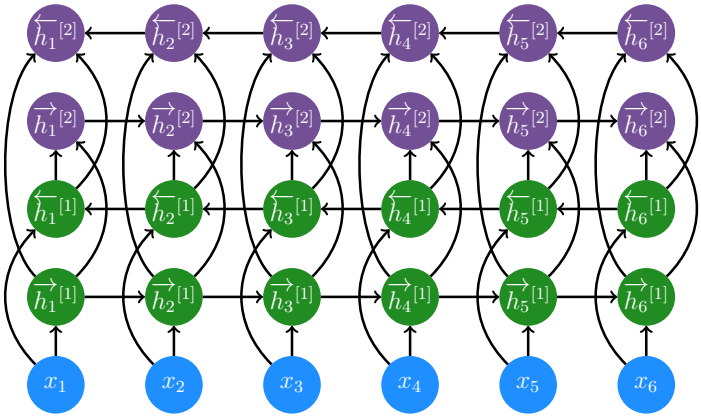
2.3.1 Speech encoder

Inspired by ASR's sequence-to-sequence models, AST pioneers (Duong et al. (2016); Bérard et al. (2016); Weiss et al. (2017); Bérard et al. (2018)) realize the need to adapt the encoder for working directly on speech signals. The reason is that input speech signals can be hundreds to thousands of frames long, so much longer than textual input sequences. This directly burdens the attention mechanism because its complexity is linear with the length of the input sequence. Chan et al. (2016) find that their attention-based decoder struggles to extract relevant information from a large number of input time steps. This takes a great deal of time for their training to converge to inferior results in comparison with their proposed adaptation. Bahdanau et al. (2016) observe that output representations of their encoder before adaptation are overly precise and contain much redundant information. For these reasons, modifications to the speech encoder have been made firstly with ASR, and consequently being inherited by AST.

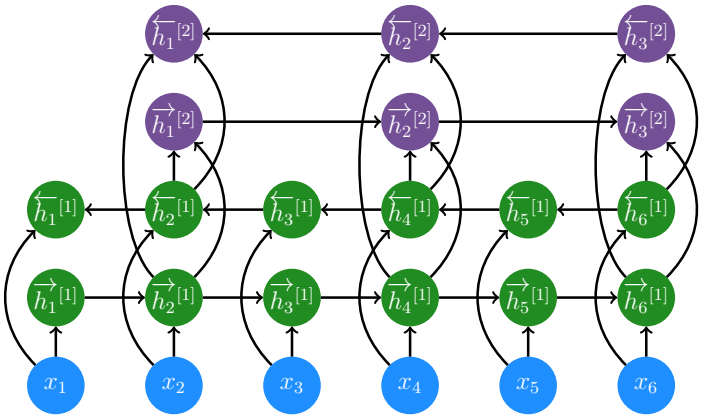
Pyramidal RNN speech encoder: is Chan et al. (2016); Bahdanau et al. (2016) independently propose to use pyramidal RNN speech encoders with minor differences. Graves et al. (2013) prove that stacking multiple layers of RNNs on top of each other improves ASR results. However, the standard deep stack of RNNs outputs a sequence of the same length as its input (Figure 2.3(a)). Chan et al. (2016); Bahdanau et al. (2016) find this suboptimal for the reasons mentioned above. Chan et al. (2016) propose to concatenate the outputs of consecutive steps of each BiRNN layer before feeding to the next layer. This helps reduce the time resolution of the representation by a factor of 2 after each BiRNN layer, and transforms Equation 2.5 of the standard deep BiRNNs to Equation 2.6 formalizing pyramidal BiRNNs (with i, j denote the time step and the layer number, respectively). Bahdanau et al. (2016) independently propose a quite similar architecture, putting pooling operations between BiRNN layers as shown in Figure 2.3(b). Duong et al. (2016) are the first to exploit this idea for the translation task.

$$h_i^j = \text{BiRNN}(h_{i-1}^j, h_i^{j-1}) \tag{2.5}$$

$$h_i^j = \text{pBiRNN}(h_{i-1}^j, [h_{2i}^{j-1}, h_{2i+1}^{j-1}]) \tag{2.6}$$



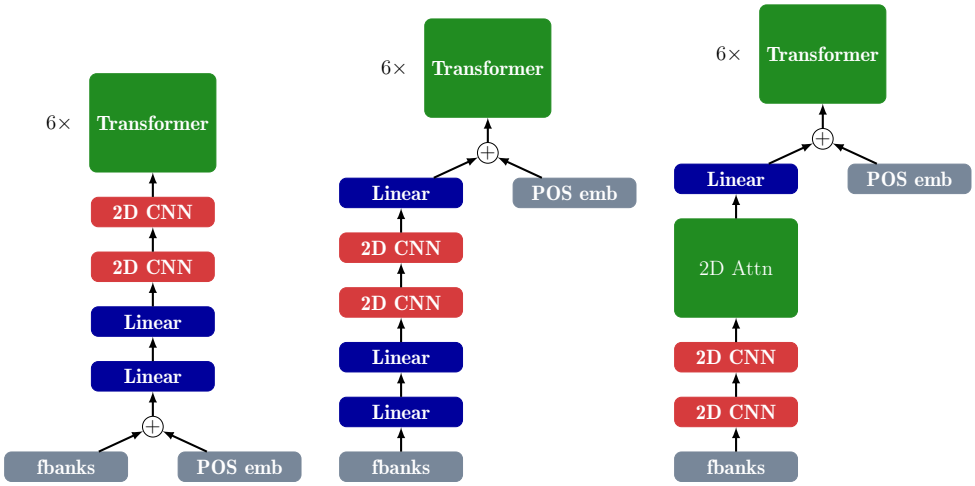
(a) Standard deep BiRNNs, drawn by Bahdanau et al. (2016).



(b) Pyramidal BiRNN proposed by Bahdanau et al. (2016).

Figure 2.3: Pyramidal BiRNNs versus standard deep BiRNNs.

Stacking CNN layers speech encoder: sharing the same purpose of reducing the length of the speech representation, stacking CNNs layers in the speech encoder is another fairly common practice (Bérard et al. (2016); Weiss et al. (2017); Bérard et al. (2018)). *Convolutional neural networks* (LeCun et al. (1989)), dubbed by *CNNs*, have been shown to effectively represent acoustic signals, because they reduce spectral variations and model spectral correlations in signals (Sainath et al. (2013b,a)). Zhang et al. (2017) introduce CNN layers everywhere in their sequence-to-sequence ASR models, adding them at the bottom of the speech encoder (before LSTM layers), and also in between LSTM layers. Weiss et al. (2017); Bérard et al. (2018) adapt this idea of adding CNN layers before LSTM layers to their AST speech encoders, stacking two layers of CNN before multiple Bidirectional LSTM (BLSTM) layers. Both of these works use a stride of $(2, 2)$ for each CNN, which, in effect, helps reduce both the time and feature dimension of the input by the factor of 2.



(a) B-Transformer (b) R-Transformer (c) S-Transformer

Figure 2.4: Different Transformer-based speech encoders proposed by Di Gangi et al. (2019b). In these architectures, *2D CNN* layers are stacked at different positions, but always before Transformer layers. *fbank, POS emb, linear, 2D Attn* represent input speech features, positional embedding layer, linear layer, and 2D attention layer respectively.

Transformer speech encoder: as stated in Chapter 1, Transformer has become a new state-of-the-art in NMT for several language pairs, and has consequently become a very trending MT architecture. Not only giving better translation quality in some MT settings, this architecture is faster than their RNN-based counterparts, because it can parallelize computation along all the time steps. For these reasons, Transformer is soon applied for ASR tasks (Zhou et al. (2018a,b); Dong et al. (2018). While Zhou et al. (2018a,b) only add a linear transformation with a layer normalization to the encoder to transform the log-Mel filterbank feature to the model dimension d_{model}, Di Gangi et al. (2019b) argue that this minor modification is far from enough for Transformer to work well on speech signals. They discuss several challenges for this, including: (1) speech input is, again, much longer than textual input, directly adding extra complexity to the already complex computation of Transformer models; and (2) the bidimensional dependencies along time and frequency of speech features make them more difficult to handle. Therefore, Di Gangi et al. (2019b) rather adapt the Speech-Transformer proposed by Dong et al. (2018), stacking a few $2D$ CNN layers at different positions before Transformer layers of the speech encoder (Figure 2.4).

2.3.2 End-to-end Automatic Speech Translation models

With the speech encoders being introduced, we can now talk about the full end-to-end AST model. Most end-to-end AST models so far are variants of the backbone *LAS (Listen, Attend, and Spell)* model proposed by Chan et al. (2016). Although it is referred to as a different term (LAS), this model is technically an attention-based encoder-decoder model. As illustrated in Figure 2.5, this model consists of a *Listener*, whose job is equivalent to the encoder's of Bahdanau et al. (2015)'s MT model, transforming the input speech feature sequence X into a higher level representation $h = (h_1, h_2, ..., h_U)$ with $U \leq T$ (Equation 2.7). In the original LAS model, Listener is modeled by a pyramidal BLSTM formalized by Equation 2.6. Another important component of LAS model is *Speller* (the upper part of Figure 2.5), which is technically an attention-based decoder. The core function of Speller is *AttendAndSpell*, which consumes h to calculate a probability distribution over target token sequence (Equation 2.8).

$$h = \text{Listen}(X) \tag{2.7}$$

$$P(Y|X) = \text{AttendAndSpell}(h, Y) \tag{2.8}$$

The AttendAndSpell() version of Chan et al. (2016) is an attention-based LSTM decoder, which computes at each decoder's step i the probability distribution over the next output character conditioned on all the previous characters as the following:

$$c_i = \text{AttentionContext}(s_i, h) \tag{2.9}$$

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \tag{2.10}$$

$$P(y_i|X, y_{<i}) = \text{CharacterDistribution}(s_i, c_i) \tag{2.11}$$

with AttentionContext() is an attention mechanism calculating a context vector c_i, RNN() is modelized by 2 LSTM layers, and CharacterDistribution() is a *Multilayer Perceptron (MLP)* with softmax over output characters.

In practice, despite borrowing the concept of the LAS model, successors of Chan et al. (2016) can adapt any kind of speech encoders, some of which are mentioned in Section 2.3.1. The Speller side can also be varied with a wide range of choices from the types of RNN (LSTM, GRU, etc.), the depth of RNN layers can be deeper or shallower, different kinds of attention mechanisms can be considered, different output granularity (characters or subword units, etc.), additional components can be utilized, or even Transformer-based decoder can replace the RNN-based one, etc.

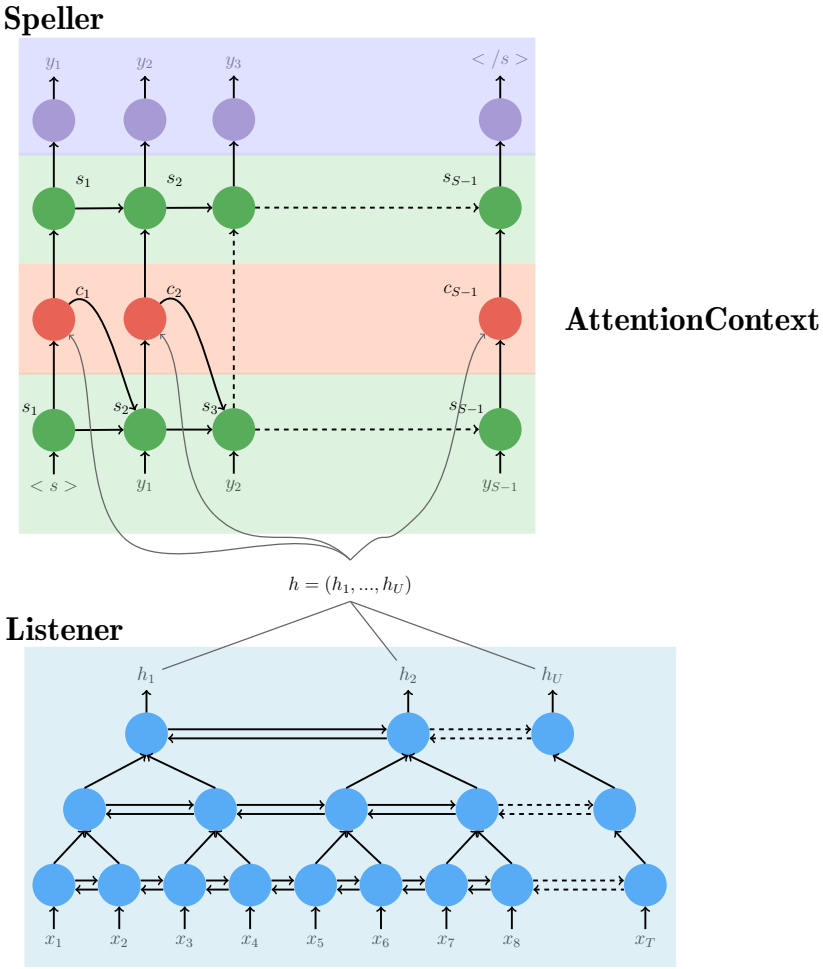


Figure 2.5: *Listen, Attend and Spell (LAS)* model (Chan et al. (2016)). The source input sequence X is encoded into shorter sequence h by the pyramidal BLSTM *Listener*. After this, *AttentionContext* creates context vector c_i from h and s_i and *Speller* predicts output token y_i.

2.3.3 Challenges for end-to-end models

End-to-end AST has sought to tackle the drawbacks of the traditional cascaded methods mentioned in Section 2.2. However, it has to face great challenges too. In fact, until recently, most end-to-end models are behind cascaded models in terms of performance. End-to-end AST is considered a much more difficult task than the two tasks ASR and MT, whose challenges seem to be inherited and combined by end-to-end AST models. Particularly, AST models have to map audio features into text in a different target language, which might have different ordering and or ambiguous meaning, etc.

Besides the aforementioned challenge that comes from the differences between speech signal and textual input requiring adaptations mentioned in Section 2.3.1, *data availability* is another well-known challenge of end-to-end methods for AST. Parallel corpora of speech aligned with translation text which are required for training AST models end-to-end are not as abundantly available as data for training ASR or MT models. When such data is available, its amount is rather modest in comparison with ASR or MT's data. There have been several solutions to this problem:

- **Data augmentation**: are techniques that can be used for augmenting AST training data. For example, combining various AST corpora of the same language pair (Nguyen et al. (2019); Elbayad et al. (2020b)) or using weakly supervised (synthetic) data. Synthetic data can either come from augmenting an ASR corpus with automatic translations (Bérard et al. (2016); Bérard et al. (2018); Jia et al. (2019); Pino et al. (2019); Elbayad et al. (2020b)), or augmenting an MT corpus with synthesized speech Jia et al. (2019); Pino et al. (2019). Using synthetic data is shown by Jia et al. (2019) to be more data-efficient than multi-task training when large MT and ASR corpora can be augmented. Pino et al. (2019) show that augmenting ASR data is more effective than augmenting MT data. Data augmentation can also be done by some tricks that operate directly on the speech signals, for instance, speed pertubation as in Nguyen et al. (2019); Elbayad et al. (2020b) or SpecAugment (Park et al. (2019)) that operates on spectral features (Bahar et al. (2019); Elbayad et al. (2020b)).

- **Multi-task learning**: as defined by Sperber and Paulik (2020), consists of techniques that pair either model inputs or outputs with data from an arbitrary auxiliary task through multi-task training. Some works, for example, Weiss et al. (2017); Bérard et al. (2018), incorporate ASR and MT data into direct models by using auxiliary models sharing parts of the parameters with the main model. The main model and auxiliary models are trained jointly, but auxiliary models are discarded in inference time.

- **Pre-training**: is usually done in AST by pre-training either the encoder or decoder (Bérard et al. (2018); Bansal et al. (2019)) in ASR or MT tasks, where the training data is more abundant. These pre-trained components are then used to initialize the corresponding components of the AST model. Pre-training can also be done as part of the feature extraction process, where latent representations of either speech (Section 5.3) or text (Devlin et al. (2018)) are learned from a huge amount

of unlabelled data.

2.4 Automatic Speech Translation Corpora

The development of AST, especially end-to-end AST is owed to the creation of speech corpora which pair speech utterances with their corresponding translation text. Table 2.1 lists out several publicly available AST corpora [2]. In the context of this thesis, three main AST corpora are used including:

- **MuST-C** [3] (Di Gangi et al. (2019a)): is a large multilingual corpus for AST from English into 8 different languages including German, Spanish, French, Italian, Dutch, Portuguese, Romanian, and Russian. This corpus of $\sim$ 400 hours of speech data, which is derived from English TED Talks, was introduced as one of the main corpora for the speech translation track of the IWSLT 2019 Evaluation Campaign.

- **How2** [4] (Sanabria et al. (2018)): is a large multimodal corpus of instructional videos paired with spoken utterances, English subtitles and their crowdsourced Portuguese translations, as well as English video summaries. This corpus is therefore suitable for various NLP tasks including ASR, MT, AST and Summarization. The data part that is suitable for training end-to-end AST contains $\sim$ 300 hours of speech. Along with MuST-C, this corpus was introduced as one of the main corpora for the speech translation track of the IWSLT 2019.

- **EuroParl-ST** [5] (Iranzo-Sánchez et al. (2020)): is a multilingual AST corpus containing speech from debates held in the European Parliament in the period between 2008 and 2012. In the initial release of this corpus, the speech utterances from each of the 6 supported European languages (English, German, French, Spanish, Italian and Portuguese) are paired with translations from the other 5 languages of this same combination, making up a total of 30 different translation directions. This corpus was introduced in the IWSLT 2020 Evaluation Campaign.

[2] Some statistics is taken from `https://github.com/kahne/SpeechTransProgress`.
[3] `https://ict.fbk.eu/must-c/`
[4] `https://github.com/srvk/how2-dataset`
[5] `https://www.mllp.upv.es/europarl-st/`

No.	Corpus	Direction	Size (in hours)	Speech type
1	**CoVoST 2** (Wang et al. (2020b))	{Fr, De, Es, Ca, It, Ru, Zh, Pt, Fa, Et, Mn, Nl, Tr, Ar, Sv, Lv, Sl, Ta, Ja, Id, Cy}→En and En→{De, Ca, Zh, Fa, Et, Mn, Tr, Ar, Sv, Lv, Sl, Ta, Ja, Id, Cy}	2880	Read, CommonVoice
2	**mTEDx** (Salesky et al. (2021))	{Es, Fr, Pt, It, Ru, El}→En,{Fr, Pt, It}→Es, Es→{Fr, It},{Es,Fr}→Pt	765	TED talks
3	**CoVoST** (Wang et al. (2020a))	{Fr, De, Nl, Ru, Es, It, Tr, Fa, Sv, Mn, Zh}→En	700	Read, CommonVoice
4	**MuST-C** (Di Gangi et al. (2019a))	En→{De,Es,Fr,It,Nl,Pt,Ro,Ru}	504	TED talks
5	**How2** (Sanabria et al. (2018))	En→Pt	300	Instructional videos
6	**Augmented LibriSpeech** (Kocabiyikoglu et al. (2018))	En→Fr	236	Read audiobooks
7	**EuroParl-ST** (Iranzo-Sánchez et al. (2020))	{En, Fr, De, Es, It, Pt, Pl, Ro, Nl}→{En, Fr, De, Es, It, Pt, Pl, Ro, Nl}	280	European Parliament proceedings
8	**Kosp2e** (Cho et al. (2021))	Ko→En	198	Read news, textbooks,AI agent command, Diary
9	**Fisher + Callhome** (Post et al. (2013))	Es→En	160	Phone conversations
10	**MaSS** (Boito et al. (2019))	{En, Es, Eu, Fi, Fr, Hu, Ro, Ru}→{En, Es, Eu, Fi, Fr, Hu, Ro, Ru}	172	Bible readings
11	**LibriVoxDeEn** (Beilharz et al. (2019))	En→De	110	Read audiobooks
12	**BSTC** (Zhang et al. (2021))	Zh→En	68	Simultaneous interpretation
13	**STC** (Shimizu et al. (2014))	En↔Jp	22	Simultaneous interpretation
14	**IWSLT2018** (Jan et al. (2018))	En→De	273	TED talks

Table 2.1: AST corpora.

2.5 Conclusion

In this chapter, we review cascaded methods which had been state-of-the-art for Automatic Speech Translation for a great deal of time. However, these methods, which couple an Automatic Speech Recognition system before a Machine Translation system, attract less attention recently due to the appearance of end-to-end Neural Speech Translation. This newly proposed method is more appealing because it promises to tackle the shortcomings of cascaded methods, for example, the error propagation problem and the expensive development cost. However, end-to-end models require being trained on a huge amount of speech translation data (i.e, parallel data of speech paired with corresponding translation text), which is not easy to be satisfied in most cases. Along with the introduction of new speech translation corpora, various methods including data augmentation, multi-task learning, and pre-training, have been introduced for mitigating this data availability challenge. We spend a great volume of this chapter reviewing different end-to-end models for Automatic Speech Translation, most of which are based on the Listen, Attend, and Spell (LAS) model. LAS model is basically a sequence-to-sequence model similar to those used in Machine Translation, however, the speech encoders are adapted for better dealing with input speech features.

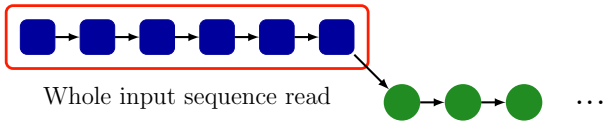
Chapter 3

Online Neural Machine Translation

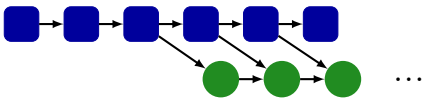
3.1 Definition

Previous chapters introduce *offline* machine translation, whose models enjoy the advantage of having the entire source sequence exposed to them at once. In *offline* translation, target tokens are produced one-by-one conditioned on the full source sequence and the previously decoded prefix (Figure 3.1(a)). *Online* (also known as *simultaneous*) machine translation, on the other hand, refers to automatic translation systems which start generating an output hypothesis before the entire input sequence has been consumed (Figure 3.1(b)). *Online* machine translation is extremely useful in real-life applications that require translation tasks to be done in real-time, for example, simultaneous interpreting. However, gathering research evidence in psychology, linguistics and interpretation, Kroll and De Groot (2005) show that these *"mental gymnastic"* tasks are *"the most complex language tasks imaginable"* even for professional interpreters. Interpreting quality significantly decreases over time because interpreters are continuously bombarded by both physical and cognitive fatigue and stress (Moser-Mercer et al. (1998)). Recognizing the enormous challenges for both machine and human interpreters, Fügen et al. (2007) suggest several advantages of using machines for the *online* translation task, such as the possibility of using adaptation techniques to increase the overall performance of the system, the capability of storing long sequences of words of the computer, and the reusability in the same domain or language, etc. This encourages the development of *online* machine translation systems. This chapter reviews the state-of-the-art of *online* machine translation, concentrating on text-to-text translation. Before doing so, let us formalize the task as the following:

Let (X, Y) be a pair of source-target sequences of lengths $|X|$ and $|Y|$ respectively. *Online* machine translation consists of executing a sequence of interleaved READ and WRITE operations, consuming tokens from the source $X = [x_1, x_2, ..., x_{|X|}]$ and producing tokens for the target $Y = [y_1, y_2, ..., y_{|Y|}]$. Let $g(t)$ be a monotonic non-decreasing function of t that denotes the number of source tokens processed by the *online* translation system in order to produce the target token y_t. *Online* machine translation systems predict y_t conditioning on the target prefix $y_{<t}$ and the partial source context $x_{\leq g(t)}$.



(a) Offline translation can wait for the whole input sequence to be made fully available before starting generating the first output token.



(b) Online translation must start generating output tokens from partial input.

Figure 3.1: Offline translation versus online translation.

3.2 Evaluation

In offline translation tasks, one only concerns with the quality of the translation output. Online translation, on the other hand, is required to make a trade-off between translation quality and latency. In practice, this trade-off is usually illustrated by a $2-D$ point whose first coordinate is chosen from one of the latency values and the second is chosen from one of the quality values. This section reviews some metrics commonly used in research that depict this quality-latency trade-off.

3.2.1 Quality metrics

In terms of quality measuring, common evaluation metrics that are mentioned in Chapter 1 can be reused for evaluating the translation quality of an *online* system. Amongst these automatic metrics, BLEU is still the most commonly used. Elbayad et al. (2020c) also suggest the use of other quality metrics such as METEOR, TER, ROUGE-L (Lin (2004)) and BERTScore (Zhang et al. (2019)) for evaluating translation quality. Most of the time, one will not find these metrics stand alone in assessing *online* translation systems, but going hand in hand with some latency metrics presented in the next subsection.

3.2.2 Latency metrics

Latency metrics aim to measure, either directly or indirectly, the delay incurred by the online translation system during the translation process. Some of the most commonly used latency metrics are briefly presented as follows:

Average Proportion (AP) (Cho and Esipova (2016)): indirectly measures the latency by averaging the absolute source delay incurred by each target token:

$$AP = \frac{1}{|X||Y|} \sum_{t=1}^{|Y|} g(t) \tag{3.1}$$

The value of AP falls between $[0, 1]$. Computing AP is simple, however, as observed by (Ma et al. (2019)), this metric is sensitive to the input length. They observe that their $wait - 1$ policy, when $|X| = |Y|$, incurs different AP values as the sequence length changes (for example, $AP = 1$ when $|X| = |Y| = 1$, $AP = 0.75$ when $|X| = |Y| = 2$ and its value approaches 0.5 when $|X| = |Y| \to \infty$). Secondly, Ma et al. (2019) also observe that, being a percentage, this metric is not obvious to the user the actual delays in number of source tokens.

Average Lagging (AL) (Ma et al. (2019)): measures the average rate by which the online translation system lags behind their ideal $wait - 0$:

$$AL = \frac{1}{\tau} \sum_{t=1}^{\tau} g(t) - \frac{t-1}{\gamma} \tag{3.2}$$

with the cut-off step τ is defined as the earliest timestep at which the MT system has consumed the entire source sequence:

$$\tau = argmin_t[g(t) = |X|] \tag{3.3}$$

and the target-to-source length ratio $\gamma = |Y|/|X|$ is a scale factor accounting for the source and target having different sequence lengths.

AL has an advantage of being length-invariant and intuitive. Moreover, it directly describes the lagging behind the ideal policy. However, Cherry and Foster (2019) show several disadvantages of AL. Firstly, they argue that AL is not differentiable because of the $argmin$ in Equation 3.3. Therefore, this metric cannot be used for optimization purposes. In the quest for removing τ from Equation 3.2, they discover that τ is necessary for AL as it ignores completely the steps beyond τ, which decrease the average lagging undesirably. By truncating its average, AL enforces an implicit and poorly defined delay for writing target tokens but only when $t > \tau$. Cherry and Foster (2019) argue that we should charge this writing cost on the translation system all the time (i.e, both when $t \leq \tau$ and $t > \tau$). The following latency metric is a remedy for such underlying problems of AL.

Differentiable Average Lagging (DAL) (Cherry and Foster (2019)): is designed to solve the presented problems of AL. In details, they enforce a

minimum delay of $\frac{1}{\gamma} = \frac{|X|}{|Y|}$ units for writing any target token. This transforms the delay $g(t)$ in Equation 3.2:

$$g'(t) = \begin{cases} g(t) & t = 1 \\ max[g(t), g'(t-1) + \frac{1}{\gamma}] & t > 1 \end{cases} \tag{3.4}$$

$g'(t)$ represents the amount of delay incurred just before writing a target token, taking into account both the number of source tokens read (in order to write) and the time spent writing this target token. DAL is defined as following:

$$DAL = \frac{1}{|Y|} \sum_{t=1}^{|Y|} g'(t) - \frac{t-1}{\gamma} \tag{3.5}$$

3.3 Online Neural Machine Translation

Some early works in *online* statistical machine translation recognize the word-ordering problems when translating an SOV language such as Japanese or German to an SVO language such as English or Chinese: the system has to wait until the source language's verb appears (at the end of the source sentence) before predicting the corresponding target language's verb (which appears early in the target sentence) (Figure 3.2). Matsubara et al. (2000) tackle this problem of translating Japanese to English by predicting the English verb early. By contrast, Grissom II et al. (2014, 2016) try to predict the final verb of the source language (German) in advance in order to reduce the latency when translating the sentence into English. Grissom II et al. (2014) train a system which can make different *actions* (*Wait/Commit/Next Word/Verb*) based on the source input, target translation so far, and predictions of the unseen words. Oda et al. (2015) extend the idea of using prediction, training a statistical model that predicts future syntactic constituents based on features of the input segment, and then apply this syntactic prediction on deciding whether to *wait* for more input when the current context is not enough to generate a fluent translation. He et al. (2015) propose to rewrite the reference translations to make their word order closer to the source language's word order (more monotonic (interpretation-like) translations), and train their MT systems on this rewritten data.

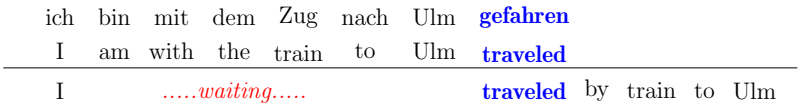


Figure 3.2: Grissom II et al. (2014)'s example of translating from a German (a SOV language) to English (a SVO language). The verb "**gefahren**" appears at the end of the sentence, forcing the system to wait until the final source word is revealed.

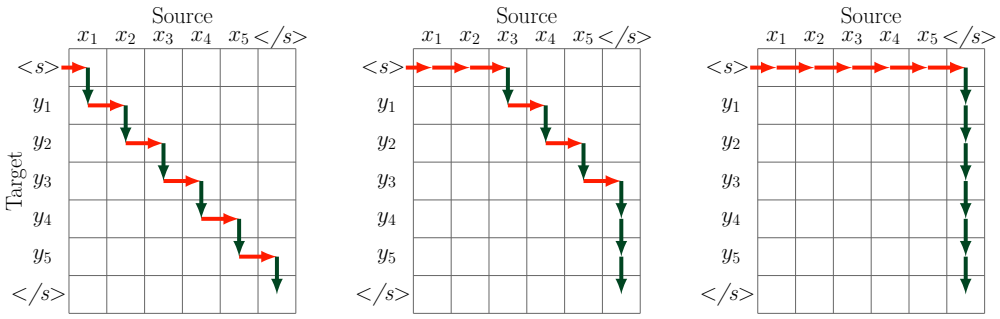
3.3.1 Deterministic online translation policy

In one of the pioneer works on *online* neural machine translation, Cho and Esipova (2016) introduce a manually designed non-trainable waiting criteria that alternates *READ/WRITE* operations. This decoding strategy allows them to reuse pre-trained offline models in online decoding mode. Inheriting the concept of *agent* who can make *READ/WRITE* decisions, Dalvi et al. (2018) design a static *READ/WRITE agent*, who first reads S input tokens, and alternates between a same number of *WRITE* and *READ* operations until the entire source sequence is consumed. They show that their approach outperforms those of Cho and Esipova (2016), and the fact that number of *WRITE/READ* can be tuned allows their approach to have better control of translation delay. In the same spirit, Ma et al. (2019) propose to train their *online* NMT network end-to-end based on the Transformer architecture. This network integrates a *wait-k* decoding policy, which now becomes one of the most commonly used decoding approaches for online MT.

wait-k: is a *prefix-to-prefix* framework that consists of an *agent* which reads k source tokens at the first step, and then alternates single *WRITE/READ* operations until all source tokens are consumed (Figure 3.3). The $g(t)$ function and the cut-off step of this policy are formally defined as in Equation 3.6 and 3.7, respectively.

$$g_{wait-k}(t) = \min\{k + t - 1, |x|\} \tag{3.6}$$

$$\tau_{g_{wait-k}}(|x|) = |x| - k + 1 \tag{3.7}$$



(a) Wait-1 (b) Wait-3 (c) Wait-∞

Figure 3.3: Wait-k decoding (illustration from Elbayad (2020)) as a sequence of READ and WRITE operations over a source (horizontal) and target (vertical) grid. After reading the first k source tokens, the decoder alternates between WRITE and READ operations. In wait-∞ decoding, the source is fully read before any WRITE operation.

This framework allows the model to be trained end-to-end to optimize the training objective over a data set D given in Equation 3.8, with the probability of translating simultaneously X into Y given in Equation 3.9.

$$\ell_g(D) = -\sum_{(X,Y^*)\in D} \log p_g(Y^*|X) \tag{3.8}$$

$$p_g(Y|X) = \prod_{t=1}^{|Y|} p(y_t|X_{\leq g(t)}, Y_{<t}) \tag{3.9}$$

Ma et al. (2019) show that, by choosing k appropriately, they can have control over the latency with this policy. Furthermore, end-to-end models with wait-k policy are shown to be able to make implicit anticipation on the target side after being trained in online mode. This is contrasted with forcing pre-trained offline model (train with $k = \infty$) to decode with smaller k values (*test-time* wait-k), which usually gives worse results (Figure 3.4). This, however, directly brings up a non-trivial question of how k should be set during training time. Elbayad et al. (2020a) propose to jointly optimize across multiple $wait - k$ paths instead of training separately many models with specific k values. They show that by doing so, the burden of manually choosing k and training many models according to the specific chosen values of k is relieved, whereas they can achieve a single model that performs relatively well in comparison with training on a single manually selected path.

	1	2	3	4	5	6	7				
	布什	总统	在	莫斯科	与	普京	会晤				
	Bush	**president**	**in**	**Moscow**	**with/and**	**Putin**	meet				
(a)	*...wait 2 words...*		pres.	bush	**met**	with	putin	in	moscow		
(b)	*.....wait whole sentence.....*							pres.	bush	met	with putin in moscow
(c)	*...wait 2 words...*		pres.	bush	in	moscow	and	po-ite	meeting		

Figure 3.4: An example from Ma et al. (2019) depicting that a trained wait-2 model (a) can correctly predict the English verb "met" given just the first 4 Chinese words (in **bold**), saying **"Bush president in Moscow"**, even though the Chinese word corresponding to "met" has yet to appear. The baseline offline model while reading the whole input sequence (b) can produce the same translation at the expense of a much bigger latency. Whereas, this baseline when being tested with test-time wait-2 (c) gives bad translation.

Wait-k is desirable for its simplicity, however, when aggressively generating only one hypothesis word at a time, it is prone to anticipation errors because committed errors from any steps would be propagated to the later steps, inducing more mistakes in the future. Even though the conventional beam search can be applied on the "tail" of the output when all source tokens have been processed ($Y_{\geq|X|-k}$), the direct application of the traditional beam search on earlier steps is impossible. Zheng et al. (2019c) propose *Speculative Beam Search*, which "hallucinates" w more steps into the future to implicitly accumulate more scores from a target language model, which consolidate the confidence of the chosen current word (Figure 3.5). Inspired by this method, Zheng et al. (2020b) suggest to use wait-k policy but deliberately over-generating w extra target tokens ($\hat{y}_t^{\leq w} = \{y_t^1, ..., y_t^w\}$) at each decoding step t, beside the hard committed target token y_t of step t. These extra words ($\hat{y}_t^{\leq w}$) are made visible to the user, but can be promptly corrected when more context of step $t+1$ indicates so, while y_t stays intact (Figure 3.6). This enjoys the advantage of re-translation as the latency stays lower, but at the same time, reduces the inconvenience for the users when only a small number of target tokens get changed.

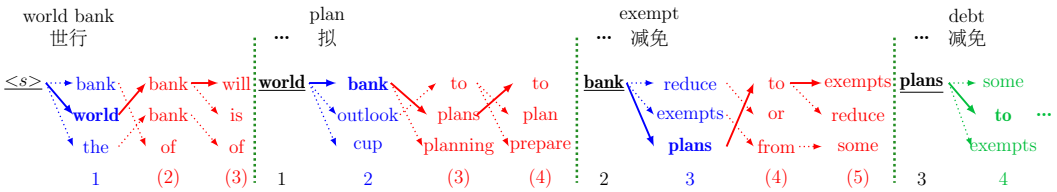


Figure 3.5: The illustration of the Speculative Beam Search with wait-1 policy (Zheng et al. (2019c)). There are two extra words (drawn in red) ($w = 2$) considered in order to consolidate the confidence of the chosen word (drawn in blue). When the source last word "债务"(debt) is reached, the conventional beam search is applied (drawn in green).

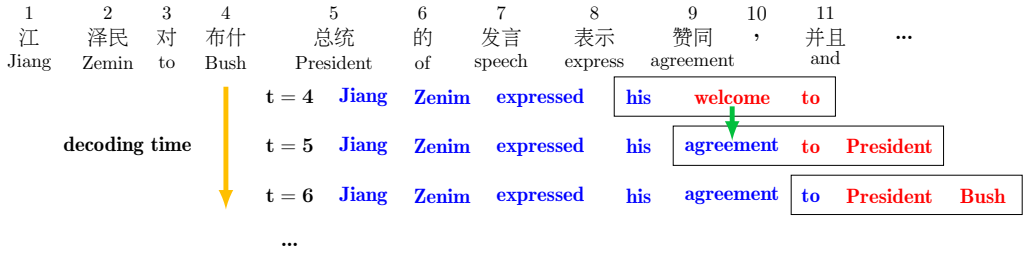


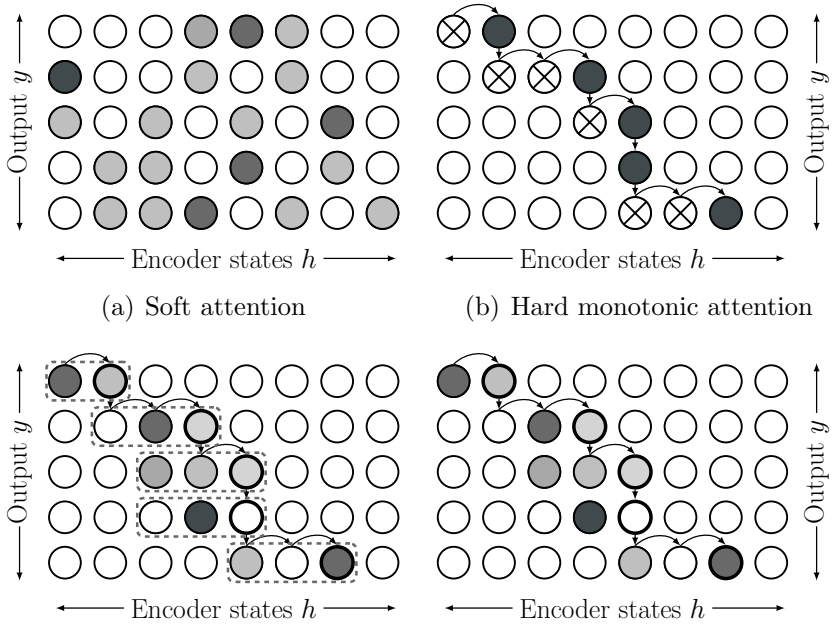
Figure 3.6: Zheng et al. (2020b)'s Opportunistic Decoding purposely generates two extra words y_4^1, y_4^2 = "welcome to" at time $t = 4$ along side with y_4 = "his" when the input x_9 = "赞同"(agreement) has not appeared yet. At $t = 5$, this word appears, the decoder promptly corrects the previously made mistake "welcome" by y_5 = "agreement" and generates two extra target words y_5^1, y_5^2 = "to President".

3.3.2 Adaptive online translation policy

Different from the above attempts to build deterministic waiting policies, Gu et al. (2016) use reinforcement learning in order to train an *agent* (parameterized by an RNN) on top of a pre-trained offline NMT model. This *agent* is trained to make decisions on whether to *READ* another source word or to *WRITE* a target word using the pre-trained base model's outputs, while the base model's parameters remain untouched. Also trying to develop a dynamic *agent*, Alinejad et al. (2018) propose to equip their *agent* with another *action*: *PREDICT* (alongside with the *READ* and *WRITE* action). In this *PREDICT* action, their end-to-end online NMT predicts next words of the input and uses this prediction to augment the context from which the decoder decides the next target words. Zheng et al. (2019a) use a pre-trained

model to generate a sequence of *READ/WRITE action* for each parallel text in the corpora. From the sentence pair and this additional *READ/WRITE* sequence, they then apply supervised-learning method to learn a parameterized policy for their simultaneous translation *agent*. Other the other hand, Zheng et al. (2019b) achieve their dynamic *agent* by training an end-to-end model, which can predict an additional *"delay"* token (from the target vocabulary), from scratch using the restricted dynamic oracle learning policy. Their model makes explicit *WRITE* actions (i.e, continuing predicting next target words) until it predicts the *"delay"* token, which tells the model to commit a *READ* action. To obtain an adaptive policy, Zheng et al. (2020a) design a simple algorithm which can heuristically choose at each step a $wait - k'$ from a set of $wait - k$ policies that is most confident with its output at that step.

Adaptive policies as attention: Also attempting to develop dynamic online translation, Raffel et al. (2017) encourage another research branch that focuses on learning monotonic alignments, which enables linear time computation of weights and online decoding instead of requiring access to the full source sequence to compute weights as the conventional attention usually does. Instead of monotonically attending to a single entry in memory at each output timestep, Chiu and Raffel (2017) propose **M**onotonic **Ch**unkwise **A**ttention (MoChA) which allows the model to perform soft attention over small chunks of the memory preceding chosen to attend by a hard monotonic attention mechanism. Also using hard monotonic attention head to adaptively schedule the reading of source tokens, Arivazhagan et al. (2019) introduce **M**onotonic **I**nfinite **L**ookbac**k** (MILk) attention, which allows a soft attention head extending from where the monotonic head stops back to the beginning of the source sequence. Ma et al. (2020d) propose yet another attention mechanism named **M**onotonic **M**ultihead **A**ttention (MMA), which extends the monotonic attention mechanism to multihead attention.



(a) Soft attention

(b) Hard monotonic attention

(c) Monotonic chunkwise attention (MoChA)

(d) Monotonic Infinite Lookback attention (MILk)

Figure 3.7: Illustrations of adaptive policies as attention, where the possibility of the model attending to a given memory entry (horizontal axis) at a given output time step (vertical axis) is presented by each colored node (darker colors present bigger possibilities). (a) In soft attention, the context vector is a weighted sum of the probabilities assigned by the model to each memory entry at each output time step. (b) In monotonic attention (Raffel et al. (2017)), the model chooses whether to move on to the next memory entry (shown as nodes with $\times$) or stop and attend (shown as black nodes) by inspecting the memory entries left-to-right, then hard-assigning the context vector to the memory entry that was attended to. (c) MoChA (Chiu and Raffel (2017)) performs soft attention over small chunks of the memory preceding chosen to attend by a hard monotonic attention mechanism. (d) MILk attention (Arivazhagan et al. (2019)) performs soft attention head extending from where the monotonic head stops back to the beginning of the source sequence.

3.3.3 Re-translation

Different from the above approaches, which do not allow translated text to be modified throughout the translating process, Niehues et al. (2016b)'s approach allows translated text to be revised (re-translated) when more context is available during online translation to alleviate decoding constraints. Zheng et al. (2020c) propose an opportunistic decoding technique. This approach, which is presented as a variant of wait-k policy, can also be considered as a re-translation approach in which, a certain number of extra words are over-generated at each step on purpose. This is coupled with a timely correction ability, which corrects the mistakes in these over-generated words when more source context is available and indicates so.

Arivazhagan et al. (2020a) make the first first comparison of re-translation and online decoding strategies for simultaneous translation, in which they find that re-translation can be as good or better than the state-of-the-art *wait-k* in *online* translation. However, the main drawback of re-translation is that it causes inconvenience to the user when generated outputs are corrected at later steps.

3.4 Conclusion

In this chapter, we present the state-of-the-art of Online Neural Machine Translation, which is distinguished from the offline translation task which enjoys the advantage of having the full input sequence to condition the generation of the output tokens. Online Machine Translation, on the other hand, must start generating partial output hypotheses based on partial input in order to balance the trade-off between translation quality and latency. Our focus is on the discussion about different online decoding strategies of both deterministic and dynamic nature. The dominant approach of deterministic strategies is the wait-k policy, which is a prefix-to-prefix framework consisting of an agent which reads k source tokens at the first step, and then alternates single WRITE/READ operations until all source tokens are read. By contrast, dynamic strategies aim to either train the agent to make more dynamic READ/WRITE actions or to learn more monotonic attention instead of the traditional soft attention which requires access to the full source sequence. Beside translation quality, online translation is also evaluated based on latency which is most often measured by Average Lagging. Other automatic metrics for latency measuring presented in this chapter include Average Proportion and Differentiable Average Lagging.

Chapter 4

Online Neural Speech Translation

4.1 Definition

Similar to online (text-to-text) MT, online (or sometimes called *simultaneous*) AST consists of AST systems which start generating an output hypothesis of either text or speech before the entire input speech sequence has been made available. Although such systems are enormously useful in real-life applications, for example, video caption translations and real-time language interpretation, etc., online AST is extremely challenging as it not only preserves the challenges of online MT but also inherits difficulties transferred from online ASR. This chapter is dedicated to discussing the state-of-the-art of online AST, stressing mainly on end-to-end neural speech-to-text approaches. Before embarking on the details, let us formalize the task as the following:

Similar to online MT, let (X, Y) be a pair of source-target sequences of lengths $|X|$ and $|Y|$ respectively. In the context of this thesis, we only concern with speech as the source and text as the target. Therefore, the source X can be denoted as a sequence of acoustic features $X = [x_1, x_2, ..., x_{|X|}]$ extracted from speech samples every T_s ms. *Online* AST alternately switches between interleaved READ and WRITE operations, consuming acoustic frames from the source X and producing tokens for the target Y. Let $g(t)$ be a monotonic non-decreasing function of t that denotes the number of source acoustic frames processed by the *online* translation system in order to produce the target token y_t. *Online* AST systems predict y_t conditioning on the target prefix $y_{<t}$ and the partial source context $x_{\leq g(t)}$.

4.2 Evaluation

The same as text-to-text online machine translation, online speech translation uses both quality and latency metrics to measure the performance of the translation systems. BLEU remains the first choice when one needs to assess the quality of the translation. Besides, METEOR or TER can also be used for translation quality measuring. As for latency metrics, all previously presented latency metrics (Section 3.2.2) can be used in this task. However, while AP

and DAL can be used directly, AL, on the other hand, needs some modifications to be more suitable for speech-to-text translation. Ma et al. (2020a) make some changes to adapt this metric (Section 3.2.2) to the speech task.

Adaptive Average Lagging: Ma et al. (2020a) find that the second term inside the sum of Equation 3.2 ($\frac{t-1}{\gamma} = (t-1)\frac{|X|}{|Y|}$) is not robust for speech models, which tend to generate the end of sentence token too early, for example, after a long pause even though the entire source input has not been processed by the model. They argue that for this reason, speech model can obtain relatively good quality-latency (BLEU/AL) trade-offs, which does not reflect the reality. Therefore, instead of using the hypothesis sequence length $|Y|$, they propose to use directly the reference length $|Y^*|$ (Equation 3.2). Moreover, they also introduce T_s into Equation 3.2 in order to evaluate lagging based on time instead of steps.

$$AL_{speech} = \frac{1}{\tau}\sum_{i=1}^{\tau} g(t)T_s - (i-1)\frac{|X|}{|Y^*|}T_s \tag{4.1}$$

Computation Aware Average Lagging: (Ma et al. (2020c)) propose a variant of AL, which replaces the term $g(t)T_s$ in Equation 4.1 by the actual time spent on generating y_t. They argue that this metric reflects a more realistic evaluation, especially in low-latency regimes, and it can distinguish streaming capable systems.

4.3 Online Neural Speech Translation

Early attempts in online AST consist of optimizing segmentation methods of the input sequence (Bangalore et al. (2012); Sridhar et al. (2013); Oda et al. (2014); Fujita et al. (2013); Yarmohammadi et al. (2013)). While Bangalore et al. (2012) segment the input based on silence, Sridhar et al. (2013) experiment with conjunction-based segmentation and comma-based segmentation strategies. Yarmohammadi et al. (2013) propose a segmentation method that aims at splitting the source sentence into segments that can be monotonically translated to the target language. Fujita et al. (2013) develop a segmentation method based on the phrase table and reordering probabilities used in phrase-based translation systems. Different from these heuristics segmentation methods, Oda et al. (2014) propose learning methods that aim directly at maximizing the BLEU score of the translation system, while Niehues et al. (2016a) focus on re-translation that allows initial output to be displayed at a low latency and allows the system to iteratively revise translations by re-translating new source text sent by an ASR component.

Inspired by these early works, more recent efforts in online AST have been making an extensive use of neural networks to solve this challenging task, either by coupling neural ASR systems with neural MT systems or directly translating the input speech incrementally using end-to-end online AST models. This section gives an overview of these two different neural methods for online AST.

4.3.1 Cascaded models

Before the emergence of end-to-end approaches, most efforts in online AST revolve around cascaded systems, which exploit simultaneous text-to-text translation modules presented in Section 3 to translate the partial transcription outputted by a streaming ASR module into target-language text (Xiong et al. (2019); Wilken et al. (2020)). Xiong et al. (2019) propose a context-aware translation model named DuTongChuan, which constantly reads streaming text from an ASR model, and determines the boundaries of information units simultaneously. These information units are detected by a detector, which is basically a pre-trained language model (Devlin et al. (2018)) fine-tuned on a classification task that predicts the potential separator tokens *SEP*. These tokens are considered as segmentation units, which split the input sentence into smaller sub-sentences. These sub-sentences are then translated with two decoding strategies: (1) partial decoding which is technically a wait-k like MT model tailored on their settings to translate the very first sub-sentence and (2) context-aware decoding, their proposed online MT model which translates from the second sub-sentence till the end. Wilken et al. (2020) segment the streaming input text from ASR by using a source chunk boundary detection, which is modeled by an LSTM encoder followed by a softmax layer, making a binary decision if the current source token is the end of a chunk or not. Their translation model is attention-based, comprising of this LSTM encoder and an LSTM decoder which is triggered whenever a source chunk boundary is detected. On this event, the decoder consumes the output representation of the encoder in order to produce target tokens until the end-of-chunk signal is predicted similar to what has been done for the source side. This whole model is trained jointly on the chunk-based bilingual data generated by their proposed method based on word alignment to generate the source and target chunks.

Different from the above methods, a few attempts (Nguyen et al. (2020b); Arivazhagan et al. (2020b)) focus on re-translation as inspired by Niehues et al. (2016a). These works rely on cascaded systems of a streaming ASR model followed by a neural MT model adapted for the re-translation purpose.

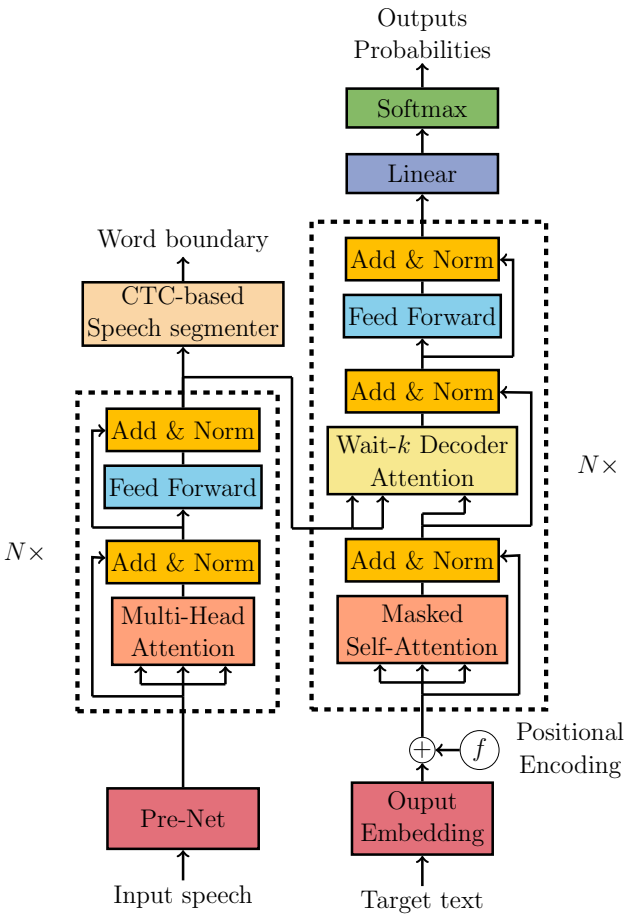


Figure 4.1: SimulSpeech model (Ren et al. (2020)).

4.3.2 End-to-end models

Attempts to exploit end-to-end models to directly translate the source speech into target text simultaneously have arisen recently, mostly adapting the text-based wait-k to the speech task (Ren et al. (2020); Ma et al. (2020b); Han et al. (2020); Ma et al. (2021)).

SimulSpeech: proposed by Ren et al. (2020) is one of the first end-to-end online AST models. This model (depicted in Figure 4.1) has a Transformer-based encoder-decoder backbone, which handles the online translation task by adapting the wait-k policy. The switching between the $READ/WRITE$ operations is decided by a speech segmenter, which is trained on top of the Transformer speech encoder to align source speech frames with source text based on CTC loss. Equation 4.2 formulates this loss, with $\mathcal{X}, \mathcal{Y}^{src}$ denote the parallel data set containing source speech X paired with source text Y^{src}. The term $\sum_{z \in \phi(Y^{src})} P(z|X)$ can be interpreted as the likelihood of Y^{src} evaluated

as the sum over all the probabilities of intermediate representation CTC paths ($\phi(Y^{src})$). Besides, in order to stabilize the training of the attention matrix of the online AST model, this work introduces *Attention-Level Knowledge Distillation*, which makes use of the less challenging auxiliary simultaneous ASR and NMT task to learn the corresponding attention matrices. These two are then multiplied and binarized to guide the training of the main attention matrix of the SimlSpeech model via a loss $\mathcal{L}_{attn\_kd}$. Furthermore, *Data-Level Knowledge Distillation* (Kim and Rush (2016); Tan et al. (2019)), which gives the third loss term $\mathcal{L}_{data\_kd}$ of the total loss to train SimulSpeech (Equation 4.3), is also used in order to further boost the performance of the main online AST model. Details of these two losses can be found in the paper (Ren et al. (2020)). SimulSpeech is trained to minimized the total loss in Equation 4.3, where $\lambda_1, \lambda_2, \lambda_3$ denote the weights of the three losses, and serve as hyperparameters of the model.

$$\mathcal{L}_{ctc} = \sum_{(X,Y^{src})\in(\mathcal{X}\times\mathcal{Y}^{src})} \sum_{z\in\phi(Y^{src})} P(z|X) \tag{4.2}$$

$$\mathcal{L} = \lambda_1\mathcal{L}_{ctc} + \lambda_2\mathcal{L}_{attn\_kd} + \lambda_3\mathcal{L}_{data\_kd} \tag{4.3}$$

Simul-ST: also attempting to delevelop end-to-end online AST systems, (Ma et al. (2020b)) evaluate the two most commonly used types of online MT, such as Monotonic Multihead Attention (referred to as the flexible policy) and wait-k (referred to as the fixed policy), on the online AST task. Ma et al. (2020b)'s adaptation of these online MT approaches on the speech translation task also includes implementing an additional *Pre-Decision module* (Figure 4.2), which pre-decides either to continue reading source input frames or let the online policy make READ/WRITE decisions based on a probability p_{tr} generated on each encoder state ($p_{tr} > 0.5$ would trigger the online policy to make a decision). Two types of Pre-Decision modules are introduced in Ma et al. (2020b), including *Fixed Pre-Decision* which triggers the writing decision after a fixed number of frames, and *Flexible Pre-Decision* which uses the alignment between encoder states and source labels to determine the source boundaries of either word or phoneme level that trigger the writing decision. Coupling either of the Fixed/Flexible Pre-Decision module with Fixed/Flexible online MT policy, Ma et al. (2020b) obtain in total four different online AST models, whose speech encoders are all pre-trained by the ASR task. It is also worth to note that ASR training data is also required for the Flexible Pre-Decision module to work because it relies on the aligment between source frames and source labels.

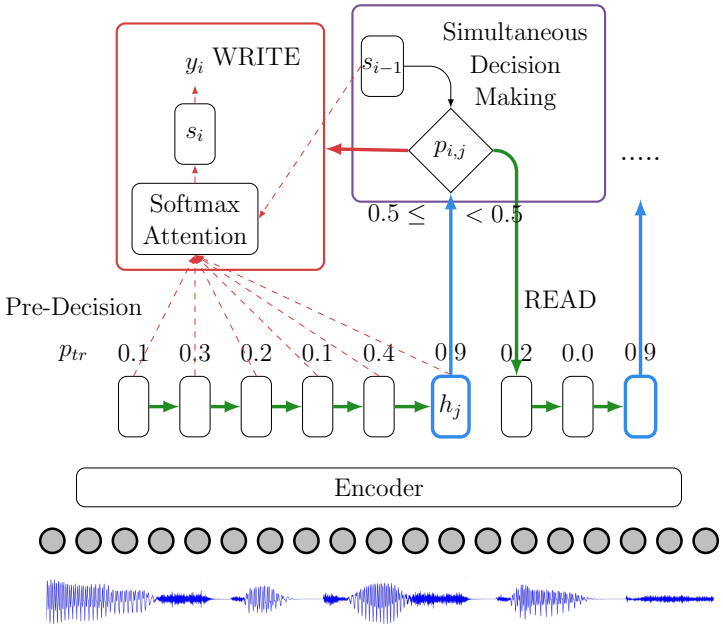


Figure 4.2: Illustration of Simul-ST architecture with Pre-Decision module proposed by Ma et al. (2020b). From each encoder hidden state, the Pre-Decision module computes the probability p_{tr} which decides to trigger the Simultaneous Decision Making governed by the online policy when $p_{tr} > 0.5$. This is depicted as blue arrows corresponding to $p_{tr} = 0.9$ in the Figure.

Augmented Memory Encoder: adapting the text-based wait-k policy for the speech task, (Ma et al. (2021)) learn a Transformer-based end-to-end online AST model consisting an augmented memory encoder and a simultaneous decoder. The encoder is designed to sufficiently encode the input speech sequence incrementally by exploiting an *Augmented Memory Encoder* originated by Wu et al. (2020). Illustrated by Figure 4.3, instead of attending to the entire input sequence X, the self-attentions of this encoder are designed to attend on a sequence of sub-utterance level segments $S = [s1, ...]$. Each sub-utterance s_n, beside having its main context c_n of size C, overlaps with s_{n-1} by a left context segment l_n of L frames, and with s_{n+1} by a right context segment r_n of R frames. Self-attention is computed at the segment level, with the query, key and value for each segment are defined in Equation 4.4,4.5 and 4.6.

$$q_n = W_q[l_n, c_n, r_n, \sigma_n] \tag{4.4}$$

$$k_n = W_k[M_{n-N:n-1}, l_n, c_n, r_n] \tag{4.5}$$

$$v_n = W_v[M_{n-N:n-1}, l_n, c_n, r_n] \tag{4.6}$$

$$m_n = \sum_{j'} \alpha_{-1,j'}(v_n)_{j'} \tag{4.7}$$

where $\sigma_n = \sum_{x_k \in s_n} x_k$ is referred to as the summarization query. The key aspect of this encoder is the introduction of the memory bank M_n which captures the information of the previous segments. A new slot of this memory (m_n) is generated at each step n as in Equation 4.7. Instead of storing all the history of the processed segments, Ma et al. (2021) propose to keep only N current banks instead: $M_{n-N:n} = [m_{n-N}, ..., m_{n-1}]$. The self-attention z_n which represents the current sub-utterance s_n is then calculated as in Equation 4.8 and 4.9. Note that only central encoder states of size C are kept in z_n ($N + L < j \leq N + L + C$) and the concatenation of the segment states $Z = [z_1, ...]$ is passed to the decoder. The decoder exploits the wait-k policy, which makes simultaneous READ and WRITE decisions based on chunks of a fixed number of encoder states. They propose to pre-train their models on the ASR task to better initialize the speech encoder.

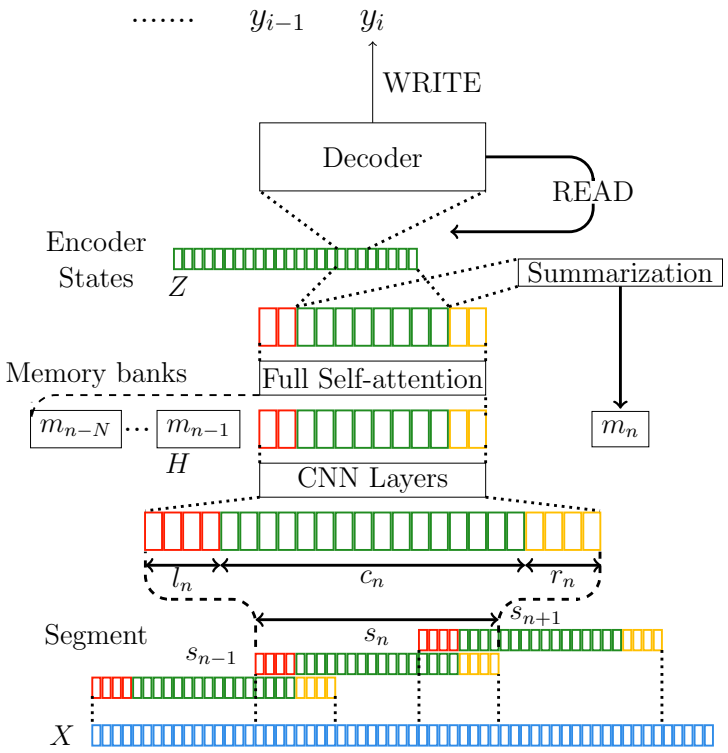


Figure 4.3: Illustration of the streaming AST model with Augmented Memory Encoder proposed by Ma et al. (2021).

$$\alpha_{j,j'} = \frac{\exp\left(\beta(q_n^\top)_j(k_n)_{j'}\right)}{\sum_a \exp\left(\beta(q_n^\top)_j(k_n)_a\right)} \tag{4.8}$$

$$(z_n)_j = \sum_{j'} \alpha_{j,j'}(v_n)_{j'} \tag{4.9}$$

Direct application of wait-k: another extension of the wait-k policy for the online AST task can be found in Han et al. (2020). However, they make one important observation that states that usually speech sequences are much longer than their text counterparts, and hence, the translation system ought to read more than one input frame at each encoding step. For this reason, the original text-based wait-k policy is slightly modified as in Equation 4.10 which formulates the number of frames the system consumes at each reading step: $s \geq 1$ (Figure 4.4). Note that at step 0-th, $s = k$, and $s = 1$ reduces Equation 4.10 to the original text-based wait-k policy. The catch-up frequency is calculated as: $c = r - 1$, with $r = \overline{|Y|/|Y^{src}|}$ is the text-to-text source-target emission rate which is loosely set to 1.0 in their experiments with the English-German language pair. Beside making use of different data augmentation techniques in order to substantially increase the amount of training data, Han et al. (2020) also exploit Modality Agnostic Meta-Learning approach proposed by Indurthi et al. (2020) for better initializing the parameters of the online AST model. The auxiliary source tasks used for this purpose including ASR, MT and offline AST.

$$g_{wait-k,c,s}(t) = \min\{(k + t - 1 - ct) * s, |X|\} \tag{4.10}$$

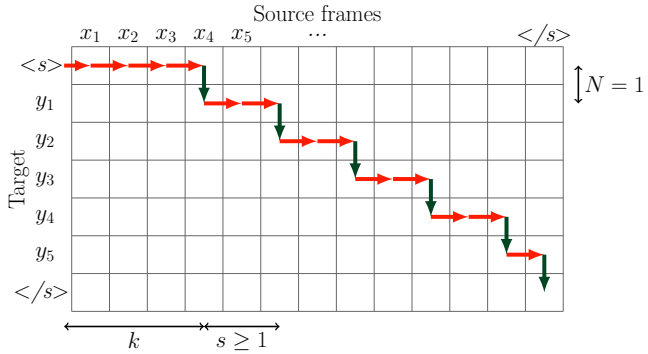


Figure 4.4: Wait-k policy, when adapted for online AST by Han et al. (2020), reads more than 1 source frames ($s \geq 1$) at each decoding step after the first step in order to write only one target token ($N = 1$).

4.4 Conclusion

In this chapter, we present state-of-the-art of Online Speech Translation. Most efforts so far in this field focus on building cascaded systems, which exploit simultaneous text-to-text translation modules to translate the partial transcription outputted by a streaming ASR module. By contrast, attempts to exploit end-to-end models to directly translate the source speech into target text simultaneously have emerged very recently, mostly adapting the wait-k proposed for text-to-text translation to the speech task. Early end-to-end models either make READ/WRITE operations in a deterministic manner (i.e, making READ and WRITE decisions on chunks of a fixed number source frames), or are trained jointly with an additional module, for example, a CTC-based Speech Segmenter or Pre-Decision module, which evokes dynamic READ and WRITE decisions.

Chapter 5

Self-supervised Learning Speech Representation

5.1 Definition

The same as other Machine Learning tasks, how to represent the input effectively is one of the most important questions AST needs to give an answer for. Extracting features from speech is not at all trivial, because speech signals carry a great load of information, for instance, the words spoken, the emotional state of the speaker, the speaker's dialect and identity, the language used, etc., which can directly or indirectly affect the performance of the translation system. The ultimate objective of the AST feature extraction process is, therefore, to mitigate or eliminate information in the input signal that is not relevant to the spoken content, and enhance the useful aspects of the signal that contribute positively to the detection of phonetic differences. From the beginning, AST had inherited speech representations developed for earlier born speech processing tasks, especially ASR speech representations Schafer and Rabiner (1975). Davis and Mermelstein (1980) divide significant traditional speech representations into two big groups: (1) those based on the Fourier spectrum (*Mel Filter-bank features (Filterbank)*, *Mel-Frequency Cepstrum Coefficients (MFCCs)*), and (2) those based on the linear prediction spectrum (*Linear Prediction Coefficients (LPC)*, *Linear Prediction Cepstral Coefficients (LPCCs)*). Recently, these handcrafted surface features are criticized for not adequately capturing high-level properties of speech (Chung et al. (2019)). For this reason, brand new approaches for representing input speech using *Self-Supervised Learning (SSL)* to learn useful latent features from large amounts of unlabelled speech data have been proposed (Chung et al. (2019); Chung and Glass (2020b); Oord et al. (2018); Baevski et al. (2019); Schneider et al. (2019a); Baevski et al. (2020b)). These newly proposed features have been gradually beating manually specified features in speech tasks, and therefore, making them largely redundant.

This chapter reviews several commonly used handcrafted speech representations in Section 5.2 and their promising alternatives SSL features in Section 5.3.

5.2 Conventional Approaches

Conventional speech representations, which are labelled by Chung et al. (2019) as surface features, have been staying in practice for a long time. These methods selectively use mathematical tools to directly transform the digitized speech signals into a denser representation which is possible for computers to store and process, and more importantly, suitable for the application in question. Schafer and Rabiner (1975) examine several speech representations of this kind, which are still in use these days by different speech tasks, in an increasing order of the complication of computation. In the context of this thesis, we only review two of the most commonly used speech features for ASR and AST, *Log Mel Filterbank features* (commonly dubbed as *Filterbank*) and *Mel-Frequency Cepstrum Coefficients (MFCCs)*, both of which are based on *short-time analysis*. While the former is based on *spectrum analysis*, the latter applies *Inverse Discrete Fourier Transform (IDFT)* on the former's output, and therefore, is based on *cepstrum analysis* (**ceps**trum is a wordplay of **spec**trum, where the four initial letters **c-e-p-s** is backward-written from **s-p-e-c**).

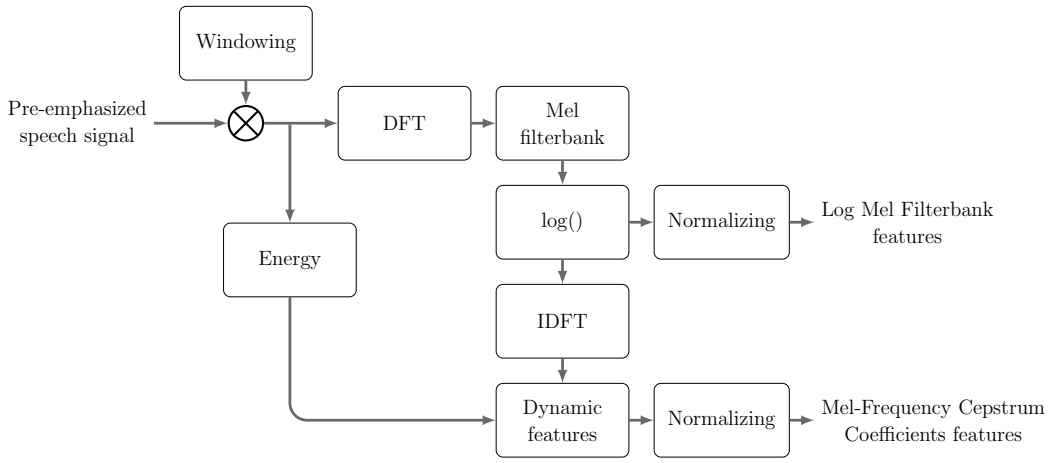


Figure 5.1: The calculation pipeline of *Log Mel Filterbank features* and *Mel-Frequency Cepstrum Coefficients (MFCCs)*.

Figure 5.1 illustrates the calculation pipeline of Filterbank and MFCC features. This process consumes pre-emphasized digitized speech signals (a speech processing trick which helps boost the amount of energy in the high frequencies), and chops them up into smaller frames. This practice is built on the concept of *short-time analysis* which assumes that speech waveform can be considered *stationary* over a sufficiently short-time interval, even though it is not so over a long-time interval. Framing the speech signal is done by

using sliding *windows* (discrete mathematical functions) whose width is usually chosen between 20 $milliseconds(ms)$ to $40ms$. In practice, three common windows are used including *rectangular window, hanning window* and *hamming window* (Taylor (2009)), which are distinguished from each other by the mathematics formalizing them, and consequently transform the signal differently. Another common practice of using sliding windows is to overlap them. Popular settings are $25ms$ for the window size and placing consecutive windows $10ms$ after each other.

The next step, which is abbreviated as ***DFT*** in Figure 5.1, is to calculate the *power spectrum* of each frame. This, also called as *periodogram estimate* of the power spectrum, helps identify which frequencies are present in the frame. This is inspired by an organ in the human's ear called *cochlea* which has small hairs located at different spots. These hairs vibrate depending on the frequencies of the incoming sound, and depending on the location in the cochlea that is triggered, different nerves fire electrical signals to the brain informing that certain frequencies are present. *Discrete Fourier Transform (DFT)* is the core of this step, which transforms the each frame $x_i(n)$ in time domain (i denotes the $i-th$ portion of the whole input signal $x(n)$, which gets chopped by the window $h(n)$ with length N) to frequency spectrum (Equation 5.1). The power spectrum is calculated on the frequency spectrum (Equation 5.2).

$$X_i(k) = \sum_{n=1}^{N} x_i(n)h(n)e^{-j2\pi kn/N} \tag{5.1}$$

$$P_i(k) = \frac{1}{N}|X_i(k)|^2 \tag{5.2}$$

where $X_i(k)$ denotes the frequency spectrum of windowed frame $x_i(n)h(n)$, $P_i(k)$ is its corresponding power spectrum, and K is the length of the DFT.

The third step is also motivated by human's perception of sound. In detail, we cannot tell the difference between two closely spaced frequencies, and perceived frequency resolution decreases as the frequency increases. This is depicted by the *Mel scale*, which relates the perceived frequency with its actual measured frequency (Equation 5.3).

$$M(f) = 1125\ln(1 + f/700) \tag{5.3}$$

In practice, we apply triangular bandpass filters to the previously calculated periodogram bins to get an idea of how much energy exists in various frequency regions. These filters, which are called ***Mel filterbank***, are illustrated in Figure 5.2. As shown in the figure, the first filter is very narrow, followed by increasingly wider filters. This is to mimic the decay of perceived

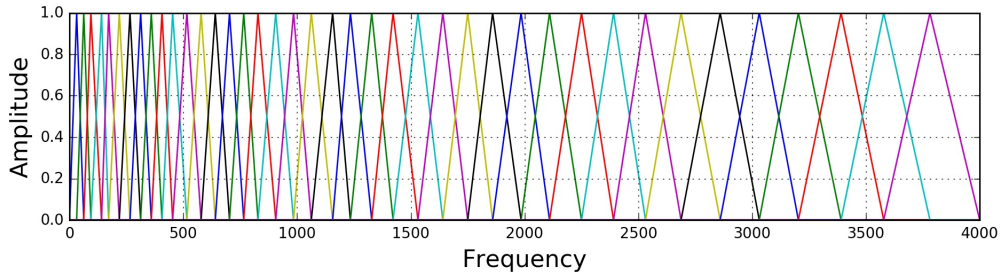


Figure 5.2: Mel filterbanks
(illustration from `https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html`).

frequency resolution in humans as the frequency increases. Figure 5.2 draws 40 mel filterbanks, but in pratice, one can use from 20 to 40 of these band-pass filters. The number of filters will result in the number of scalar values representing the energies of each frame.

Coming next is the ***log*** step, where we take the logarithm of the filterbank energies calculated from step 3th. This is because human's hearing is not linear. The logarithm is hoped to make our features match more closely to the sound humans actually perceive.

If we only concern with Log Mel Filterbank features, we can apply *mean and variance normalization* to the output of the log step above, which results in our desired speech features.

By contrast, if the Mel-Frequency Cepstrum Coefficients representation is desired, the ***IDFT*** step needs to be involved. One of the reasons for doing this is that, as shown in Figure 5.2, filterbanks are all overlapping, and this consequently makes the filterbank coefficients computed in the previous step highly correlated, which could be problematic for some Machine Learning algorithms. This could be tackled by applying IDFT, or specifically its *Discrete Cosine Transform (DCT)* equivalent, which is commonly used in practice, in order to decorrelate the filterbank coefficients. Here we do cepstrum analysis, which, on the surface, seems like reversing the spectrum analysis' job (*"cepstrum"* is indeed a wordplay of *"spectrum"*, which has the first four letters **c-e-p-s** being backward-written from **s-p-e-c**). However, one should not expect to obtain the original input speech as the result, because the mel filterbank and the log operation were in place. Note that even though we usually choose to use 20 to 40 mel filterbanks in the previous step, typically for ASR, we only keep 12 coefficients for each frame, and these are called Mel Frequency Cepstral Coefficients.

The two last steps on the bottom of Figure 5.1 are optional, which give 27 other coefficients to make up the total 39 Mel-Frequency Cepstrum Coeffi-

cients which are very commonly used in practice. Note that appended to the 12 resulting coefficients of the previous step is the energy of each frame, that makes the 13-th coefficient. 13 others would come from the first difference of signal features ***Deltas***, which describes the dynamic transitions between consecutive frames. Finally, the last 13 coefficients, called ***Delta-Deltas*** would be calculated in the same manner, but on dynamic changes of Deltas instead of on the static coefficients.

5.3 Self-supervised Learning Approaches

Self-supervised Learning has been recently proposed as an interesting alternative for data representation learning. Proven useful learned representations can be found both in vision (Bachman et al. (2019); Chen et al. (2020)) and in NLP (Devlin et al. (2018); Peters et al. (2018)). The attractiveness of SSL in general and SSL from speech in particular is that it can leverage huge amounts of un-annotated data, which, as stated earlier, are not easily available for downstream tasks such as ASR or AST. This can be done by resolving pseudo-tasks, which do not require human annotation, as pre-training a feature extractor, which is then used to extract useful speech representations for the real (downstream) tasks. The two most commonly used approaches for SSL from speech are *Autoregressive Predictive Coding (APC)* and *Contrastive Predictive Coding (CPC)*. The former's pseudo-task considers the sequential structure of speech and predicts information about a future frame (Chung et al. (2019); Chung and Glass (2020b)), whereas the latter's consists of distinguishing a future speech frame from distractor samples (Oord et al. (2018); Baevski et al. (2019); Schneider et al. (2019a)) which is an easier learning objective compared to APC. These representations have been proven to improve the performance in several speech tasks (Chung and Glass (2020a)), while being less sensitive to domain and/or language mismatch (Kawakami et al. (2020)) and being transferable to other languages (Riviere et al. (2020)).

This section gives an overview of these SSL approaches with a major stress on CPC and some of its variants which are mainly used in our experiments related to SSL.

5.3.1 Contrastive Predictive Coding

Oord et al. (2018) aim at learning useful representations from high-dimensional un-annotated data by predicting the future in latent space. More particularly, they desire to learn representations that encode the underlying shared information between different parts of the data, and discard low-level more

local information and noise. With the hope to force their models to relate more global structure of the signal as predicting further in the future might require, they propose *Contrastive Predictive Coding (CPC)*, which optimizes their autoregressive models by a probabilistic contrastive loss. Their approach is applicable for a wide range of downstream tasks of different modalities including vision and NLP tasks. We focus on the underlying key aspects of this approach when being applied to speech tasks in the section.

Figure 5.3 illustrates Oord et al. (2018)'s CPC model, which consists of a non-linear *encoder* and a *context network*, denoted as g_{enc} and g_{ar} in the figure, respectively. The encoder $g_{enc} : \mathcal{X} \rightarrow \mathcal{Z}$ is responsible for mapping raw input sequence $x = x_1, x_2, ..., x_{|x|}$ to a sequence of potentially lower frequency latent feature representation $z = z_1, z_2, ..., z_{|z|}$: $z = g_{enc}(x)$. The autoregressive context network $g_{ar} : \mathcal{Z} \rightarrow \mathcal{C}$ combines multiple time-steps of the encoder's output to obtain contextualized representations $c_t = g_{ar}(z_{\leq t})$. These two components are trained jointly to make predictions of the future observations that maximize the *Mutual Information* between x and c defined in Equation 5.4.

$$I(x; c) = \sum_{x,c} p(x, c) \log \frac{p(x|c)}{p(x)} \tag{5.4}$$

However, as Oord et al. (2018) argue, modeling $p(x|c)$ directly would be too computationally intense. Therefore, they propose to maximize the mutual information between the encoded future latent representations (z_{t+k}) which is k steps in the future from the current time step t and the present context vector (c_t) instead. This is described by Equation 5.5.

$$I(z_{t+k}; c_t) = \sum_{z_{t+k},c_t} p(z_{t+k}, c_t|k) \log \frac{p(z_{t+k}|c_t, k)}{p(z_{t+k})} \tag{5.5}$$

They show that maximizing $I(z_{t+k}; c_t)$ is equal to minimizing a loss that is based on *Noise-Contrastive Estimation* (Gutmann and Hyvärinen (2010)). This loss, which is named *InfoNCE*, is formalized by Equation 5.6.

$$\mathcal{L}^{NCE} = -\mathbb{E}_X \left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)} \right] \tag{5.6}$$

with $\mathbb{E}_X$ denotes the expectation averaging the scores over a set $X = \{x_1, x_2, ..., x_N\}$ of N random samples consisting of one positive samples from $p(x_{t+k}|c_t)$ and $N-1$ negative samples from $p(x_{t+k})$. Oord et al. (2018) use *log-bilinear* for modeling the scoring function $f_k(x_{t+k}, c_t)$ (Equation 5.7). Negative samples can be drawn directly from the input sequence or from other sequences in the minibatch.

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^{\top} W_k c_t\right) \tag{5.7}$$

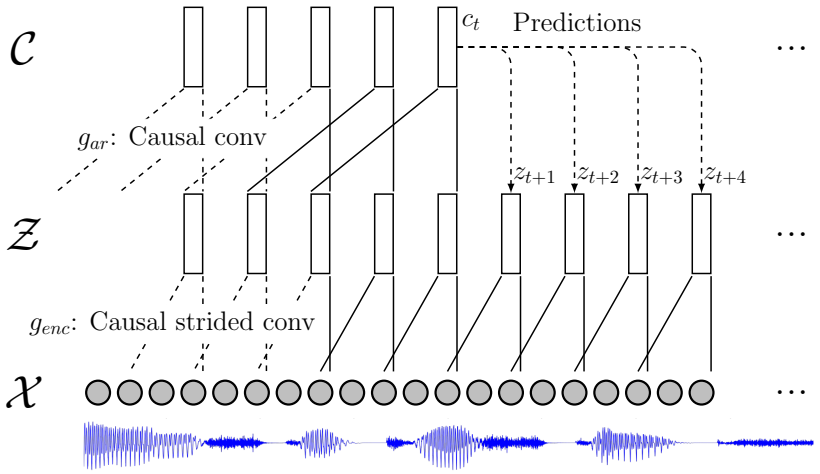


Figure 5.3: Illustration (borrowed from Kawakami et al. (2020)) of the CPC framework, which is pre-trained from raw audio data $\mathcal{X}$ which is encoded with two causal convolutional neural networks: $g_{enc} : \mathcal{X} \rightarrow \mathcal{Z}$ and $g_{ar} : \mathcal{Z} \rightarrow \mathcal{C}$ stacked on top of each other. The whole model is optimized to solve a next time step prediction task.

For extracting latent representations for the downstream tasks, either of z_t and c_t could be used.

5.3.2 wav2vec

One of the most famous variants of the CPC approach is *wav2vec* (Schneider et al. (2019a)), which decomposes Equation 5.6 for each $k = 1, 2, ..., K$ as in Equation 5.8.

$$\mathcal{L}_k^{NCE} = -\sum_{t=1}^{T-k} \left(\log \sigma(z_{t+k}^{\top} h_k(c_t)) + \lambda \mathbb{E}_{\tilde{z} \sim p_n}[\log \sigma(-\tilde{z}^{\top} h_k(c_t))]\right) \tag{5.8}$$

where $\tilde{z}$ denotes encoded representations of negative samples drawn from a proposal distribution p_n. $\sigma(z_{t+k}^{\top} h_k(c_t))$ denotes the probability of z_{t+k} being the true sample, and $\sigma(x)$ is the sigmoid function. $h_k(c_i) = W_k c_i + b_k$ is a step-specific affine transformation different for each step k. Schneider et al. (2019a) optimize the overall loss $\mathcal{L}^{NCE}$ by summing over all the step-specific loss $\mathcal{L}_k^{NCE}$ defined in Equation 5.8 as following:

$$\mathcal{L}^{NCE} = \sum_{k=1}^{K} \mathcal{L}_k^{NCE} \tag{5.9}$$

The expectation $\mathbb{E}_{\tilde{z}\sim p_n}$ is approximated by sampling 10 negative examples by choosing distractors from the same audio sequence based on a uniform distribution: $p_n = \frac{1}{T}$, with T denotes the sequence length ($T = |z|$).

Schneider et al. (2019a)'s wav2vec fully relies on convolutional architecture at both encoder and context network. Their models, which are different in sizes, are pre-trained on different combinations of raw audio training data (*Wall Street Journal (WSJ)* (Garofalo et al. (2007)) and *Librispeech* (Panayotov et al. (2015))) in order to simulate different scenarios where unlabelled training data is either scarce or abundant. After the pre-training of these models, the output c_i of the context network is used as input for downstream ASR task instead of surface representations such as log-mel filterbank features. Their best system manages to outperform the state-of-the-art character-based ASR system (Amodei et al. (2016)), improving the WER by 0.67% (from 3.1% to 2.43%) on the WSJ benchmark. They also show that this approach helps them reach the best reported result in the literature on *TIMIT* (Garofolo et al. (1993)). Furthermore, their experiments show that wav2vec representations also outperform the baseline using log-mel filterbank features even in the simulated low-resource scenario when only 8 hours of annotated audio data is available, and increasing the amount of pre-training unlabelled data also helps improve the performance further.

5.3.3 vq-wav2vec

Baevski et al. (2020a) propose *vq-wav2vec* which extends wav2vec representations, learning *Vector Quantized (VQ)* representations of audio data through solving the same future time-step prediction task. Their motivation consists of observations indicating that discrete representations are potentially a more natural fit for many modalities, for instance, language is inherently discrete, and speech can usually be represented by a sequence of symbols. Furthermore, discretization of audio makes it easier to exploit a great load of text-based approaches, which are less data expensive, for speech. Specifically in this work, BERT representations (Devlin et al. (2018)) trained on the discretized unlabelled speech data are shown to be better than log-mel filterbank as well as wav2vec representations on some benchmarks.

Figure 5.4 illustrates the vq-wav2vec architecture, which is an extension of the wav2vec architecture drawn in Figure 5.3. Similar to the wav2vec architecture, it contains two convolutional networks including $g_{enc} : \mathcal{X} \rightarrow \mathcal{Z}$

responsible for encoding a raw audio input sequence x to a latent representation z, and $g_{gr} : \hat{\mathcal{Z}} \rightarrow \mathcal{C}$ aggregating multiple time-steps of the preceding network to obtain contextualized representations c. The network that precedes the context network g_{gr} is a new *quantization* module $q : \mathcal{Z} \rightarrow \hat{\mathcal{Z}}$ responsible for building discrete representations. The quantization module is what makes vq-wav2vec distinguishes itself from the wav2vec architecture. This module (depicted as q in Figure 5.4) turns the dense representations z (outputted from the encoder network) into discrete indices $\hat{z}$ drawn from a fixed-size codebook $e \in \mathbb{R}^{V \times d}$ containing V representation vectors of size d. Next, instead of z, $\hat{z}$ is fed into the context network g_{ar} and a similar context prediction objective is optimized (Equation 5.10).

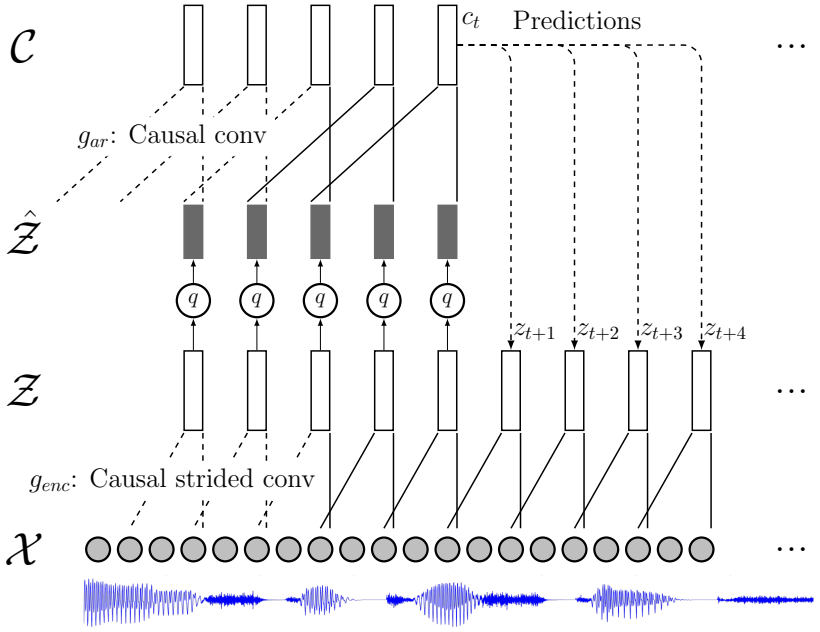


Figure 5.4: Illustration of the vq-wav2vec framework, which consists of an encoder mapping raw audio $\mathcal{X}$ to a dense representation $\mathcal{Z}$ which is then quantized (q) to $\hat{\mathcal{Z}}$ and aggregated into context representations $\mathcal{C}$.

$$\mathcal{L}_k^{NCE} = -\sum_{t=1}^{T-k} \left(\log \sigma(\hat{z}_{t+k}^{\top} h_k(c_t)) + \lambda \mathbb{E}_{\tilde{z} \sim p_n} [\log \sigma(-\tilde{z}^{\top} h_k(c_t))] \right) \tag{5.10}$$

In their original work, Baevski et al. (2020a) consider two quanization approaches for choosing the codebook variables differentiablely, including *Gumbel-Softmax* approach similar to (Jang et al. (2016)) and online k-means clustering

similar to the *Vector Quantised Variational AutoEncoder (VQ-VAE)* (Oord et al. (2017)). However, Baevski et al. (2020a) recognize the collapse mode issue (i.e. only some of the codewords would actually be used), and propose to perform multiple vector quantizations over different parts of z (similar to *Product Quantization* (Jegou et al. (2010))) for mitigating this problem.

The most interesting point of this approach is that it not only provides quite a handful of choices for extracting latent features from speech (i.e. one can either use the output z of the encoder network, the contextualized output c of the context network (similarly used by Schneider et al. (2019a)), or the learned discrete units outputted by the quantization module referred to as the vq-wav2vec features (Baevski et al. (2020a))), but also offers the possibilities for applying other NLP algorithms that require discrete inputs. In Baevski et al. (2020a)'s work, a BERT [1] model is trained on the discretized representations of audio from the same unlabelled training data. This pipeline of speech representation learning (briefly depicted in Figure 5.5) helps them obtain a new state-of-the-art on the TIMIT phoneme classification and WSJ ASR task.

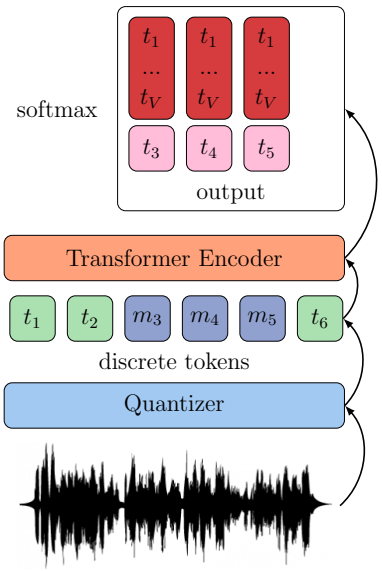


Figure 5.5: Illustration of the standard BERT pre-training over learned vq-wav2vec discrete units Baevski et al. (2019)

Baevski et al. (2019) reaffirm the advantages of vq-wav2vec coupled with BERT pre-training for the ASR downstream task. The standard BERT pre-training over learned vq-wav2vec discrete units (Figure 5.5) outperforms other investigated speech features including spectral MFCC or fbank features or fea-

[1] BERT (Devlin et al. (2018)) is a pre-training approach for text-based NLP tasks using a Transformer encoder model to build text representations.

tures obtained from the proposed BERT-style model learning directly from the continuous audio data (Baevski et al. (2019)) on the Librispeech benchmark. Furthermore, this work goes further to fine-tune the BERT model in a supervision manner with labelled ASR data. This is done by stacking a linear projection layer initialized randomly on top of the pre-trained BERT model (depicted by *"Transformer Encoder"* in Figure 5.5) to solve the ASR task on different subsets of Librispeech and *Libri-light* (Kahn et al. (2020)). These subsets are randomly sampled with different respective sizes of 10 minutes, 1 hour, 10 hours, and 100 hours to better highlight the impact of fine-tuning. These fine-tuned models are optimized by minimizing the *Connectionist Temporal Classification (CTC)* loss (Graves et al. (2006)). Their findings are that fine-tuning (even with only 10 minutes of labelled data) is enormously beneficial, WER scores drop when the size of the fine-tuning subset increases, and fine-tuning on 100 hours of labelled data helps beat the best known result on *test-other* of Librispeech while relying on much less labelled data.

5.3.4 wav2vec2.0

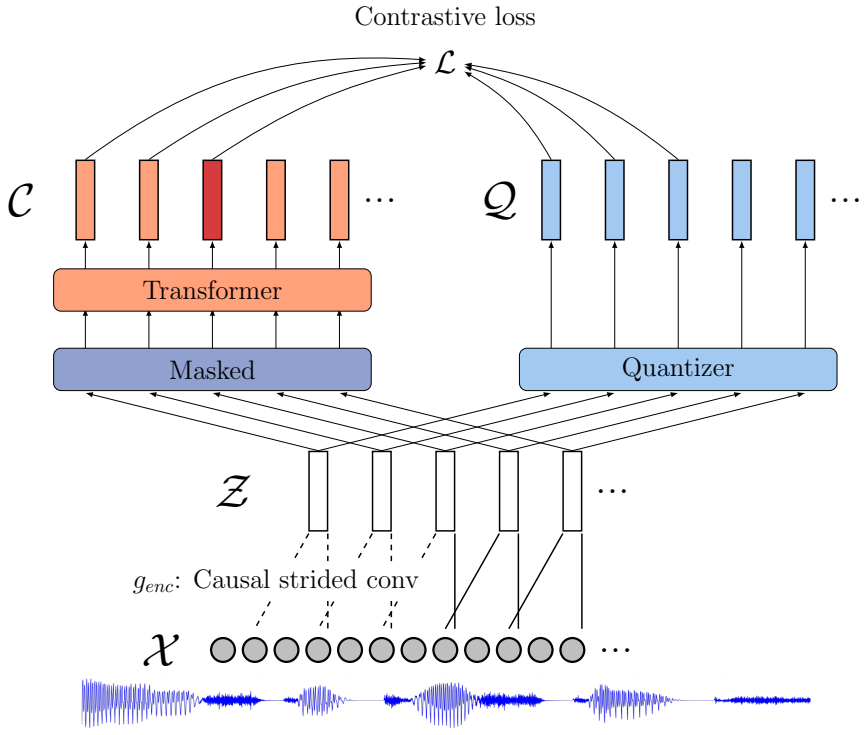


Figure 5.6: Illustration of the wav2vec2.0 framework, which jointly learns contextualized speech representations and an inventory of discretized speech units.

Wav2vec2.0 proposed by Baevski et al. (2020b) is yet another extension of Schneider et al. (2019a); Baevski et al. (2020a, 2019). Depicted in Figure 5.6, similar to the wav2vec framework, it consists of a multi-layer convolutional feature encoder $g_{enc} : \mathcal{X} \rightarrow \mathcal{Z}$, which transforms raw input audio x into latent speech representations $z = \{z_1, z_2, ..., z_T\}$ for T time-steps. These latent features are then fed into a Transformer $g : \mathcal{Z} \rightarrow \mathcal{C}$ for building contextualized representations $c = \{c_1, c_2, ..., c_T\}$ that capture the infomation of the whole sequence. This is different from the context network of both the wav2vec and vq-wav2vec framework, which is modeled by a multi-layer convolutional neural network usually downsampling their input. Similar to vq-wav2vec, wav2vec2.0 also performs discretization on the output of the feature encoder z_t to q_t by using a quatization module $\mathcal{Z} \rightarrow \mathcal{Q}$. However, the difference between these two frameworks is that Baevski et al. (2020a) refer to their contextualized representations as the output of a pipeline of their vq-wav2vec framework followed by a BERT-like pre-training on discrete speech units. By contrast, wav2vec2.0's Transformer network learns contextualized representations directly from continuous speech representations (z) via time-step masking and constrative task which identifies the true quantized latent audio representation in a set of distractors for each masked time step. This consequently allows Baevski et al. (2020b) to train wav2vec2.0 in an end-to-end fashion, in which all the components of the presented architecture are trained jointly toward minimizing an objective (Equation 5.11, 5.12, 5.13), rather than cascading the training procedure as in Baevski et al. (2020a).

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{5.11}$$

$$\mathcal{L}_m = -\log \frac{\exp(sim(c_t, q_t)/\kappa)}{\sum_{\tilde{q} \sim Q_t} \exp(sim(c_t, \tilde{q})/\kappa)} \tag{5.12}$$

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{5.13}$$

In Equation 5.11, the training objective is defined as the sum of two components:

- **Contrastive Loss $\mathcal{L}_m$**: is defined in Equation 5.12. Particularly, given c_t centered over the masked time step t, the model is trained to contrast the true quantized latent speech representation q_t from K quantized latent distractors $\tilde{q} \in Q_t$ uniformly sampled from other masked time steps of the same utterance. $sim(a, b) = a^\top b / ||a|| ||b||$ measures the *cosine similarity* between context representations c_t and quantized latent speech representations q_t.

- **Diversity Loss $\mathcal{L}_d$**: which is defined in Equation 5.13 helps increase the use of the quantized codebook representations, encouraging the model to equally use all the V entries in each of G codebooks by maximizing the entropy of the averaged softmax distribution over the codebook entries for each codebook $\bar{p}_g$ across a batch of utterances (Dieleman et al. (2018)). In Equation 5.11, $\mathcal{L}_d$ is scaled by α, which is a tunable hyper-parameter.

Masking: time-step masking mentioned earlier is done by randomly sampling without replacement a certain proportion p of all time steps to be starting indices and then mask the subsequent M consecutive time steps from every sampled index. Note that spans may overlap, and inputs to the quantization module are not masked (the right side of Figure 5.6).

Fine-tuning: similar to Baevski et al. (2020a)'s vq-wav2vec, wav2vec2.0 framework also allows fine-tuning the pre-trained model directly on ASR labelled data by stacking a linear projection layer initialized randomly on top of the context network. Fine-tuned models are optimized by minimizing a CTC loss. They show that fine-tuning even on only 10 minutes of labeled training data (48 recordings of 12.5 seconds on average) helps achieve respective WER of 4.8 and 8.2 on test-clean, and test-other of Librispeech.

5.3.5 Other approaches

Beside CPC, other SSL approaches for learning useful speech representations have been proposed, some of which are presented as follows:

Autoregressive Predictive Coding (*APC*): inspired by language modeling, Chung et al. (2019); Chung and Glass (2020b) propose to use APC for inferring a future frame x_{t+n} (of a full utterance $x = (x_1, x_2, ..., x_N)$ of length N) that is n steps ahead of x_t. This is done by using an autoregression RNN to process each frame x_t sequentially to make prediction about y_t, which has the same dimension as x_t, for $t = 1, 2, ..., N-n$. The model is trained to minimize the frame-wise L1 loss between the predicted sequence $y = (y_1, y_2, ..., y_{N-n})$ and the target sequence $(x_{1+n}, x_{2+n}, ..., x_N)$ (Equation 5.14). Once this model is trained, the output of the last RNN layer $(h_1, h_2, ..., h_N)$ is taken as the extracted features for the input sequence $x = (x_1, x_2, ..., x_N)$.

$$L_f(x) = \sum_{t=1}^{N-n} |x_{t+n} - y_t| \tag{5.14}$$

Masked Prediction of Hidden Units: Hsu et al. (2021) propose *HuBERT* (dubbed for *Hidden Unit BERT*), which exploits a BERT model to predict pre-determined cluster assignments from masked continuous speech features.

In detail, firstly, discrete speech units are pre-computed by an offline $k-means$ model (Lloyd (1982)). Secondly, from the masked input sequence, a BERT model is exploited to produce a feature sequence, which is then used (along with the pre-determined discrete sequence computed from the input sequence) to compute a cross-entropy loss. Their large model trained with iteratively refining k-means cluster assignments using learned latent representations consistently outperforms wav2vec2.0 of the comparable model size on the Librispeech and Libri-light (Kahn et al. (2020)) with $10min$, $1h$, $10h$, $100h$, and $960h$ fine-tuning subsets. Moreover, they show that they can further improve the performance by scaling the model size up to the extra-large model of 1 billion parameters.

5.4 Conclusion

We review in this chapter two most commonly used conventional speech representation approaches, including Log Mel Filterbank and Mel-Frequency Cepstrum Coefficients. These methods selectively use mathematical tools to directly transform the digitized speech signals into a denser representation. Our focus in this chapter, however, is on Self-supervised Learning from Speech representations, which is a more recent approach for extracting speech features. More specifically, we present Contrastive Predictive Coding (CPC) which aims at learning useful representations from high-dimensional un-annotated data by predicting the future in latent space. This approach allows to leverage huge amounts of un-annotated data for learning useful representations from speech. These representations are then used in downstream tasks, for example, Automatic Speech Translation, which usually suffers from data craving. The different variants of CPC are presented in this chapter, namely, wav2vec, and two of its extensions including vq-wav2vec, which learns vector quantized representations and wav2vec2.0, which masks raw input speech in the latent space and solves a contrastive task defined over quantized speech representations.

Part II

Contributions

Chapter 6

Offline Neural Speech Translation

6.1 Introduction

In Chapter 2, we discuss in detail end-to-end neural approaches for automatic speech translation. These approaches are distinguished from the conventional cascaded speech-to-text translation, which operates in two steps: (1) source language speech recognition (ASR) and (2) source-to-target text translation (MT). By contrast, end-to-end approaches attempt to build direct translation systems without using source language transcription during learning or decoding (Bérard et al. (2016); Weiss et al. (2017)) or using it at training time only (Bérard et al. (2018)). Since these proof-of-concept works, end-to-end AST has been attaining great development velocity. As the result, an opulent resource of offline end-to-end speech translation neural architectures, development toolkit, frameworks and different speech corpora have been introduced. Therefore, investigations amongst these are of great importance for this thesis, which focuses on end-to-end AST. The investigations were carried out at the beginning of this thesis, building a foundation for other contributions that follow. This chapter is dedicated to telling the story about this quest, which results in several offline end-to-end AST systems in two different language pairs English-Portuguese (En→Pt), and English-German (En→De) described in two system description papers accepted for presentation at the IWSLT 2019 (Nguyen et al. (2019)) and IWSLT 2020 (Elbayad et al. (2020b)) workshop respectively:

- M. Elbayad, H. Nguyen, F. Bougares, N. Tomashenko, A. Caubrière, B. Lecouteux, Y. Estève, and L. Besacier, ***"ON-TRAC Consortium for End-to-End and Simultaneous Speech Translation Challenge Tasks at IWSLT 2020"***, in The International Conference on Spoken Language Translation ACL - 17th IWSLT, Seattle, WA, United States, Jul. 2020.

- Nguyen, H., Tomashenko, N., Boito, M. Z., Caubrière, A., Bougares, F., Rouvier, M., Besacier, L., and Estève, Y. ***"ON-TRAC consortium end-to-end speech translation systems for the IWSLT 2019 shared task"***, *International Workshop on Spoken Language Translation (IWSLT 2019).*

This chapter is organized as to emphasize our contributions in the context of these two evaluation campaigns, firstly talking about our work for IWSLT 2019, and then describing the details of our work for IWSLT 2020.

6.2 Translation systems for the IWSLT 2019

Until 2019, The International Workshop on Spoken Language Translation (IWSLT) is an annual scientific workshop, associated with an open evaluation campaign on spoken language translation. In the workshop, both scientific papers and system descriptions are presented. IWSLT 2019[1] is the 16th workshop, taking place in Hong Kong. The scope of this campaign is limited to speech-to-text translation systems of either cascaded or end-to-end fashion. Submissions to this campaign are required to translate (automatically) English audio data extracted from TED talks into German or Portuguese. End-to-end models are defined by the organizers of the campaign to possess the following properties (Jan et al. (2019)): (1) *"Do not exploit intermediate discrete representations (e.g., source language transcription or hypotheses fusion in the target language)"*, and (2) *"Rely on parameters that are all jointly trained on the end-to-end task"*.

Participating in this evaluation campaign, we develop several translation systems for the end-to-end model task for the En→ Pt language pair. These systems consist of both cascaded and end-to-end models based on different granularities (BPE or characters). Our end-to-end models, described in detail in the coming sections, are based on encoder-decoder architecture with attention mechanism. The ultimate goal of this participation is to answer the following scientific questions:

- Question 1: Does pooling heterogeneous corpora (How2 and MuST-C) help the AST training?
- Question 2: What is the better tokenization unit on the target side (BPE or characters)?
- Question 3: Considering that segmentation is an important challenge of AST, what is the optimal way to segment the speech input?
- Question 4: Does fine-tuning increase the system's performance?
- Question 5: Is our end-to-end AST model better than an ASR+MT pipeline?

[1] `https://workshop2019.iwslt.org/`

6.2.1 Data

Choosing to develop translation systems which translate English speech into Portuguese text, we only concern ourselves with the speech corpora of this language pair. In particular, How2 (Sanabria et al. (2018)) and MuST-C (Di Gangi et al. (2019a)) corpora are used in our work. Since we focus on En→Pt AST tasks, only the En→Pt portion of MuST-C corpus is taken. The statistics of these two corpora, along with the corresponding provided evaluation data, can be found in Table 6.1. In order to answer the first scientific question mentioned earlier, we pool these two corpora together to create a merged corpus whose details can also be found in the same table.

Corpus	**#Segments**	**Hours**	**#src words**	**#tgt words**
MuST-C	206,155	376.8	3.9M	3.7M
How2	184,624	297.6	3.3M	3.1M
Merged corpus	390,779	674.4	7.2M	6.8M
MuST-C eval	2,571	5.4	-	-
How2 eval	2,497	4.5	-	-

Table 6.1: Statistics of the original MuST-C and How2 corpora, the merged version, and the official evaluation data (audio data only).

We note that the statistics for the How2 training set might slightly differ from that of the original paper (Sanabria et al. (2018)) because the original audio files of the How2 corpus were not made officially available for some technical reasons. The full corpus was only provided as pre-extracted acoustic features (40-dimensional filter bank features). Aiming to apply our own feature extraction instead, we needed to download the original video files from Youtube [2] and then extracted the audio from these downloaded video files. However, this raised one issue related to the availability of audio files on Youtube at the downloading date. In detail, on July 12th, when our version of the corpus was downloaded, 21 (out of 13,472) video files were missing. We consider this as a minor loss with regard to the possibility it gives us to extract our own acoustic features.

[2] `https://www.youtube.com/`

6.2.2 Speech segmentation

As stated in Chapter 2, end-to-end AST systems are usually trained on parallel corpora of source utterances paired with corresponding translation text. It is, therefore, important for the evaluation data to be segmented in the same manner in order to avoid mismatches between training and inference. In our case, whereas the organizers of the campaign provide a predefined segmentation for How2 evaluation data, this is not the case for the evaluation data related to TED talks. For this reason, two different approaches to segment the MuST-C (TED talks) audio stream are investigated:

- **Speaker Diarization-based approach**: uses LIUM_SpkDiarization toolkit (Meignier and Merlin (2010)), which is a well-known open-source toolkit for speaker diarization, to segment the input utterance. The default configuration is used for our purpose.

- **ASR-based approach**: uses an ASR system as a speech segmenter (Figure 6.1). In order to do this, firstly, all the validation and evaluation datasets are transcribed automatically and without segmentation with a Kaldi-based ASR system (Daniel et al. (2011)) trained on TEDLIUM 3 (Hernandez et al. (2018)) [3]. After being trained, this ASR system produces recognized words with timing information including start time and duration for each word, from which silence duration between two words (when silence or non-speech events occur) can be measured. In the final step, we set some thresholds based on which audio files can be split. In detail, for segments that have less than 40 words, if a silence between two words is higher than a threshold of 0.65 seconds, we split the audio file. In case the segment contains more than 40 words, this threshold is reduced to 0.15 seconds, in order to avoid exceedingly long segments. We particularly use these thresholds for the segment duration distribution in the evaluation data close to the one observed in the training data.

Table 6.2 summarizes statistics about segment duration on training data whose segmentation is provided by the organizers, and evaluation data, which is segmented by the presented approaches.

[3] In the context of the campaign, the use of some TEDLIUM 3 files is forbidden. These files have been removed before training the ASR system. We did not try to optimize the ASR system on our training data.

Corpus/Segmentation	Min size	Max size	Average	std dev
Train/Organizers	0.17	30.00	6.31	4.72
Eval/ASR-based	0.03	22.71	6.09	4.52
Eval/Speaker Diarization-based	1.51	20.00	9.62	5.33

Table 6.2: Statistics on speech segments duration (MuST-C) for 2 different segmentation approaches. All values are given in seconds.

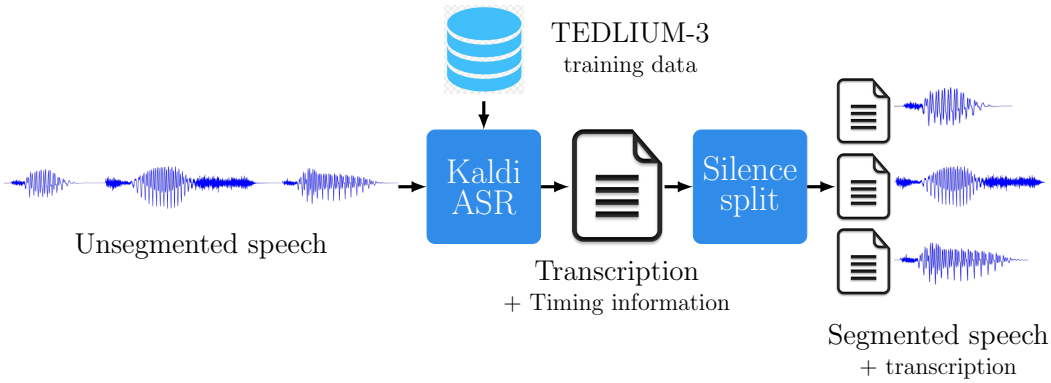


Figure 6.1: Illustration of ASR-based segmentation approach.

We choose the segmentation approach for our primary system amongst these two approaches based on the experiments on the MuST-C tst-COMMON test set. For these experiments, a preliminary version of our end-to-end system trained on the MuST-C training data is used to translate speech into lower-case text. After this, the *mwerSegmenter* tool [4] is used to realign our translations to the reference segmentation of the MuST-C tst-COMMON data, and the results are passed to the next step which measures the translation quality based on BLEU score. Table 6.3 compares the BLEU scores obtained with manual (original MuST-C annotations) with the presented segmentation strategies: ASR-based, and Speaker Diarization-based. It is clear from the table that the Speaker Diarization-based segmentation leads to inferior translation quality in comparison with its ASR-based counterpart. It is also notable that manual segmentation still outperforms our best automatic segmentation (25.50 against 22.03 BLEU), showing that automatic segmentation of the audio stream is an important issue for the speech translation task. This answer question 3 imposing in Section 6.2.

[4] `https://www-i6.informatik.rwth-aachen.de/web/Software/mwerSegmenter.tar.gz`

Segmentation	BLEU
Manual (original)	25.50
Speaker Diarization (Meignier and Merlin (2010))	21.01
ASR-based	22.03

Table 6.3: BLEU scores (lower-case evaluation) obtained on the tst-COMMON (MuST-C corpus) data with different speech segmentation strategies.

Based on these findings, the ASR-based segmentation approach is chosen for our primary system applied to the TED talks (MuST-C) evaluation data, whose segmentation information is not available. For the How2 evaluation data, we use the manual segmentation provided by the organizers.

6.2.3 Experimental setups

Participating in this workshop, this thesis contributed in exploring several deep learning frameworks for speech processing, including OpenNMT (Klein et al. (2017)) and ESPnet (Watanabe et al. (2018)). ESPnet was chosen for its flexibility, speech friendly, and superior results on several initial experiments. Based on ESPnet, we develop several speech translation systems including both a pipeline system that served as our baseline and end-to-end systems trained on different settings for the En→Pt language pair.

6.2.3.1 End-to-end speech translation

In this section, we detail the setups for training all our end-to-end models that have similar architectures and differ mainly in the following aspects: (1) training corpus, (2) type of tokenization units, and (3) fine-tuning and pre-training strategies.

Speech features: we apply the same 80-dimensional Mel filter-bank acoustic features concatenated with 3-dimensional pitch features [5] for training all our models. These features are extracted from $25ms$ windows with a frame shift of $10ms$, and normalized by cepstral mean and variance normalization computed on the training set. In terms of data augmentation, we use speed perturbation with factors of 0.9, 1.0, and 1.1 on training data (Ko et al. (2015)).

[5]Pitch-features are computed using the Kaldi toolkit (Daniel et al. (2011)) and consist of the following values (Ghahremani et al. (2014)): (1) probability of voicing (POV-feature), (2) pitch-feature and (3) delta-pitch feature. For details, see `http://kaldi-asr.org/doc/process-kaldi-pitch-feats_8cc.html`

Text preprocessing: we normalize punctuation, and tokenize all the Portuguese text using Moses normalization scripts [6]. The target text is case-sensitive and contains punctuation. Furthermore, we observe that the text of the MuST-C corpus contains non-speech events, for example, *"Laughter"*, *"Applause"* marks. We keep these when training the model on MuST-C data only, but remove them from the text when training the models on the combination of both corpora to ensure consistency.

Development sets: are generated by randomly sampling $2,000$, $2,000$, and $4,000$ sentences from MuST-C, How2 and the merged corpus respectively. These sentences are removed from the corresponding training sets.

Data filtering: in order to make the training feasible with our limited computational resources, long sentences that exceed $3,000$ frames ($\approx 30s$) or 400 characters are removed from the training and the development set. Consequently, we suffer a minor loss of 6%, 8% and 7% of speech data for How2, MuST-C and the merged corpus respectively.

The summarization of the training data after preprocessing can be found in Table 6.4.

Set	#Segments	#src words	#tgt words
MuST-C train	597,871	10.9M	10.3M
MuST-C dev	1,994	36.4K	34.4K
How2 train	538,231	9.4M	8.9M
How2 dev	1,984	33.7K	32.0K
Merged train	1,136,084	20.9M	19.2M
Merged dev	3,978	72.4K	66.5K

Table 6.4: Statistics for the training data after preprocessing.

Architecture: our end-to-end AST models are based on an attention-based encoder-decoder architecture (Figure 6.2), whose encoder has two VGG-like (Simonyan and Zisserman (2014)) CNN blocks. Each VGG block contains two $2D$-convolution layers followed by a $2D$-maxpooling layer whose aim is to reduce both time (T) and frequency dimension (D) of the input speech features by a factor of 2. These two VGG blocks transform the input speech features' shape from $(T \times D)$ to $(T/4 \times D/4)$ with T is the duration of input speech represented by the number of speech frames, and D is the dimension of speech features. After these VGG-like blocks, we stack five 1024-dimensional BLSTM layers. The decoder has two 1024-dimensional LSTM layers. Bahdanau et al.

[6] `http://www.statmt.org/moses/`

(2015)'s attention mechanism is used in all our experiments to bridge the encoder and the decoder.

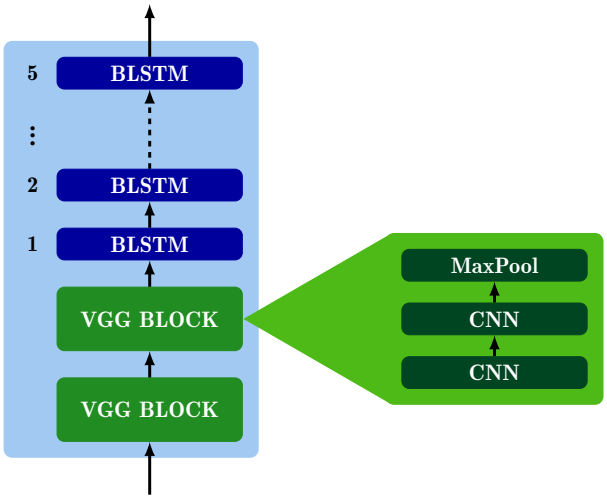


Figure 6.2: Architecture of the speech encoder used in Nguyen et al. (2019): a stack of two VGG-like CNN blocks followed by five BLSTM layers. Each VGG block contains two $2D-$convolution layers followed by a $2D-$maxpooling layer whose aim is to reduce both time (T) and frequency dimension (D) of the input speech features by a factor of 2. These two VGG blocks transform input speech features' shape from $(T \times D)$ to $(T/4 \times D/4)$.

Hyperparameters' details. In all our experiments, dropout is set only on the encoder part with the probability of 0.3. Adadelta is chosen as our optimizer.

6.2.3.2 Cascade baseline

Beside the development of end-to-end AST models, in this work, we also develop a pipeline system of an ASR model coupled with an MT model, that serves as our baseline system. This pipeline approach is described in detail in this section.

ASR system: is built by the Kaldi speech recognition tookit. Our ASR system is similar to the one built from the *tedlium/s5_r3* recipe [7]. We train our acoustic model on TEDLIUM-3 (Hernandez et al. (2018)) and a subset of the MuST-C corpus. We use TDNN-F (11 TDNN-F layers) structures for acoustic modeling with 40-dimensional MFCC features. As for the language

[7] `https://github.com/kaldi-asr/kaldi/tree/master/egs/tedlium/s5_r3/`

model (LM), a simple 3-gram LM is trained using TEDLIUM-3, MuST-C and How2 corpus, with SRILM toolkit (Stolcke (2002)). With this ASR system, we achieve a case-insensitive WER of **21.71%** and **26.89%** on MuST-C tst-COMMON and How2 val set respectively.

MT system: is built based on Transformer model as implemented in `fairseq` (Ott et al. (2019)), which is the state-of-the-art model for MT. Our models are based on the small Transformer settings which have, at both the encoder and the decoder side, 6 Transformer layers with an embedding layer of size 512, a feed-forward layer with an inner dimension of 1024, and 4 heads for the multi-head attention layers. This NMT system is trained on the merged corpora (Table 6.1) with a vocabulary of $30K$ units based on a joint source and target BPE.

Results of the pipeline speech translation system are reported in the last line of Table 6.5.

Evaluation set	ASR	Ref
How2 val	34.23	51.37
MuST-C tst-COMMON	22.14	28.34

Table 6.5: Detokenized Case-sensitive BLEU scores for different evaluation sets when translating the automatic (ASR) and human (Ref) transcription.

6.2.4 Experiments and results

In this section, we describe our experiments and the corresponding results, which help answer the scientific questions mentioned earlier in this chapter.

6.2.4.1 Question 1: choosing the training corpus

We train three character-based end-to-end models with the architecture described in the previous section using three different training corpora: (1) MuST-C, (2) How2, and (3) the merged version of the two corpora. These models are then evaluated on the MuST-C tst-COMMON, and the How2 val set. The results are reported in the first three lines of Table 6.6, from which we can observe that the model trained on the merged corpora outperforms the ones trained on MuST-C (difference of 3.32 BLEU) and How2 (difference of 3.11 BLEU). This model (line #3 of the table) is, therefore, used for our IWSLT primary system submission for both evaluation datasets.

No.	Experiment	Token	MuST-C tst-COMMON	How2 val
1	Must-C	char	23.59	-
2	How2	char	-	39.86
3*	Merged	char	**26.91**	42.97
4	Merged	BPE-400	24.73	**43.82**
5	Merged	BPE-2k	23.11	41.45
6	Merged	BPE-5k	22.25	41.20
7	Merged	BPE-8k	21.75	40.07
8	FT / Unfreeze	char	-	43.02
9	FT / Freeze	char	-	43.04
10	Pipeline (Table 6.5)	BPE-30k	22.14	34.23

Table 6.6: Detokenized case-sensitive BLEU scores for different experiments. Two lines with *FT* correspond to the models trained on the merged training corpus and fine-tuned (FT) using only the How2 corpus.

6.2.4.2 Question 2: choosing the tokenization units

This series of experiments are designed in order to investigate the impacts of the tokenization units on the performance of the translation system. Two types of tokenization units are investigated, including characters and subword units based on BPE. For this purpose, we train four additional BPE-based models with different vocabulary sizes of 400, 2000, 5000 and 8000. The results for the these models are given in Table 6.6, lines #4–7. It is notable from the table that having fewer output tokens on the decoder side is beneficial. For this reason, we conclude that, in our settings, characters seem to be the best tokenization units on the MuST-C, and BPE-400 units provide the best results for the How2 task [8].

6.2.4.3 Question 3: segmentation

We have seen in the earlier section that our ASR-based segmentation leads to better BLEU scores than the off-the-shelf Speaker Diarization-based segmentation approach. For this reason, we use the ASR-based segmentation approach for processing TED talks for our primary system, while we use the Speaker Diarization-based approach for a contrastive system.

[8]However, since the BPE-400 result for How2 was obtained after the evaluation deadline, our official submission uses characters for both MuST-C and How2 datasets.

6.2.4.4 Question 4: fine-tuning impact

In this work, we also investigate fine-tuning scenarios where we extend the training of the model which uses the merged corpora (line #3 in Table 6.6) for one more epoch on the How2 corpus only (our evaluation target). The fine-tuning scenarios can be either: (1) fine-tuning both the encoder and the decoder (denoted as *Unfreeze* on line #8 of Table 6.6) and (2) fine-tuning the decoder only (denoted as *Freeze* on line #9 of the same table). It can be seen from line #8 and #9 of Table 6.6 that there is a slight but not significant gain with fine-tuning and no difference between *Freeze* and *Unfreeze* options.

6.2.4.5 Question 5: pipeline or end-to-end

The results of the pipeline model for both corpora are available in the last line (#10) of Table 6.6. It is clear from the table that our best end-to-end speech translation results (lines #3 and #4) outperform this baseline model by a difference of 4.77 points for TED talks and 9.59 points for How2. However, it is important to mention two following points: (1) we do not fully optimize the ASR, NMT system and their combination, and (2) these results still highlight the performance of our end-to-end speech translation systems.

6.2.4.6 System submission and official results

As mentioned earlier, we use the ASR-based segmentation approach for automatically segmenting the campaign's official testset *tst2019*, and after that, we use model 3∗ of Table 6.6 in order to automatically generate hypothesis translation for our primary submission. The official results of this submission are shown in Table 6.7

Testset	BLEU	TER	BEER	characTER	BLEU(CI)	TER(CI)
TED En→Pt	24.57	67.92	49.16	52.33	25.87	65.41
How2 En→Pt	44.08	39.94	64.22	31.27	44.55	39.31

Table 6.7: IWSLT 2019's official results of our primary submission (Jan et al. (2019)).

We highlight that our result on the TED En→Pt testset (line 1 of Table 6.7) ranks first in this campaign, outperforming the ranked-second system by a large margin (24.57 versus 9.95). Our result on How2 En→Pt testset (line 2 of Table 6.7) ranks second, standing behind a cascaded system which scores 47.86 on the same testset. The detailed ranking can be found in Jan et al. (2019).

6.3 Translation systems for the IWSLT 2020

In 2020, the 17th IWSLT [9] becomes a part of the ACL2020 conference [10]. The major change of this workshop, however, is the expansion in terms of challenge tracks, with the introduction of four new tracks including *Simultaneous Speech Translation*, *Video Speech Translation*, *Open Domain Translation* and *Non-native Speech Translation*, in addition to the *Offline Speech Translation* and the *Conversational Speech Translation* track which existed in the previous edition (Ansari et al. (2020)). We participate in this workshop in two different evaluation tracks: end-to-end consecutive (offline) speech translation, and simultaneous (online) speech translation (Elbayad et al. (2020b)). This chapter concentrates mainly on the distributions in the offline speech translation track, in which participants are required to translate (automatically) English audio data extracted from TED talks into German text. In this campaign, we only focus on developing end-to-end AST models, although cascaded systems are allowed by the organizers. We develop end-to-end models based on encoder-decoder with an attention mechanism, trying to investigate the impacts of data augmentation and ensembling of multiple models on the translation quality.

Name	#segments	Total length (in hours)
MuST-C train	229.703	400
MuST-C dev	1.423	2.5
MuST-C tst-COMMON	2.641	4.1
MuST-C tst-HE	600	1.2
Europarl train	32.628	77
Europarl dev	1.320	3.1
How2 synthetic	176.564	285.5
tst2019	2.813	5.1
tst2020	2.263	4.1

Table 6.8: Statistics of training and evaluation data. The statistics of tst2019 and tst2020 are measured on the segmented version provided by IWSLT2020 organizers.

[9] `http://iwslt.org/doku.php`
[10] `https://acl2020.org/`

6.3.1 Data

For all the experiments presented in this section, we rely on MuST-C En→Ge (hereafter called MuST-C original to distinguish between the synthetic version presented shortly after) and Europarl-ST (Iranzo-Sánchez et al. (2020)) En→Ge as our main corpora. Besides, we automatically translate (into German) the English transcription of MuST-C and How2 in order to augment the training data. This results in two synthetic corpora, which are called *MuST-C synthetic* and *How2 synthetic* respectively. The statistics of these corpora, along with the provided evaluation data, can be found in Table 6.8. We experiment with different ways of combining those corpora. The details of these experiments are presented later in this section.

6.3.2 Experimental setups

Speech segmentation: the same as our presented works for IWSLT 2019 (Section 6.2), we reuse the ASR-based segmentation approach for acquiring the segmentation information of the evaluation and development data, along with the segmentation provided by the IWSLT organizers. These two segmentation types are dubbed as "ASR seg" and "IWSLT seg" in Table 6.9 and 6.10 respectively.

Speech features and data augmentation: we follow exactly the same procedure as presented in Section 6.2 in order to extract the acoustic features for our models (mean and variance normalized 80-dimensional Mel filter-bank features, concatenated with 3-dimensional pitch features). In terms of data augmentation, beside concatenating different corpora presented earlier, we use speed perturbation with factors of 0.9, 1.0, and 1.1. We also apply SpecAugment (Park et al. (2019)) to the training data. All three SpecAugment methods are used, including time warping ($W = 5$), frequency masking ($F = 30$), and time masking ($T = 40$).

Text preprocessing: the same as in Nguyen et al. (2019), we normalize punctuation marks, and tokenize all the German text using Moses. Target text is case-sensitive and contains punctuation marks. Furthermore, the non-speech events (*i.e "Laughter"*, *"Applause"*, etc.) are also removed from the target text. This results in a vocabulary of 201 characters. We later find that some of these characters should not appear in the German text, for example, ♪, 你, 葱, 送, etc. Therefore, they are manually excluded from the vocabulary. Finally, we obtain an output vocabulary of 182 characters.

Architecture: we reuse our attention-based encoder-decoder architecture presented in Section 6.2. We would like to mention that Transformer-based models have also been tested and showed weaker results compared to the

LSTM-based encoder-decoder architecture. For this reason, we keep using the aforementioned LSTM-based architecture for our experiments.

Hyperparameters' details: all of our models are trained in maximum 20 epochs, with early stopping being set so that the training would stop after 3 epochs if it shows no improvement in terms of accuracy on the development set. Dropout is set to 0.3 on the encoder part, and Adadelta is chosen as our optimizer. Parameters for inference are carefully tuned for our settings. In particular, we use a beam size of 10, and we prevent the models from generating too long sentences by setting a $maxlenratio$[11] $= 1.0$.

In summary, all our end-to-end models have similar architecture, and differ mainly in the following aspects: (1) training corpora, (2) type of tokenization units and (3) fine-tuning and pre-training strategies.

6.3.3 Experiments and results

We discussed in Section 6.2 the benefit of combining different corpora for augmenting our training data. For this reason, in this evaluation campaign, we continue exploring different combinations of different corpora as well as the use of synthetic data. Our baseline model is trained on the combination of [MuST-C original + Europarl-ST], whose target translation is original. The first two rows of Table 6.9 show that combining How2 synthetic with the baseline data set [MuST-C original + Europarl-ST] does not yield significant improvement. It can be clearly seen that this data pool is even worse than the baseline on both tst2015 (IWSLT seg) and tst2015 (ASR seg). However, line #3 shows that applying SpecAugment on this same combination helps outperform the baseline on every investigated test set, with substantial gain in terms of BLEU scores can be found on both MuST-C tst-COMMON and MuST-C tst-HE. For this reason, we consistently apply SpecAugment to all the remaining experiments. Surprisingly, we find that adding MuST-C synthetic to this data combination leads to the shrinkage of BLEU scores on both the MuST-C test sets, yet at the same time substantially increases the scores on both tst2015 (IWSLT seg) and tst2015 (ASR seg). However, due to the time constraint of the evaluation campaign, we are not able to investigate further on this matter. Therefore, instead of choosing model 4* for fine-tuning, we choose model 3*, which, as we can see, yields fairly good performance amongst all the test sets. This model is fine-tuned on MuST-C original and MuST-C original+synthetic. Table 6.9 shows that the impact of fine-tuning is very limited. Furthermore, it can be noticeable that adding MuST-C synthetic does not make much difference. Finally, the results of ensembling all six models at

[11] $maxlenratio = \frac{maximum\_output\_length}{encoder\_hidden\_state\_length}$

decoding time are indicated on the last row of the table. We can see clearly that ensembling yields the best BLEU scores across all the test sets.

No.	Experiment	MuST-C tst-COMMON	MuST-C tst-HE	tst2015 (iwslt seg)	tst2015 (ASR seg)
1	MuST-C original + EuroParl	20.18	19.82	12.59	14.85
2	MuST-C original + Europarl + How2 synthetic	20.51	20.10	12.10	13.66
3*	MuST-C original + Europarl + How2 synthetic	23.55	22.35	13.00	15.30
4*	MuST-C original + Europarl + How2 synthetic + MuST-C synthetic	22.75	21.31	14.00	16.45
5*	Finetune 3* on MuST-C original	23.60	22.26	13.71	15.30
6*	Finetune 3* on MuST-C original + MuST-C synthetic	23.64	22.23	13.67	15.29
7	**Ensemble (1 to 6)**	**25.22**	**23.80**	**15.20**	**16.53**

Table 6.9: Detokenized case-sensitive BLEU scores for different experiments - * represents experiments that apply SpecAugment.

In summary, Table 6.9 reaffirms two important points: (1) ensembling all six models is the most promising among all presented models, (2) our own ASR-based segmentation approach is better than the provided one. For these reasons, we choose as our primary submission the translations of the ASR-based segmentation generated by the ensemble of all six models. Model 3* and 4* (Table 6.9) are also used to translate our contrastive submission runs, whose ranks are shown in Table 6.10. The official results for all our submitted systems can be found in Table 6.11. They confirm that our proposed segmentation approach is beneficial.

Model	IWSLT seg	ASR seg
3*	constrastive5	constrastive3
4*	constrastive4	constrastive2
Ensemble	constrastive1	**primary**

Table 6.10: The ranking of out submitted systems. Model 3* and 4* are respectively corresponding to No.3* and No.4* of Table 6.9.

No.	Set	BLEU	TER	BEER	CharacTER	BLEU(ci)	TER(ci)
1	2019.contrastive1	17.57	71.68	47.24	58.03	18.64	69.66
2	2019.contrastive2	17.83	71.60	48.66	53.49	18.9	69.26
3	2019.contrastive3	19.03	66.96	49.12	54.10	19.97	65.01
4	2019.contrastive4	15.08	78.79	45.87	59.06	16.06	76.62
5	2019.contrastive5	15.87	74.17	46.18	59.96	16.86	72.15
6	**2019.primary**	**20.19**	**66.38**	**49.89**	**52.51**	**21.23**	**64.26**
7	2020.contrastive1	18.47	71.85	48.92	55.83	19.46	69.88
8	2020.contrastive2	19.31	69.30	49.55	52.68	20.36	67.14
9	2020.contrastive3	20.51	64.88	50.19	53.06	21.5	62.99
10	2020.contrastive4	15.48	83.45	46.68	57.56	16.42	81.33
11	2020.contrastive5	16.5	75.15	47.23	57.90	17.42	73.22
12	**2020.primary**	**22.12**	**63.87**	**51.20**	**51.46**	**23.25**	**61.85**

Table 6.11: IWSLT 2020 official results (offline track) on tst2019 and tst2020.

We note that our primary results on both the test sets fall behind the ranked-first system by 3.18 (on tst2019) and 3.77 (on tst2020) BLEU, which is trained on much more data with more complicated dual training Potapczyk and Przybysz (2020). The detailed ranking of the campaign can be found in Ansari et al. (2020).

6.4 Conclusion

In this chapter, we have presented our quest for finding end-to-end AST models that are most suitable for our condition. A series of experiments, which were carried out for our participation in two speech translation evaluation campaigns namely IWSLT 2019 and IWSLT 2020, agree that LSTM-based encoder-decoder with attention architecture performs well in our settings. We

also observe through experiments that combining speech translation corpora significantly helps improve the performance of the translation system. Experiments on both language pairs En→De and En→Pt show that the results are in favor of using characters as output tokens. Furthermore, ensembling is considerably beneficial, while fine-tuning shows limited or no improvement on the performance of the system.

From the presented results, character-based encoder-decoder with attention models will be consistently reused in our following experiments, which shall be presented in the remaining of this thesis. However, the readers of this thesis should be warned that, since this research field is proliferating with great velocity, this finding might not agree with more recent ones from other research groups.

Chapter 7

Self-supervised Learning Speech Representation

7.1 Introduction

In Chapter 2 we accentuated that parallel corpora, which is essentially required for training end-to-end AST models, are not as abundantly available as data for training end-to-end MT or ASR models. In the same chapter, several techniques that help mitigate this issue had been discussed including data augmentation, multi-task training and pre-training, which either endeavor to enlarge the scanty amount of AST data available on hand by different data manipulation approaches, or attempt to leverage training data from other tasks which can be more easily found. Chapter 5 discussed the advantages of Self-supervised Learning (SSL) from speech, which allows to leverage a substantial amount of unlabelled speech data for learning useful speech representations. Gathered evidence of the effectiveness of these kinds of representations when applied to other speech tasks (Chung et al. (2019); Chung and Glass (2020b); Oord et al. (2018); Baevski et al. (2019); Schneider et al. (2019a); Baevski et al. (2020b)) suggests that SSL might be a promising solution for the aforementioned data availability challenge imposed on end-to-end AST. One of the major contributions of this thesis is an in-depth study about the impact of self-supervised pre-training for end-to-end AST, which, to the best of our knowledge, is amongst the first attempts to utilize SSL speech representations for end-to-end AST. This work is presented at the Interspeech 2020 conference (Nguyen et al. (2020a)) and the SAS 2020 workshop [1]:

- **Ha Nguyen**, Fethi Bougares, Natalia Tomashenko, Yannick Estève, Laurent Besacier, ***"Investigating Self-supervised Pre-training for End-to-end Speech Translation"***, Interspeech 2020, Oct 2020, Shangai (Virtual Conf), China.
- **Ha Nguyen**, Fethi Bougares, Natalia Tomashenko, Yannick Estève, Laurent Besacier, ***"Investigating Self-supervised Pre-training for***

[1] ICML 2020 workshop on Self-supervision in Audio and Speech (SAS): `https://icml-sas.gitlab.io/`.

End-to-end Speech Translation", ICML 2020 workshop on Self-supervision in Audio and Speech (SAS), 2020.

Along with the investigation on the impacts of SSL pre-trained English model on end-to-end AST tasks involving English speech, this thesis also contributes directly to the training of French SSL models which are of immense importance for the following collective works:

- S. Evain, **H. Nguyen**, H. Le, M. Z. Boito, S. Mdhaffar, S. Alisamir, Z. Tong, N. Tomashenko, M. Dinarelli, T. Parcollet, et al., ***"Lebenchmark: A reproducible framework for assessing self-supervised representationlearning from speech"***, in Interspeech 2021, Brno, Czech Republic, 2021.

- Evain, S., **Nguyen, H.**, Le, H., Boito, M. Z., Mdhaffar, S., Alisamir, S., et al., ***"Task Agnostic and Task Specific Self-Supervised Learning from Speech with LeBenchmark"***, NeurIPS 2021.

- Solène Evain, **Ha Nguyen**, Hang Le, Marcely Zanon Boito, Salima Mdhaffar, Sina Alisamir, Ziyi Tong, Natalia Tomashenko, Marco Dinarelli, Titouan Parcollet, Alexandre Allauzen, Yannick Estève, Benjamin Lecouteux, François Portet, Solange Rossato, Fabien Ringeval, Didier Schwab and Laurent Besacier ***"Modèles neuronaux pré-appris par auto-supervision sur des enregistrements de parole en français"***, Les 34e Journées d'Études sur la Parole (JEP2022).

- Hang Le, Sina Alisamir, Marco Dinarelli, Fabien Ringeval, Solène Evain, **Ha Nguyen**, Marcely Zanon Boito, Salima Mdhaffar, Ziyi Tong, Natalia Tomashenko, Titouan Parcollet, Allauzen Alexandre, Yannick Estève, Benjamin Lecouteux, François Portet, Solange Rossato, Didier Schwab and Laurent Besacier, ***"LeBenchmark, un référentiel d'évaluation pour le français oral"***, Les 34e Journées d'Études sur la Parole (JEP2022).

This chapter is organized to emphasize these contributions. It discusses our in-depth study about the impacts of English SSL models for end-to-end AST (Section 7.2), before discussing different aspects of pre-training SSL models that specifically concern French speech (Section 7.3).

7.2 English SSL models for end-to-end AST

As stated in Chapter 5, SSL from speech consists in resolving pseudo-tasks, for instance, target predicting next samples or solving ordering problems, not

requiring human annotations as a pre-training for real tasks. Recent works on ASR suggest that the use of SSL is promising to reduce dependence on labeled data for building speech systems through acoustic representation learning. For these reasons, in this contribution, we investigate the possibility to leverage unlabeled speech for end-to-end AST. We steer our concentration on low-resource scenarios particularly where:

- ASR pre-training from the source language is not possible due to the fact that transcriptions of the source recordings are not available (the recordings might not be transcribed for some reasons or the source language in question is poorly written).
- Only a small-medium amount of parallel training data (speech aligned to translations) is available.
- There exists a larger amount of unlabeled speech, for example, there are situations when the system needs to translate from speech in a language with poorly standardized orthography or even from an unwritten language.

In summary, this section aims to accentuate the following contributions:

- We propose an in-depth study on the impact of self-supervised pre-training for AST.
- We show that self-supervised pre-training is particularly efficient in low-resource settings and that fine-tuning pre-trained representations on the AST training data is beneficial.
- We show that even in high resource settings, ensembling models trained with filter-bank and self-supervised representations leads to near state-of-the-art models without using ASR pre-training.
- We show through our analyses of the learned representations that they allow us to better discriminate phones, better align source and target sequences, and are more robust to speaker variability.

7.2.1 Pre-trained SSL models for English

One of the pre-trained English SSL models used in this work is an off-the-shelf wav2vec model [2] trained on LibriSpeech corpus. In order to investigate the benefit of fine-tuning on our task-specific data, we fine-tune this model on the

[2] `https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/`

full speech corpora used for our AST experiments presented shortly. We recall one important point that no transcripts nor translations are needed for this step which requires only raw speech.

These pre-trained SSL models for English, fine-tuned or not, are used as our features extractors for our downstream AST tasks. The speech representations are produced by the context network of the wav2vec model (Section 5.3.2) and are inputted to the AST encoder instead of filter-bank features.

7.2.2 Experimental setup

AST training data: How2 corpus is used for our main experiments [3]. This data is further filtered by stripping out too long sentences (sentences longer than 30 seconds or 400 characters). After this, the lower-resource scenarios are simulated by randomly splitting the corpus into four sub-corpora of roughly 10%, 20%, 30%, and 60% of the filtered full corpus. We split the corpus such that it guarantees that smaller partitions are fully included in the bigger ones. The statistics of all the partitions and the filtered version of the full corpus can be found in Table 7.1. As regards the development set, we reuse the one used for our participation in the IWSLT 2019 (Nguyen et al. (2019)). This comprises 1,984 sentences randomly excluded from the training set. How2 val set is used as our test data. As for target text processing, we normalize punctuation marks, and tokenize the text into character-level using Moses.

Partition	#segments	#hours	#src words	#tgt words
10%	17 751	28	313K	295K
20%	35 858	56	626K	591K
30%	53 698	84	887K	940K
60%	107 676	169	1778K	1883K
full	179 438	281	2963K	3139K

Table 7.1: Statistics of different How2 data partitions.

Speech features and data augmentation: from speech input, we extract different speech features [4] of either wav2vec-based or filter-bank+pitch features (later denoted as *fbanks*) as shown in Figure 7.1. Depending on the experiments, mean and variance normalization (MVN) is optionally applied

[3]Note that we reuse the version of How2 downloaded on July 12, 2019 Nguyen et al. (2019).

[4]Our preliminary experiments on How2 10% with MFCC features which lead to similar performance as filter-bank are not presented here.

to the generated features. For wav2vec feature extraction, we either use the off-the-shelf model trained on LibriSpeech or a model fine-tuned on How2 training set. MVN parameters are estimated on the speech translation training set and then applied to the training set, validation set, as well as the test set. In summary, we have four different self-supervised representations named *wav2vec*, *wav2vec + norm*, *wav2vec + FT* (fined-tuned wav2vec) and *wav2vec + FT + norm* (details can be found in Table 7.2). These representations are put into comparison with the conventional filter-bank features. Similar to Nguyen et al. (2019), filter-bank features are 80-dimensional Mel filter-bank features, concatenated with 3-dimensional pitch features from windows of $25ms$, and a frame shift of $10ms$. MVN is used in the same manner as for wav2vec features. This gives us two additional speech representations of dimension 83 named *fbanks* [5] and *fbanks + norm* shown on the last two lines of Table 7.2, respectively. Data augmentation through speed perturbation is also applied with factors of 0.9, 1.0, and 1.1 to the training data.

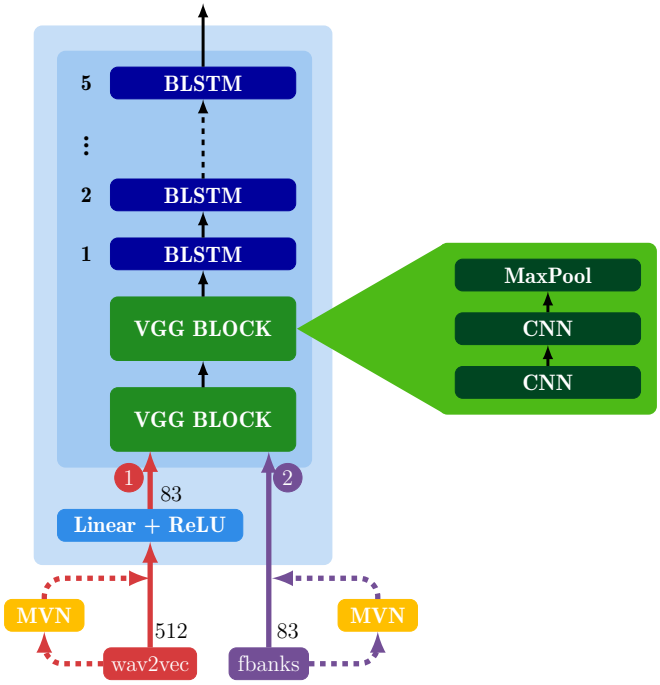


Figure 7.1: Architecture of the speech encoder: a stack of two VGG blocks followed by 5 BLSTM layers. We use as input (1) wav2vec features (that pass through an additional projection layer to reduce their dimension from 512 to 83), or (2) filter-bank+pitch features. The input features are optionally normalized (MVN).

[5]For the rest of the thesis *fbanks* will actually mean filter-bank+pitch.

AST model: we use an attention-based encoder-decoder architecture similar to the model that performs well at the IWSLT 2019 end-to-end AST track (Section 6.2). However, in order to deal with the high dimensionality of wav2vec features, we make a minor modification to the speech encoder as illustrated in Figure 7.1. In detail, in order to compare both input representations with a similar parameter budget in the architecture (and also because training an architecture with input features of dimension 512 would be more computationally expensive), we add a projection block at the bottom of the encoder [6]. This block contains a linear layer followed by a ReLU, aiming to reduce the dimension of wav2vec feature size from 512 to 83 (Figure 7.1). By contrast, when dealing with fbanks features, the model is exactly the same as Nguyen et al. (2019)'s model.

Name	Dimension
wav2vec	$512 \rightarrow 83$
wav2vec + norm	$512 \rightarrow 83$
wav2vec + FT	$512 \rightarrow 83$
wav2vec + FT + norm	$512 \rightarrow 83$
fbanks	83
fbanks + norm	83

Table 7.2: Different representations used in our experiments. "FT" stands for "fine-tuned", while "norm" means that mean and variance normalization is applied. Feature dimension of wav2vec is further projected to a smaller dimension ($512 \rightarrow 83$).

Hyperparameters' details: all the models are configured to be trained in maximum 20 epochs with early stopping after 3 epochs if the accuracy on the development set does not improve. Adadelta is chosen as optimizer and dropout is set to 0.3 on the encoder side. In inference time, we set the $beam\_size = 10$, and we $maxlenratio$ [7] $= 1.6$ to prevent the models from generating too long sentences. As stated earlier, all our end-to-end models are similar in terms of architecture except for the speech encoder which is slightly different between wav2vec and fbanks experiments. The main difference, however, lies in the following aspects: (1) the amount of training data; (2) speech

[6] Our implementation of the wav2vec speech encoder, as well as the detailed recipes for our experiments can be found online: `https://github.com/mhn226/espnet/tree/interspeech2020`.

[7] $maxlenratio = \frac{maximum\_output\_length}{encoder\_hidden\_state\_length}$

representations (wav2vec or fbanks); and (3) the use of MVN normalization or not.

7.2.3 Experiments and results

7.2.3.1 Experiments on How2

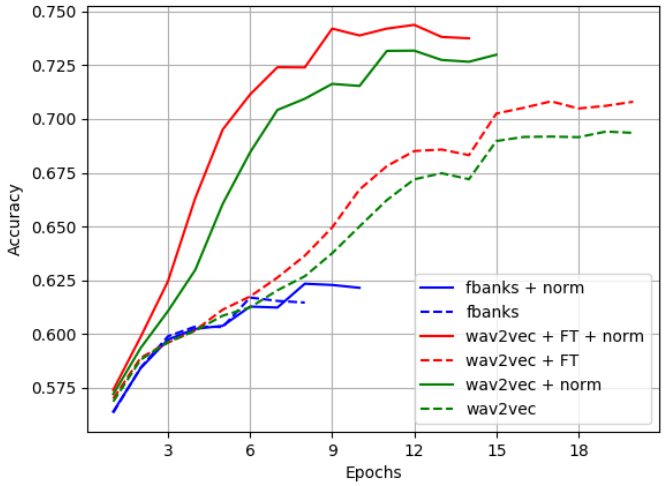
As stated earlier, on each partition of How2 corpus, we train 6 models which take as input different speech representations (Table 7.2), thus in total 30 models shown in Table 7.3. These models are evaluated on How2 val set, which contains 2,022 segments (about 3.2 hours of speech), in the same conditions as discussed in Section 6.2.

No.	Feature	10% (28h)	20% (56h)	30% (84h)	60% (169h)	100% (281h)
1	wav2vec	11.33	26.75	30.83	36.33	41.02
2	wav2vec + FT	12.52	27.30	32.11	37.78	42.32
3	wav2vec + norm	16.52	27.33	31.27	37.62	41.08
4	wav2vec + FT + norm	18.50	27.68	32.17	37.75	41.30
5	fbanks	1.03	18.61	27.32	37.23	41.63
6	fbanks + norm	2.11	24.58	30.21	37.56	42.51
7	Ensemble [5, 6]		25.28	31.90	40.39	44.35
8	Ensemble [4, 6]		29.87	34.67	41.22	45.02
9	**Ensemble [1,2,3,4,5,6]**		**31.88**	**36.80**	**42.62**	**46.16**

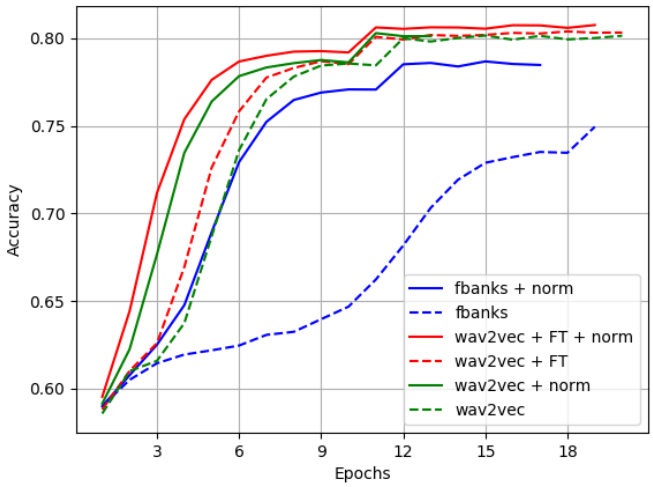
Table 7.3: Detokenized case-sensitive BLEU scores measured on How2 val set of different models trained on different partitions of How2 corpus (En→Pt) with different speech features. **FT** means fine-tuned and **norm** stands for MVN normalization.

The impact of SSL features on low-resource settings, where only 28 and 56 hours of speech data are available, can be clearly seen in the table. It shows that wav2vec features significantly outperform fbanks on these settings. Figure 7.2 confirms this and shows that models trained with wav2vec representations converge better and faster. It can be seen in Figure 7.2(a) that, in the extremely low-resource setting of 25 hours of speech, fbanks features (normalized or not) do not converge.

The impact of normalization and fine-tuning is also notable from both Table 7.3 and Figure 7.2. In very low-resource settings (like the case of 28 hours), fine-tuning wav2vec can greatly help, and with normalization, the performance further improves. These impacts of fine-tuning and normalization as well as the differences between wav2vec and fbanks fade away in higher-resource settings (169 and 281 hours of translated speech).



(a) How2 10% (28 hours)



(b) How2 20% (56 hours)

Figure 7.2: Learning curves (accuracy) of models trained on different partitions of How2.

However, our ensembling experiments show that it is beneficial to ensemble different models trained on different speech representations. In detail, lines 7 and 8 on 100% of How2 indicate that ensembling the best system (fbanks+norm, line 6) with a system trained with wav2vec (wav2vec+FT+norm, line 4) is more beneficial than with a better model (fbanks, line 5) also based on fbanks features, even though wav2vec+FT+norm underperforms fbanks

on this partition. The results on line 9 also indicate that the ensemble of all our models leads to $BLEU > 30$ even in very low-resource training conditions (56 hours).

Lang	Features	tst-COMMON	tst-HE
EN-DE	wav2vec	7.56	7.21
	wav2vec+norm	7.83	8.12
	fbanks	1.50	1.09
	fbanks+norm	4.89	4.87
EN-FR	wav2vec	12.08	12.41
	wav2vec+norm	12.58	12.58
	fbanks	0.54	0.00
	fbanks+norm	7.10	6.37

Table 7.4: AST BLEU on MuST-C 56 hours for En→De and En→Fr.

Lang	Features	tst-COMMON	tst-HE
EN-DE	wav2vec	10.57	10.43
	wav2vec+norm	10.30	10.27
	fbanks	0.74	0.66
	fbanks+norm	7.68	7.84
EN-FR	wav2vec	16.18	16.68
	wav2vec+norm	16.84	16.37
	fbanks	1.65	0.97
	fbanks+norm	14.31	13.86

Table 7.5: AST BLEU on MuST-C 84 hours for En→De and En→Fr.

Finally, in order to compare ourselves with the state-of-the-art of the time (Inaguma et al. (2020)), we decode How2 dev5 (a.k.a How2 test [8]), which consists of $2,305$ segments (about 3.7 hours of speech), using the ensemble of

[8]This test set is commonly used by some other works, for instance, Inaguma et al. (2020). However, as explained earlier, at the time we downloaded How2 corpus, several videos were missing from Youtube, including some files corresponding to this test set. Therefore, we took How2 val set to evaluate our models for IWSLT 2019 instead, and continue doing that in this work. Later, this full How2 dev5 set was provided directly by the authors of the corpus.

all our models trained on the full corpus (line 9). This gives us near state-of-the-art BLEU: we obtain 46.16 on How2 val and 47.17 on How2 dev5. This latter score on dev5 is to be compared with 48.04 reported with an ensemble model in Inaguma et al. (2020) where ASR and MT pre-training were used, as well as data augmentation with SpecAugment.

7.2.3.2 Validation on two other language pairs

In order to see if the effectiveness of SSL speech features can be generalized in low-resource settings of other language pairs (whose source speech is in English), we train our models on two subsets of MuST-C En→De and English-to-French (En→Fr) training data. These subsets are obtained by randomly sampling 56 and 84 hours from the corresponding MuST-C data of each language pair, simulating training sizes similar to How2 20% and 30%. Four different types of speech features are extracted from these subsets for training the same networks described earlier in this chapter (Table 7.4 and 7.5).

As illustrated by both the tables, MuST-C is more challenging than How2 (as aslo confirmed by the official IWSLT 2019 evaluation results (Jan et al. (2019))), but for both language pairs, wav2vec features significantly outperform fbanks. This confirms that self-supervised pre-training is useful in low-resource scenarios.

7.2.4 Analysis of Learnt Representations

The purpose of our analyses presented in this section is to answer the question of why wav2vec representations perform better than filter-bank features. These analyses shall reveal that wav2vec might be (1) better at discriminating phones, (2) better at aligning source and target sequences, and (3) more robust to speaker variability.

7.2.4.1 Better phone discrimination

We first replicate an experiment from Schneider et al. (2019b) for phoneme recognition on TIMIT (Garofolo et al. (1993)). Speech representations of four types wav2vec, wav2vec+norm, fbanks and fbanks+norm are extracted from train, dev and test split of TIMIT. These features are fed into a simple attentional encoder-decoder model consisting of an encoder with 4 BLSTM layers of hidden size 320, a decoder with 1 LSTM layer and location-based attention. The results of Table 7.6 confirm that wav2vec representations (normalized or not) are much better at recognizing phones than the fbanks counterparts.

No.	Feature	TIMIT dev	TMIT test
1	wav2vec	13.0	15.0
2	wav2vec + norm	13.9	15.8
3	fbanks	22.2	24.9
4	fbanks + norm	20.7	23.5

Table 7.6: Phone error rate (PER %) on TIMIT dev and test set.

7.2.4.2 Better source-target alignments

In this series of experiments, we evaluate the entropies of the soft alignments obtained with different speech representations in teacher forcing mode. The entropy of the probability distribution α_t is evaluated for every target token as in Equation 7.1:

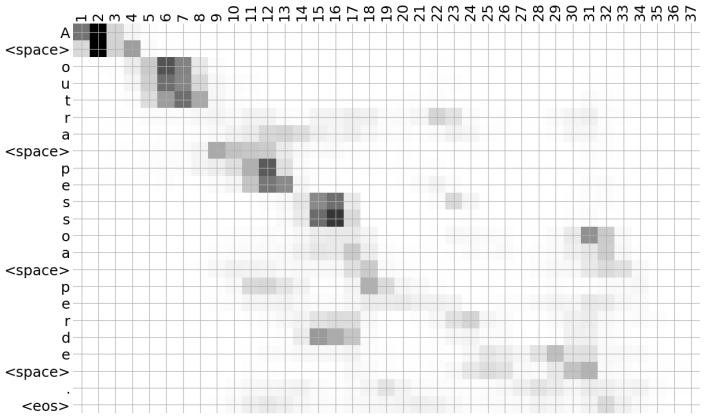
$$H_t = \sum_{j=1}^{|x|} \alpha_{tj} \log \alpha_{tj} \tag{7.1}$$

with α_{tj} being the alignment score between target token y_t and source speech frame x_j. This measure is then averaged for all tokens at the corpus level (How2 10%). A low entropy means a high level of confidence of the attention mechanism in its source-target alignments and vice-versa (see example in Figure 7.3).

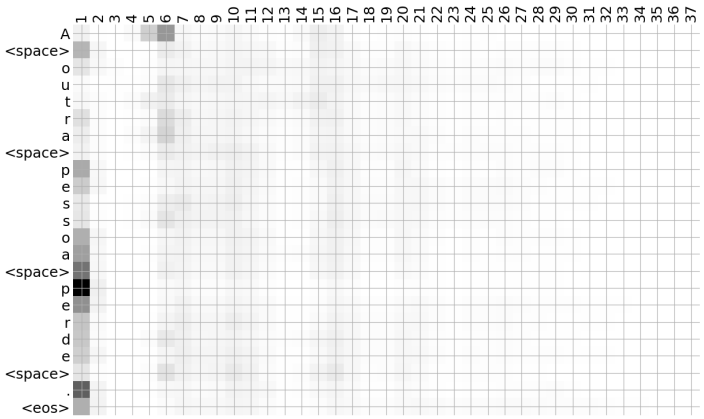
No.	Feature	How2 dev	How2 val
1	wav2vec	0.66	0.66
2	wav2vec + FT	0.65	0.65
3	wav2vec + norm	0.57	0.57
4	wav2vec + FT + norm	0.51	0.51
5	fbanks	0.89	0.90
6	fbanks + norm	0.93	0.93

Table 7.7: Averaged entropies of soft-alignments on How2 dev and val set. AST models trained on 10% partition of How2.

We can see clearly in Table 7.7 that, in our low-resource setting, wav2vec features lead to better source-target alignments (lower entropy) in comparison with fbanks features. Fine-tuning and normalization of self-supervised representations also help improve the soft alignments.



(a) wav2vec - *entropy* = 0.67



(b) fbanks - *entropy* = 0.81

Figure 7.3: Soft alignments between source speech features and target text for sentence *"A outra pessoa perde."*

7.2.4.3 Better robustness to speaker variability

In order to investigate robustness to speaker variability, several *Automatic Speaker Verification (ASV)* systems are trained on either wav2vec or fbanks features extracted from LibriSpeech train-clean-360 dataset using Kaldi. These ASV systems are based on x-vectors and *Probabilistic Linear Discriminant Analysis (PLDA)* (Snyder et al. (2018)). We use a *Time Delay Neural Network (TDNN)* model topology similar to the one described in Snyder et al. (2018) for extracting x-vectors. Input features are fbanks or wav2vec (optionally normalized) while output corresponds to 921 speakers of the training corpus. ASV experiments are conducted on the VoxCeleb1 test (Nagrani et al.

(2017)) and LibriSpeech test-clean sets [9]. ASV results (Equal Error Rate - EER) are presented in Table 7.8.

No.	Feature	VoxCeleb	Libri (f)	Libri (m)
1	wav2vec	22.75	11.22	2.23
2	wav2vec + norm	20.93	10.54	1.79
3	fbanks	15.78	5.47	0.89
4	fbanks + norm	16.25	3.47	0.67

Table 7.8: Equal error rate (EER %) on the VoxCeleb1 test and LibriSpeech test sets for female (f) and male (m) speakers.

The table shows that, in all experiments, significantly higher EER scores are obtained when using wav2vec features than when using fbanks features. This reaffirms our hypothesis that wav2vec representations remove speaker information from speech signal, and therefore, are more robust to speaker variability [10]. We would like to thank our colleague Natalia Tomashenko [11] specifically for designing this experiment.

7.3 French SSL models pre-training

The previous sections show the effectiveness of SSL speech representations on the end-to-end AST tasks. A lengthy discussion of Chapter 5 also concludes that this kind of representations is of great potential in other speech tasks, for example, ASR. However, despite these promising results, we observe that comprehensive comparisons of SSL models are difficult to make due to a noticeable lack of standardization in the evaluation process for these models. Furthermore, most works in SSL have been concentrating on English with a few exceptions in multilingual SSL (Conneau et al. (2020); Wang et al. (2021)). For these reasons, we propose ***LeBenchmark***, an open-source and reproducible framework for assessing SSL from ***French*** speech data (Evain et al. (2021a,b)). These collective works from a combination of several research groups in France make direct contributions in different aspects including gathering and documenting large-scale and heterogeneous corpora, training seven

[9] The trial and enrollment subsets of the LibriSpeech test-clean for the ASV task are described in more details in Tomashenko et al. (2020).

[10] We would also expect that mean and variance normalization increases EER but this is not the case. One explanation might be that normalization also removes channel variability and thus improves ASV.

[11] `https://scholar.google.com/citations?user=xX-frYoAAAAJ&hl=ru`

SSL wav2vec2.0 models, and providing a clear evaluation protocol for four downstream tasks: (1) ASR, (2) *Spoken Language Understanding (SLU)*, (3) AST and (4) *Automatic Emotion Recognition (AER)*. We, therefore, find it necessary to start this section by providing a clarification that the contributions of this thesis stay mainly in the training of the SSL wav2vec2.0 models, which are essential for the evaluations of the downstream tasks that follow. This chapter will review different aspects of training these models along with their impacts on the performance of the AST tasks.

No.	Corpus	#Utterances	Duration	#Speakers	Speech type
Small dataset – 1K					
1	**MLS French** (Pratap et al. (2020))	**263,055** 124,590 / 138,465 / –	**1,096:43** 520:13 / 576:29 / –	**178** 80 / 98 / –	Read
Medium dataset – 3K					
2	**African Accented French** (old (2003))	**16,402** 373 / 102 / 15,927	**18:56** – / – / 18:56	**232** 48 / 36 / 148	Read
3	**Att-Hack** (Le Moine and Obin (2020))	**36,339** 16,564 / 19,775 / –	**27:02** 12:07 / 14:54 / –	**20** 9 / 11 / –	Acted Emotional
4	**CaFE** (Gournay et al. (2018))	**936** 468 / 468 / –	**1:09** 0:32 / 0:36 / –	**12** 6 / 6 / –	Acted Emotional
5	**CFPP2000** (Branca-Rosoff et al. (2012))	**9853** 166 / 1,184 / 8,503	**16:26** 0:14 / 1:56 / 14:16	**49** 2 / 4 / 43	Spontaneous
6	**ESLO2** (Eshkol-Taravella et al. (2011))	**62,918** 30,440 / 32,147 / 331	**34:12** 17:06 / 16:57 / 0:09	**190** 68 / 120 / 2	Spontaneous
7	**EPAC** (Estève et al. (2010))	**623,250** 465,859 / 157,391 / –	**1,626:02** 1,240:10 / 385:52 / –	**Unk** – / – / –	Radio Broadcasts
8	**GEMEP** (Bänziger et al. (2012))	**1,236** 616 / 620 / –	**0:50** 0:24 / 0:26 / –	**10** 5 / 5 / –	Acted Emotional
9	**MPF** (Françoise (2017); MPF (2019))	**19,527** 5,326 / 4,649 / 9,552	**19:06** 5:26 / 4:36 / 9:03	**114** 36 / 29 / 49	Spontaneous
10	**PORTMEDIA (French)** (Lefèvre et al. (2012))	**19,627** 9,294 / 10,333 / –	**38:59** 19:08 / 19:50 / –	**193** 84 / 109 / –	Acted telephone dialogue
11	**TCOF (Adults)** (ATILF (2020))	**58,722** 10,377 / 14,763 / 33,582	**53:59** 9:33 / 12:39 / 31:46	**749** 119 / 162 / 468	Spontaneous
	Medium dataset total	**1,111,865** 664,073 / 379,897 / 67,895	**2,933:24** 1,824:53 / 1,034:15 / 74:10	–	–
Large dataset – 7K					
12	**MaSS** (Boito et al. (2020))	**8,219** 8,219 / – / –	**19:40** 19:40 / – / –	**Unk** – / – / –	Read
13	**NCCFr** (Torreira et al. (2010))	**29,421** 14,570 / 13,922 / 929	**26:35** 12:44 / 12:59 / 00:50	**46** 24 / 21 / 1	Spontaneous
14	**Voxpopuli Unlabeled** (Wang et al. (2021))	**568,338** – / – / –	**4,532:17** – / – / 4,532:17	**Unk** – / – / –	Professional speech
15	**Voxpopuli Transcribed** (Wang et al. (2021))	**76.281** – / – / –	**211:57** – / – / 211:57	**327** – / – / –	Professional speech
	Large dataset total	**1,814,242** 682,322 / 388,217 / 99,084	**7,739:22** 1,853:02 / 1,041:07 / 4,845:07	–	–

Table 7.9: Statistics for the speech corpora used to train SSL models according to gender information (male / female / unknown). The small dataset is from MLS only. Every dataset is composed of the previous one + additional data; duration: hour(s):minute(s).

7.3.1 Data

In this work, we gathered a large variety of speech corpora in French that cover different accents (MLS, African Accented Speech, CaFE), acted emotions (GEMEP, CaFE, Att-Hack), telephone dialogues (PORTMEDIA), read speech (MLS, African Accented French, MaSS) and spontaneous sentences (CFPP2000, ESLO2, MPF, TCOF, NCCFr), broadcast speech (EPAC) and professional speech (Voxpopuli) (Table 7.9). We would like to thank our colleague Solène Evain [12], who is the main contributor to data gathering.

Pre-processing for SSL training: recordings of each corpus are first converted into mono PCM $16\,bits$, $16\,kHz$ before being segmented into smaller segments using timestamp information from transcriptions. Following Baevski et al. (2020b), we remove utterances longer than $30\,s$. Finally, we group different corpora into different subsets of training data as the following:

- **Small dataset ($\approx$ 1k hours)**: comprises only the MLS corpus for comparison with wav2vec2.0 Baevski et al. (2020b) which uses only read English speech.
- **Medium dataset ($\approx$ 3k hours)**: consists of MLS corpus and the corpora presented from line 2 to line 11 of Table 7.9. This combination has in total $2,933\,h$ of different speech types detailed in the table.
- **Large dataset ($\approx$ 7.7k hours)**: contains the medium dataset and 4 additional corpora including MaSS, NCCFr and Voxpopuli (unlabeled + transcribed). This has in total $7,739\,h$ of speech with a wide range of speech types presented in the table.

7.3.2 Training and Sharing SSL Models

From the gathered French data described in Section 7.3.1, this thesis contributes directly to the training and sharing seven wav2vec2.0 pre-trained models. The same as Baevski et al. (2020b), we utilize two different wav2vec2.0 architectures namely *large* and *base*, each of which is trained on our *small* ($1K$), *medium* ($3K$) and *large* ($7K$) corpus. This results in a set of wav2vec2.0 models: W2V2-Fr-1K-*base*, W2V2-Fr-1K-*large*, W2V2-Fr-3K-*base*, W2V2-Fr-3K-*large*, W2V2-Fr-7K-*base*, W2V2-Fr-7K-*large*. In addition, a specific model (W2V2-Fr-2.7K-*base*) is trained on a subset of our *medium* set only containing MLS and EPAC ($2.7K$ hours of audio) in order to enable further investigation

[12] `https://solene-evain.github.io/`

on the impacts of spontaneous speech on SSL representations. Hyperparameters and architectures for *base*[13] and *large*[14] are identical to those proposed by Baevski et al. (2020b). Detailed summary of the hyperparameters used to train our SSL models can be found in Table 7.10.

Model	Training data	Transf blocks	Model dim	Inner dim	Heads	Updates
W2V2-Fr-1K-*base*	1,096 h	12	768	3,072	8	200K
W2V2-Fr-1K-*large*	1,096 h	24	1024	4,096	16	200K
W2V2-Fr-2.7K-*base*	2,773 h	12	768	3,072	8	500K
W2V2-Fr-3K-*base*	2,933 h	12	768	3,072	8	500K
W2V2-Fr-3K-*large*	2,933 h	24	1024	4,096	16	500K
W2V2-Fr-7K-*base*	7,739 h	12	768	3,072	8	500K
W2V2-Fr-7K-*large*	7,739 h	24	1,024	4,096	16	500K

Table 7.10: Hyperparameters of our pre-trained SSL models. *Transf blocks*, *Model dim*, *Inner dim* stand for Transformer blocks, model dimension, and inner dimension, respectively. Note that the maximum number of updates set for training each model is shown in the last column of the table, with one update corresponding to a call to the *.backward()* function in PyTorch. In practice, training is stopped at a round number of updates once the loss observed on the development set of the MLS corpus reaches a stable point (learning curves are shown in Figure 7.4).

We share our pre-trained wav2vec2.0 models with the community via HuggingFace [15] for further integration with the well-known toolkits such as SpeechBrain (Ravanelli et al. (2021)), Fairseq (Ott et al. (2019)), ESPnet (Inaguma et al. (2020)) or Kaldi.

[13] `https://github.com/pytorch/fairseq/blob/main/examples/wav2vec/config/pretraining/wav2vec2_base_librispeech.yaml`

[14] `https://github.com/pytorch/fairseq/blob/main/examples/wav2vec/config/pretraining/wav2vec2_large_librivox.yaml`

[15] `https://huggingface.co/LeBenchmark`

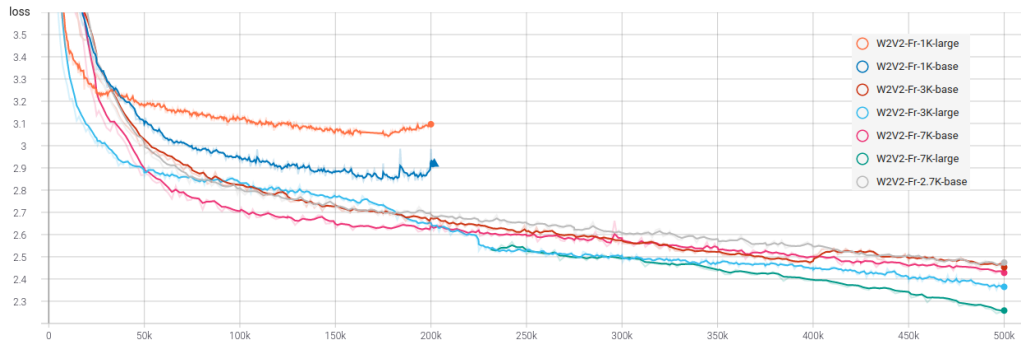


Figure 7.4: Evolution of the loss on the development set during the pre-training of the SSL models.

7.3.3 Experiments on multilingual AST

In this subsection, we briefly present the multilingual AST (translating directly from French speech to text in another language) results obtained from using the SSL speech features extracted by the aforementioned wav2vec2.0 models. Besides, we also compare the performance of our pre-trained wav2vec2.0 models with XLSR-53-*large*, a publicly available multilingual wav2vec2.0 model shared by Fairseq [16]. We state that this thesis is not responsible for producing these results. More detailed results, which are produced by our colleague Hang Le [17], can be found in Evain et al. (2021b). We selectively present some of the results here in order to illustrate the impacts of our wav2vec2.0 models to the AST tasks.

Dataset: since we only concern with the translation from French speech, the training data for the AST models are chosen from the subsets of the multilingual TEDx dataset (Salesky et al. (2021)) which have French as the source language. This work covers translation directions from French to three target languages: English (En), Spanish (Es), and Portuguese (Pt), with following training sizes 50 h (En), 38 h (Es), and 25 h (Pt).

Experiments: in Evain et al. (2021b), separate models are trained for each language pair of the multilingual dataset. These models take as input either 80-dimensional Mel filter-bank (MFB) features (baseline models) or learned representations derived from SSL models. As regards the SSL representations, beside the direct use of the pre-trained SSL models as feature extractors (referred to as *task-agnostic pre-training*), we further fine-tune (presented on the last 4 lines of Table 7.11) our pre-trained wav2vec2.0 models on: (1) unlabelled in-domain task data (referred to as *self-supervised task-specific*) and

[16] `https://github.com/pytorch/fairseq/tree/master/examples/wav2vec`
[17] `https://hangle.fr/`

(2) transcribed data (referred to as *fine-tune on ASR task-specific*). We note that the SSL models resulted from these fine-tuning processes are also used as our feature extractors for the multilingual AST task.

Results: some of the results are shown in Table 7.11 (a bigger table of results can be found in Evain et al. (2021b)). Firstly, it can be seen from the table that, in general, SSL features of different settings outperform the baselines using MFB features by a large margin. Secondly, we observe that the ASR fine-tuning approach (c) yields the best results in comparison with (a) task-agnostic and (b) self-supervised task-specific fine-tuning. Thirdly, focusing on the task-agnostic block (a), we see that French SSL models clearly outperform those pre-trained on multilingual data. Furthermore, when comparing across different French SSL model sizes (base versus large), we see that the large architecture yields considerable improvement over its base counterpart. Finally, we see that for task-specific models, Fr-7K-*large* mostly yields the best performance in each group.

Features	Valid			Test		
	En	**Es**	**Pt**	**En**	**Es**	**Pt**
MFB	1.15±0.17	0.67±0.15	0.61±0.13	1.10±0.14	0.87±0.12	0.32±0.03
(a) Task agnostic pre-training						
Fr-7K-*base*	15.13±0.45	12.78±0.40	2.65±0.20	14.50±0.45	13.61±0.44	2.66±0.23
Fr-7K-*large*	19.23±0.54	17.59±0.49	9.68±0.37	19.04±0.53	18.24±0.49	10.98±0.41
XLSR-53-*large*	7.81±0.33	0.49±0.13	0.43±0.07	6.75±0.29	0.52±0.08	0.36±0.05
(b) Task specific pre-training (self-supervised on mTEDx)						
Fr-7K-*large*	19.65±0.55	17.53±0.47	9.35±0.36	19.36±0.54	18.95±0.53	10.94±0.38
XLSR-53-*large*	6.83±0.33	0.54±0.14	0.34±0.03	6.75±0.32	0.34±0.03	0.29±0.03
(c) Task specific pre-training (fine-tuned for ASR on mTEDx)						
Fr-7K-*large*	**21.41**±0.51	20.32±0.49	**15.14**±0.48	**21.69**±0.58	**21.57**±0.52	**17.43**±0.52
XLSR-53-*large*	21.09±0.54	**20.38**±0.56	14.56±0.45	20.68±0.53	21.14±0.55	17.21±0.54

Table 7.11: BLEU on valid and test sets of multilingual TEDx (mTEDx). The highest value in each group (task-agnostic pre-training, task-specific self-supervised, and supervised fine-tuning) is underlined while the best value in each column is highlighted in **bold**. Gray numbers denote the standard deviation computed using bootstrap re-sampling (Koehn (2004)).

7.4 Conclusion

In this chapter, we have presented our investigation on the impacts of self-supervised learning from speech on end-to-end AST performance. This, to

the best of our knowledge, is one of the first efforts to use SSL for the AST task. Specifically, we use a pre-trained English wav2vec model, a CPC model pre-trained from unlabeled speech, as a feature extractor for a downstream AST task that concerns English as the source language. Our experimental results show that self-supervised pre-training is particularly efficient in low-resource and medium-resource settings, when the amount of speech translation training data is smaller than 100 hours, and that fine-tuning CPC models on the AST training data further improves the performance. Furthermore, in higher resource settings, we observe through experiments that ensembling AST models trained with filter-bank and CPC representations leads to near state-of-the-art models without using any ASR pre-training. Our analyses show that self-supervised representations show better phone discrimination, source-target alignments and speaker robustness, which might be responsible for this significant improvement in comparison with the baseline filter-bank features. This might be particularly beneficial in the situation where we need to develop a system which translates from speech in a language with poorly standardized orthography or even from speech in an unwritten language.

In addition, this chapter also presented our contributions in the training of several SSL models from French speech. Particularly, we train 7 wav2vec2.0 models of different model sizes on different combinations of training data. These models are used in our open-source and reproducible framework for assessing SSL from French speech data namely ***LeBenchmark***. We show results on a multilingual AST setting, which reaffirm that SSL features marginally outperform the baselines filter-bank features, and that fine-tuning SSL models on the task-specific data in a supervised manner can lead to great improvements.

Finally, we must state that even though this kind of speech representations is proven to be strongly effective for the AST task, in the remainder of this dissertation, we shall come back to using filter-bank features in the experiments concerning online AST. This is due to the computational constraint that we encountered during the process. Online AST shall be shown to be more expensive in terms of computation, and therefore, we have not been able to apply SSL features to this task.

Chapter 8

Online Neural Speech Translation

8.1 Introduction

In Chapter 2, we discussed offline end-to-end AST models which generate translation text conditioned on the richer context of the whole encoded input sequence. This is to be distinguished from online AST (discussed in Chapter 4) which is much more challenging due to the fact that the encoded context from which the translation text is conditioned is generated from partial input, and therefore, is somewhat not as rich as in offline AST. Most efforts so far in online AST focus either on developing online decoding strategies which allow pre-trained offline AST models to function in the online fashion; or training online models which are backboned by the architectures similar to offline models with or without additional components. This chapter is dedicated to presenting our contributions in online AST which include both the mentioned aspects:

- Developing an online decoding strategy that allows leveraging pre-trained offline models to work in online mode.
- Designing an online encoding strategy that specifically allows ULSTM speech encoders to work more effectively in online mode.
- Fine-tuning pre-trained offline models in an online training fashion, which boosts the overall performance of the online systems.

This Chapter is organized to emphasize all these contributions, firstly talking about our proposed decoding policy which is a variant of the wait-k policy presented earlier in this thesis. This is followed by our online encoding approach that boosts the performance of the online ULSTM speech encoders. Finally, we will show how fine-tuning pre-trained offline models in online mode helps improve the performance of the online translation system.

Before embarking on the detailed discussion, it is worth mentioning that our works are significantly constrained by our computational capacity. This becomes more considerable when training the models in online mode is involved. We shall see later that in order to fine-tune such models, a part of the training data which contains long sentences needs to be filtered out, and for

this reason, training the models from scratch is out of the question. Despite this great challenge, the work presented in this chapter results in the following publications:

- **H. Nguyen**, Y. Estève, and L. Besacier, ***"An empirical study of end-to-end simultaneous speech translation decoding strategies"***, in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021.
- **H. Nguyen**, Y. Estève, and L. Besacier, ***"Impact of encoding and segmentation strategies on end-to-end simultaneous speech translation"***, in Interspeech 2021, Brno, Czech Republic, 2021.

8.2 Online decoding policy

In this section, we present our proposed decoding strategy which shall be illustrated as a modification to the original wait-k policy proposed for text-to-text translation (Ma et al. (2019)), or more closely to another modification for speech-to-text translation proposed by Han et al. (2020). We shall prove that this simple yet efficient decoding approach allows leveraging any pre-trained end-to-end offline AST model for simultaneous speech translation. Our contributions are the following:

- Adapting the algorithm from Han et al. (2020), which is already an adaptation of Ma et al. (2019)'s wait-k, but introducing the possibility to write several output tokens at a time.
- Conducting empirical experiments showing that this adaptation allows to control AL/BLEU trade-off along different latency regimes with a pre-trained end-to-end AST model that does not need to be re-trained in online mode.
- Evaluating the proposed method for 2 different language pairs En→Pt and En→De and with different output granularities (characters or BPEs).

8.2.1 Pre-trained end-to-end offline model

We reuse the end-to-end offline models presented in Section 6.2 and 6.3 for evaluating our online decoding policies. We recall that these models are based on the encoder-decoder with attention architecture, whose speech encoders stack layers of BLSTM after CNN-based blocks. We shall discuss the special treatment required for this specific kind of speech encoders in online mode.

For the purpose of evaluating our proposed approach on different language pairs and different output granuarities, the following pre-trained models are leveraged:

- **EN→PT pair**: we reuse our best offline character-based (char) model, and two BPE-based (BPE400, and BPE2K) models (model 3*, 4 and 5 of Table 6.6, respectively). The char model scores 26.91 BLEU, while the BPE400 and BPE2K model score 24.73, and 23.11 BLEU respectively on MuST-C tst-COMMON, in beam search mode ($beam\_size = 10$) [1].
- **EN→DE pair**: the offline char EN→DE we choose for evaluating our method is model 3* in Table 6.9, which scores 23.55 and 22.35 BLEU on MuST-C tst-COMMON, and MuST-C tst-HE, in beam search mode ($beam\_size = 10$), respectively.

8.2.2 Simultaneous decoding strategies

We see in Chapter 3 that the deterministic wait-k (Ma et al. (2019)) is a simple but effective online decoding strategy for text-to-text translation. We also see in Chapter 4 that, when adapting this strategy for the online speech-to-text translation task, instead of reading only one input token (speech frame in this case of speech-to-text translation), Han et al. (2020) propose to read more than one speech frame at each step ($s \geq 1$). We are inspired by these works to propose yet another modification to Han et al. (2020)'s policy, which allows us to produce more than one target token at each step. Furthermore, both Ma et al. (2019) and Han et al. (2020) use Transformer-based AST models which are re-trained in an online fashion (the whole models are learned to deal with partial input). As stated in Chapter 6, our pre-trained offline models are LSTM-based instead. It is, therefore, reasonable for us to adapt the wait-k policy to our own context of leveraging LSTM-based models. Moreover, we also hypothesize that simpler LSTM speech encoders might be more robust to limited source context when no re-training is performed in online mode. Consequently, our LSTM-based end-to-end models trained in offline mode are reused without any adaptation nor re-training in this work.

In summary, for online decoding, our proposed deterministic decoding strategy is described with the following parameters (Figure 8.1):

- k **(*wait* parameter)**: presents the number of acoustic frames (at the beginning of the input speech features sequence) read before writing the

[1] Our performance in simultaneous mode will be, in contrast, given with greedy decoding mode.

first output token. This is equivalent to the k parameter introduced in Ma et al. (2019) and Han et al. (2020). In our application, $k = 100$ or 200 frames which corresponds to $1s$ or $2s$.

- s **(*stride* parameter)**: is similar to the s parameter of Han et al. (2020), which is introduced for satisfying the need of denoting the number of acoustic frames in the input speech features sequence to be consumed in order to produce each new target token. For speech-to-text translation task, s is usually greater than 1 whereas the original wait-k (Ma et al. (2019)) always sets $s = 1$. In this work, we set $s = 10$ or 20 in our experiments which is equivalent to $0.1s$ or $0.2s$.

- N **(*write* parameter)**: is our proposed parameter, which denotes the maximum number of output tokens written at each decoding step. This is different from the wait-k policies of Ma et al. (2019) and Han et al. (2020), which use a fixed $N = 1$. In our experiments presented later in this section, we set $N = 1$, 2 or 3 and output tokens can be characters or BPEs.

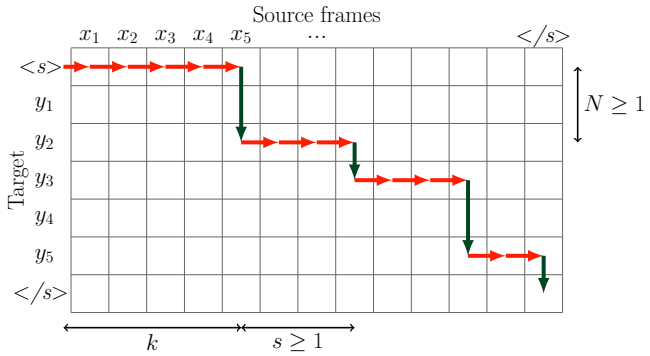


Figure 8.1: Our proposal Nguyen et al. (2021) for modification of the wait-k policy that allows the wait-k policy to read more than 1 source frames ($s \geq 1$) at each decoding step after the first step in order to write at maximum $N \geq 1$ target tokens.

To formalize our method, the same as Chapter 4, we keep the definition of (X, Y) as the source audio sequence paired with the corresponding target text translation, and $g(t)$ as the number of source frames consumed by the encoder at each decoding step t. In addition to these, we introduce $q(t)$ as the number of target tokens generated up to step t. In this work:

$$g(t) = min\{k + (t-1) * s, |X|\} \tag{8.1}$$

$$q(t) = q(t-1) + w_t \tag{8.2}$$

with $q(0) = 0$ and $0 \leq w_t \leq N$ denotes the number of target tokens emitted at step t. At each decoding step t, the model encodes $g(t)$ source frames in order to decode at maximum N target tokens.

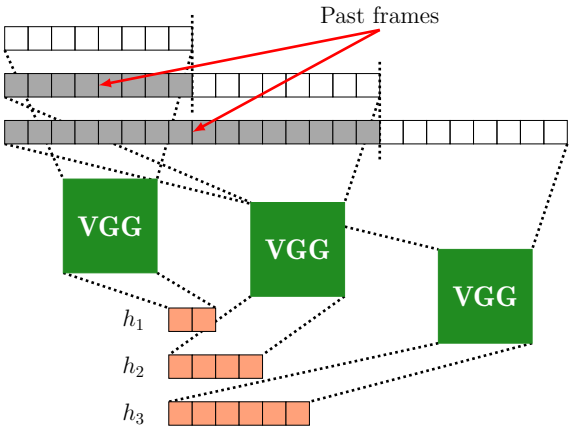


Figure 8.2: Illustration of the re-encode encoding strategy.

Since our pre-trained offline AST models are based on Bidirectional Long Short Term Memory (BLSTM) networks, every time new frames are read, we must re-encode from beginning of the source sequence (Figure 8.2):

$$h^t = encode(X^t) \tag{8.3}$$

with $X^t = x_{\leq g(t)} = (x_1, x_2, ..., x_{g(t)})$ being input buffer at step t. We shall refer to this encoding approach as "*re-encode encoding strategy*", which differs from other encoding approaches presented later in this chapter.

On the decoder side, it takes the encoder's hidden states sequence h^t, and the cached previous hidden state $z_{q(t-1)}$ to compute w_t hidden states, and predict w_t corresponding target tokens as the following:

$$z_j = decode(h^t, z_{j-1}, y_{j-1}) \tag{8.4}$$

$$y_j = predict(z_j) \tag{8.5}$$

with the target token index $j \in [q(t-1)+1, q(t)]$. We update the output buffer Y^t by simply appending $(y_{q(t-1)+1}, ..., y_{q(t)})$ to Y^{t-1}:

$$Y^t = Y^{t-1} + (y_{q(t-1)+1}, ..., y_{q(t)}) \tag{8.6}$$

This goes on until the end of sequence token </s> is predicted, or the length of the output buffer $|Y|$ exceeds a threshold [2]. In case the decoder generates </s> before the whole source sequence X is read, we only append the tokens preceding </s>, then stop generating target tokens for this step and move to the next step, beginning with reading more source frames.

8.2.3 Experiments and results

In this section, we discuss our evaluations of our presented wait-k policy with our pre-trained offline models presented in Section 8.2.1. We focus on investigating:

- Impact of decoding parameters (k, s, N).
- Impact of the target granularity (characters versus BPEs).
- Comparing our method with the state-of-the-art performance on the same task.

No.	k	s	N	tst-HE	tst-COMMON
1	100	10	3	3.01 / 743	4.42 / 800
2	100	10	2	4.26 / 1049	6.89 / 1135
3	100	20	3	5.49 / 1353	8.61 / 1441
4	200	10	3	7.07 / 1521	10.37 / 1552
5	200	10	2	8.77 / 1836	12.79 / 1931
6	100	20	2	8.77 / 2062	12.68 / 2097
7	100	10	1	9.46 / 2146	12.85 / 2157
8	200	20	3	10.38 / 2223	14.6 / 2286
9	200	20	2	13.8 / 2934	16.59 / 2840
10	200	10	1	14.11 / 2973	16.83 / 2880
11	100	20	1	14.64 / 3487	15.79 / 3086
12	200	20	1	17.15 / 4066	17.94 / 3610
13	offline			20.54 / 7005	21.38 / 5782

Table 8.1: (BLEU / AL) scores of the En→De char model evaluated on MuST-C tst-HE and MuST-C tst-COMMON. Sorted by AL of tst-HE in increasing order. AL is in *milliseconds*.

[2] In this work, we set a $max\_length\_ratio = \frac{max\_output\_sequence\_length}{encoder\_hidden\_state\_sequence\_length} = 1.0$ for En→DE experiments, and 1.6 for En→Pt experiments.

No.	k	s	N	tst-COMMON
1	100	10	3	4.67 / 611
2	100	10	2	8.38 / 907
3	100	20	3	11.43 / 1237
4	200	10	3	11.61 / 1402
5	200	10	2	15.97 / 1748
6	100	20	2	16.67 / 1948
7	100	10	1	16.95 / 1976
8	200	20	3	17.94 / 2106
9	200	20	2	20.69 / 2697
10	200	10	1	20.98 / 2735
11	100	20	1	20.64 / 2910
12	200	20	1	22.67 / 3505
13	offline			25.07 / 5986

Table 8.2: (BLEU / AL) scores of the En→Pt char model evaluated on MuST-C tst-COMMON. Sorted by AL in increasing order. AL is in *milliseconds*.

8.2.3.1 Impact of decoding parameters

In this series of experiments, our pre-trained offline char models (Section 8.2.1) are leveraged to decode in online decoding mode using our proposed wait-k policy with different combinations of (k, s, N). The experimental results (BLEU for different AL) are given in Table 8.1 for En→De and in Table 8.2 for En→Pt. We note that since our online policy only allows greedy decoding, for a fair comparison with the online mode, the offline models are re-decoded in greedy decoding mode. This gives the results on the last rows of both the tables. We observe that:

- The 3 parameters (k, s, N) of the proposed policy allow us to generate results over the whole range of AL (from very low latency regimes $< 1s$ to higher AL values between $2s$ and $3s$).
- Our strategy obtains decent BLEU scores for a latency of $2s$ (BLEU=14.60 for En→De and BLEU=17.94 for En→Pt on MuST-C tst-COMMON, respectively).
- When looking at lines 6-7 and 9-10 of Table 8.1 and 8.2, we observe that writing two characters at each decoding step ($N = 2$) with bigger stride

($s = 20$) seems to be slightly better in terms of AL, yet slightly worse in terms of BLEU scores than writing one character at a time ($N = 1$) with smaller stride ($s = 10$).

8.2.3.2 Impact of target granularity

In order to investigate the impact of different target token types, we decode our pre-trained offline BPE2K and BPE400 models (Section 8.2.1), using our proposed online decoding strategy. We alternate between different (k, s, N) triplets, with $k = [100, 200]$, $s = [10, 20]$, and $N = [1, 2]$. We then sort the results by the increasing order of AL of each model, and for better visualization, we pick the (k, s, N) combinations that give the best BLEU/AL trade-offs and strip off the points that have close AL but worse BLEU. The results are shown in Figure 8.3, whose data points correspond to $(k, s, N) = (100, 10, 2), (200, 10, 2), (200, 20, 2), (200, 20, 1)$. We note that the results are given as a trend and we need to clarify that our offline char-based and BPE-based models have different performance as well (Table 6.6).

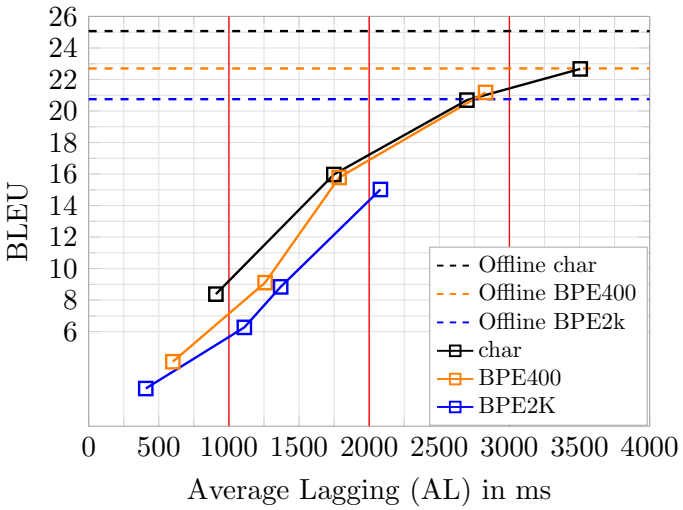


Figure 8.3: Character-based vs BPE-based models on En→Pt translation, evaluated on MuST-C tst-COMMON.

We can see from the figure that, in general, for a same (k, s, N) triplet, char model tends to give higher BLEU score, but bigger AL than the two BPE models. We can observe the same trend when comparing BPE400 with BPE2K model. Our explanation is that because BPEs are bigger token units than characters, when being forced to generate approximately the same number of target tokens (the same N) from the same amount of context in terms of

number of source frames (same (k, s)), BPE models should give worse translation quality. On the other hand, BPE models take less number of source frames than char model to make up a word, therefore they achieve smaller AL with similar (k, s, N) settings.

8.2.3.3 Comparison to the state-of-the-art

In this subsection, we compare the performance of our method with the winning system for speech-to-text online translation at the IWSLT 2020 (Elbayad et al. (2020b)). This system is a cascade of an ASR system paired with an online MT system, translating English speech into German text. The ASR system, which is used to stream the input sequence, is a strong hybrid HMM/DNN system built using the Kaldi speech recognition toolkit (scoring WER=14.2% on MuST-C tst-COMMON in offline mode). The online MT system, which is used to generate translation from partial hypotheses handed by the ASR component, is a Transformer-based wait-k decoder with a unidirectional encoder. Instead of optimizing a single decoding path corresponding to a specific k value, Elbayad et al. (2020b) jointly optimize the online MT model across multiple wait-k paths (i.e, k values are generated randomly during the training process).

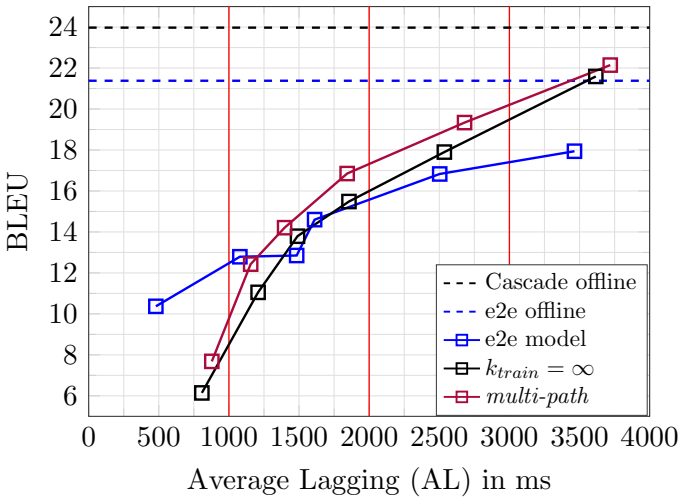


Figure 8.4: Comparison of our proposed character-based En→De end-to-end model (*e2e model*) with that of the (winning) ON-TRAC cascaded models with (*multi-path*) or without ($k_{train} = \infty$) retraining for simultaneous mode. Note that because Elbayad et al. (2020b) use the original AL metric to compute their latency, in this curve, we also use original AL (Ma et al. (2019)) to compute the latency of our end-to-end model.

In Figure 8.4, for better visualization, (k, s, N) combinations for the En→De char model which give close AL regimes to that of the above cascaded models are chosen. We can observe from the figure that our proposed decoding strategy (the blue curve) performs reasonably well in comparison with the cascaded models with re-training in online mode (*multi-path*) or without ($k_{train} = \infty$) re-training. We emphasize that our method (dubbed as *e2e model*) is better in low latency regimes while the cascaded model is better in higher latency regimes. We further stress that our method does not require re-training in online mode, it is thus fair to compare it with the corresponding cascaded model: $k_{train} = \infty$.

8.3 Online encoding strategy

We have just presented above our wait-k decoding policy which allows us to utilize any pre-trained offline AST model for the online AST task. However, since our pre-trained models' speech encoders are based on *Bi-directional* Long Short-Term Memory (BLSTM), re-encoding of the full input is needed each time a new speech block is read (Figure 8.2). We shall show in this section that this encoding approach is highly inefficient for the online translation task and that although replacing BLSTM by ULSTM encoding degrades the performance in offline mode, it actually improves both efficiency and performance in online mode. This observation is similar to that of Elbayad et al. (2020a) for online text translation. Furthermore, this section also presents our investigation of the segmentation methods for the speech flow for alternating optimally between READ (encoding input) and WRITE (decoding output) operations.

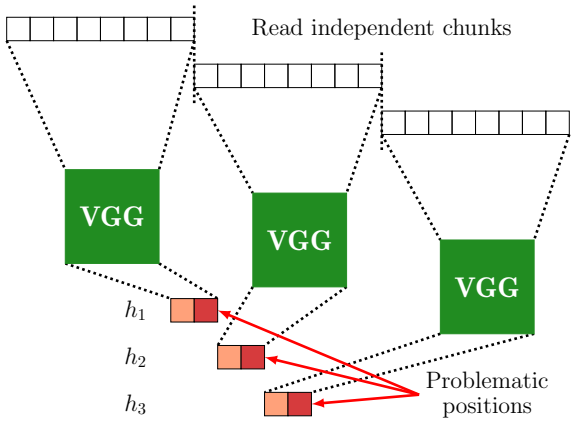
In summary, the contributions of this thesis presented in this section are:

- Proposing to replace BLSTM speech encoder by ULSTM speech encoder and conducting experiments that show when using the same re-encode encoding strategy (Figure 8.2), ULSTM speech encoder yields better inference speed and performance in comparison with BLSTM speech encoder.

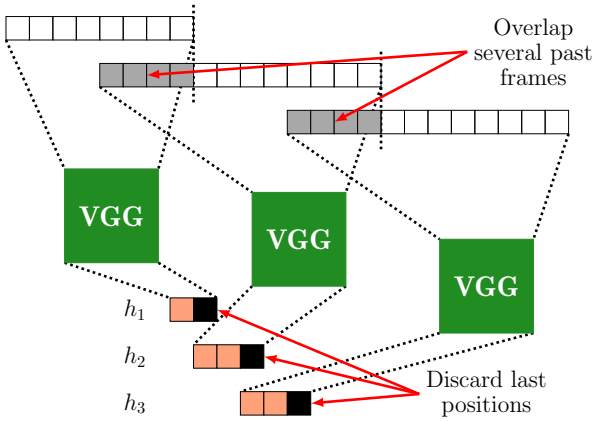
- Proposing a new encoding strategy named ULSTM *Overlap-and-Compensate* designed specifically for our ULSTM speech encoder and showing that the proposed method further improve both inference speed and performance of the translation system.

- Analyzing the impact of speech flow segmentation on the BLEU/AL trade-off, concentrating on the comparison of three segmentation meth-

ods including fixed interval boundaries, oracle word boundaries and randomly set boundaries.



(a) ULSTM reads independent chunks of frames



(b) ULSTM overlap-and-compensate encoding strategy

Figure 8.5: Illustration of different encoding strategies.

8.3.1 Overlap-and-compensate encoding strategy

As stated earlier, our speech encoder presented in Section 8.2.2 must re-encode from the beginning (from left-to-right and from right-to-left) the input speech sequence every time new input frames are consumed. This is proven to be suboptimal, at least for online text translation, by Elbayad et al. (2020a). They show that, for online text translation, using a ULSTM encoder not only has better decoding speed but also yields better BLEU/AL trade-off. This encouraging observation serves as a major force that drives our efforts for replacing BLSTM by ULSTM speech encoders. In order to do this, we

first retrain an offline model similar to the one presented in Section 8.2.2, except that the speech encoder is modified to stacked ULSTM layers instead of BLSTM layers after the VGG-like blocks. After this, using the same wait-k decoding policy, we make comparisons between BLSTM re-encode with two encoding methods for ULSTM encoder as follows:

ULSTM Re-encode strategy: in this encoding strategy (Figure 8.2), the speech encoder still has to re-process the full speech sequence left-to-right every time new speech frames are read. However, since the ULSTM speech encoder is used, this ULSTM Re-encode approach liberates us from computing the BLSTM's right-to-left re-encoding pass. Therefore, we expect that it would improve the decoding speed in comparison with BLSTM Re-encode.

ULSTM Overlap-and-Compensate strategy: is our proposed method specifically for leveraging AST models whose speech encoders are similar to ours. Firstly, we argue that even though ULSTM Re-encode improves decoding speed in comparison with BLSTM Re-encode, re-encoding the full sequence left-to-right each time new speech frames are read is still suboptimal. This motivates more local ways of encoding newly arrived speech frames, one of which is to feed chunk by chunk of input frames independently (Figure 8.5(a)) and the encoded representations at each step can rely on the current input frames plus the representation of the previous step. This turns out to be a poor solution as it gives very disappointing results in terms of translation quality (measured by BLEU). We suspect that this is probably because of the quality deterioration of the VGG blocks' output representations due to padding issues near the chunk boundaries (especially in the last several positions of the representations). As an alternative, when dealing with ULSTM speech encoders, we propose an *Overlap-and-Compensate* encoding strategy. In detail, this approach allows the encoder to read extra frames from the past in order to compensate for some discarded positions at the end of the previous output representation of the VGG-like blocks which, as we have just argued, might be the culprits that cause the aforementioned problem (Figure 8.5(b)).

Algorithm 1 describes the overlap-and-compensate approach applied to the fixed interval segmentation (i.e, our wait-k decoding policy presented in Section 8.2.2). In this algorithm, another parameter *overlap* is introduced. It decides how many past frames the encoder should read in addition to the newly arrived speech frames in order to generate better representations for each step. Note that, our re-encode strategy corresponds to $overlap = 0, offset = 0$). In the series of experiments that are coming, we set *overlap* corresponding to half of the number of input frames of the current step ($overlap = round(s/2)$). The impact of the overlap size shall be mentioned later in this chapter.

Algorithm 1 Overlap-and-Compensate encoding strategy

Input: sequence x;
Output: representation h;
Initialization step $t = 1$, wait parameter k, stride parameter s, total number of frames read so far $g = k$, $offset = 0$,
$finish\_read = False$, $h_0 = None$,
$overlap = round(k/2)$; *# Overlap half of chunk_size*
while $g < |x|$ **do**
    if $t > 1$ **then**
        $overlap = round(s/2)$;
    end
    if $g >= |x|$ **then**
        $g = |x|$; $overlap = 0$; $finish\_read = True$;
    end
    $x_t = x[offset : g]$; *# A chunk read at time t*
    $h_t = Encode(x_t, overlap, h_{t-1}, finish\_read)$;
    $g+ = s$; $t+ = 1$; $offset = g - overlap$;
end
Fuction `Encode`(x, *overlap, prev_h, finish_read*)**:**
    $num\_discard = round(overlap/4)$;
    $h_{vgg} = VGG(x)$;
    if *not* $finish\_read$ **then**
        # Discard num_discard positions in the end
        $new\_length = |h_{vgg}| - num\_discard$;
        $h_{vgg} = h_{vgg}[0 : new\_length]$;
    end
    return $h_{ULSTM} = ULSTM(h_{vgg}, prev\_h)$;

8.3.2 Experiments and results

8.3.2.1 Pre-trained models

As mentioned earlier, we aim to compare BLSTM and ULSTM speech encoder using different online encoding strategies. We rely on the char models of two language pairs En→De and En→Pt presented in Section 6.3 and 6.2 for our experiments concerning BLSTM speech encoders. Regarding the experiments related to ULSTM speech encoders, since this kind of models does not exist before, we pre-train one offline ULSTM model for each language pair with exactly the same configuration as the corresponding BLSTM model, only replacing BLSTM layers by ULSTM layers. We give an overview of the pre-trained models as follows:

En→De language pair: for the experiments related to the BLSTM model,

we reuse the char model presented in Section 6.3, which scores 21.38 and 20.54 BLEU on MuST-C tst-COMMON, and tst-HE, in greedy decoding mode, respectively. To recall, this model is trained on a combination of overall more than $750h$ of translated speech from MuST-C En→De, Europarl-ST En→De, and How2 synthetic (i.e. the German translation has been automatically generated by a text-to-text machine translation system). On this same training set, we pre-train another offline ULSTM model with exactly the same configuration as the BLSTM model, only replacing BLSTM layers by ULSTM layers. The resulting ULSTM model scores 18.21 and 17.98 BLEU on tst-COMMON, and tst-HE, in greedy decoding mode, respectively.

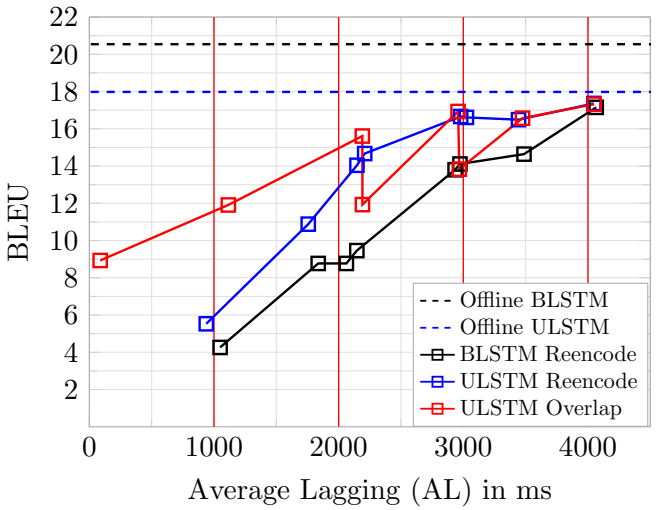
En→Pt language pair: the char model presented in Section 6.2, which scores 25.07 BLEU on MuST-C tst-COMMON in greedy decoding mode, is utilized for our experiments concerning the BLSTM model. This model is also trained on a merged training set from multiple corpora including MuST-C En→Pt and How2 corpus. On this combination of about 674 hours of training data, we train one offline ULSTM model with a similar configuration as the BLSTM model, except that ULSTM layers replace the BLSTM ones in the speech encoder. This model scores 24.13 BLEU on MuST-C tst-COMMON set.

8.3.2.2 Impact of encoding strategies

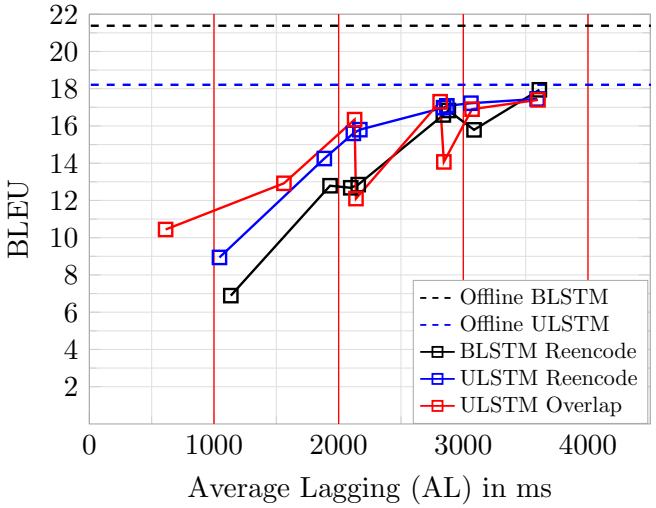
Our purpose of conducting this series of experiments is to compare different models using either BLSTM or ULSTM speech encoders with different encoding strategies (re-encode versus overlap-and-compensate).

En→De language pair: on MuST-C tst-COMMON and tst-HE, we evaluate the presented pre-trained BLSTM and ULSTM model using different encoding strategies. In terms of decoding strategy, we utilize the wait-k presented in Section 8.2.2 with different (k, s, N) triplets ($k = [100, 200]$, $s = [10, 20]$, and $N = [1, 2]$).

The BLEU/AL trade-off obtained from these experiments is illustrated in Figure 8.6. We can observe from the figure that models with ULSTM speech encoders (when using either re-encode or overlap-and-compensate encoding strategy) give consistently better BLEU/AL trade-off than the model with BLSTM speech encoder using re-encode encoding strategy, on both MuST-C tst-HE and tst-COMMON. It is also noticeable from Figure 8.6 that our proposed method, the ULSTM overlap-and-compensate strategy, outperforms the ULSTM re-encode encoding strategy, especially in low-latency regimes.



(a) MuST-C tst-HE



(b) MuST-C tst-COMMON

Figure 8.6: Comparing translation models with BLSTM re-encode / ULSTM re-encode / ULSTM overlap-and-compensate encoding strategies, evaluated on MuST-C tst-HE and tst-COMMON.

In order to investigate the actual time spent decoding each sentence of different encoding strategies, we exclusively use the same CPU machine to decode the whole MuST-C tst-HE set using either BLSTM, ULSTM re-encode or ULSTM overlap-and-compensate encoding strategy. The actual time spent decoding each sentence of each encoding approach is measured and averaged over the whole test set. For better visualization of the difference between the encoding strategies, in each latency regime, the time spent of BLSTM is set as

a speed unit, and the results of ULSTM re-encode and ULSTM overlap-and-compensate are reported relatively to this speed unit. It is shown in Table 8.3 that: (1) ULSTM models are much faster than the BLSTM model no matter which encoding approach is utilized; (2) remarkably, our proposed ULSTM overlap-and-compensate is fastest among all encoding strategies (about 17 times faster than the BLSTM, and 9 times faster than ULSTM re-encode, respectively).

Encoding	**Latency regime**		
	2000ms	**3000ms**	**4000ms**
BLSTM	1	1	1
ULSTM re-encode	0.53	0.53	0.53
ULSTM overlap-and-compensate	0.06	0.06	0.06

Table 8.3: Decoding speed for models with BLSTM/ULSTM re-encode/ULSTM overlap-and-compensate strategies in different latency regimes, measured on MuST-C tst-HE, with the average time spent by BLSTM serves as the time unit, and the average time spent of others are drawn in comparison with this unit.

En→Pt language pair: the same as the En→De language pair presented above, we conduct similar experiments, whose results are shown in Figure 8.10. It confirms that our proposed ULSTM overlap-and-compensate encoding strategy significantly outperforms the BLSTM counterpart.

8.3.2.3 Impact of speech input segmentation

In this section, we establish a goal of answering a scientific question: which methods amongst the following methods for segmenting the speech flow: (1) fixed interval boundaries segmentation, (2) oracle word boundaries segmentation, and (3) randomly set boundaries segmentation is optimal for online speech translation in the context of our presented encoding and decoding strategies? We focus on the En→De language pair in this section.

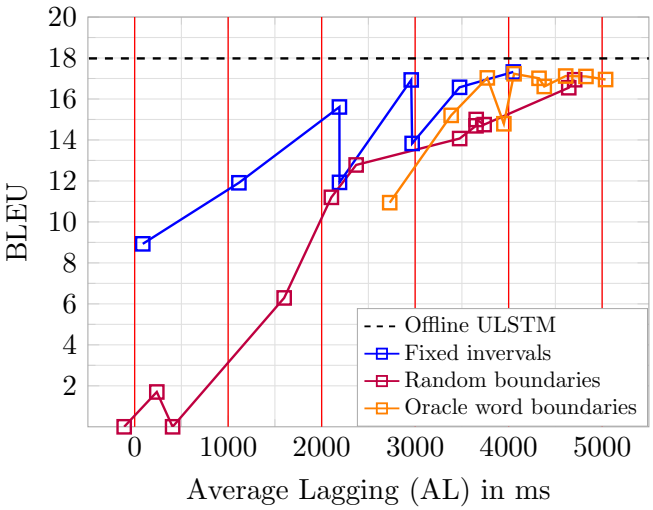


Figure 8.7: BLEU/AL trade-off for different speech input segmentation methods, evaluated on En→De MuST-C tst-HE, using ULSTM *overlap-and-compensate* approach.

Fixed interval boundaries segmentation: is technically the way of feeding input speech frames as in our wait-k policy (Section 8.2.2. To recall, this deterministic decoding strategy reads at the first reading operation k first acoustic frames of the input speech features sequence. At each reading operation after this, the system continues consuming fixed intervals of s frames. A writing operation is put after each reading operation, which writes at maximum N output tokens.

Oracle word boundaries segmentation: attempts to feed relatively precise word-by-word speech chunks instead of fixed-length chunks to the system at each time step. In order to do this, we segment the input audio which is at phrase level into words using Montreal Forced Aligner (McAuliffe et al. (2017)). We rely on a publicly available pre-trained English model [3]. In terms of decoding strategies, we slightly modify our wait-k policy as the following:

- Wait parameter k: remains the number of frames the encoder should wait before starting writing. This serves as an upper bound rather than a precise number of frames read. In detail, at the first decoding step, the encoder reads the first several pre-segmented words until the sum of their corresponding numbers of frames $total\_number\_of\_frames \geq k$. In this work, we experiment with $k = [0, 50, 100, 150, 200]$.

[3]Details about this model can be found here: `https://montreal-forced-aligner.readthedocs.io`. Note that we use this model out of the box, without specifying the beam width.

- Stride parameter s: is the number of source words (chunks of frames) that is read at each decoding step after the first step. In this work, we keep $s = 1$ for all our experiments related to this segmentation approach.

- Write parameter N: remains the maximum number of output tokens (characters) written at each decoding step. We experiment with $N = [1, 2]$ in our experiments related to this segmentation approach.

Randomly set boundaries segmentation: cuts the audio input into random-sized audio chunks and feeds each of these chunks to the translation system at each reading step. Note that we avoid unreasonable fluctuation of the size of each chunk by setting a lower bound (the minimum number of frames) and an upper bound (the maximum number of frames) for each chunk. Within these constraints, the number of frames in each chunk is randomly generated. In this work, we experiment with $[low\_boundary, high\_boundary] = [5, 10], [5, 20], [5, 50], [5, 100], [10, 50], [10, 100]$. The number of frames in the last chunk is adjusted so that the sum of frames in all chunks is equal to the length of the input sequence. Regarding the experiments related to this segmentation method, $N = [1, 2]$.

We note that: (1) algorithm 1 when applied to the oracle word boundaries segmentation and the randomly set boundaries segmentation method would slightly change: $s = |segment[t]| - |segment[t-1]|$, and $k = |segment[0]|$; (2) as for the oracle word boundaries, $segment[0]$ corresponds to all words read at the first decoding step.

Figure 8.7 illustrates that the ULSTM overlap-and-compensate encoding strategy, which is our best presented setting so far, yields the best performance with the fixed interval boundaries segmentation. By contrast, it astonishes us that our system performs worse with the oracle word boundaries segmentation than with the fixed interval boundaries as it almost always yields bigger AL in order to achieve comparable BLEU scores. Finally, as expected, our system performs the worst with the randomly set boundaries segmentation. Its BLEU scores approach 0 (the red dots at the bottom of Figure 8.7) when the segment sizes are too small ($[low\_boundary, high\_boundary] = [5, 10]$).

8.3.2.4 Highlighting the most difficult utterances for simultaneous decoding

Elbayad et al. (2020c) introduce a metric to measure the lagging difficulty of an utterance by using its source-target text alignment as an indicator of how difficult it is to translate an input. In order to measure this, after estimating source-target ((X, Y)) alignments (for instance with *fast-align* (Dyer et al. (2013))), they define a non-decreasing function $z^{align}(t)$, denoting the number

of source words needed to translate a target word. This function guarantees that at a given decoding position t, $z^{align}(t)$ is larger than or equal to all the source positions aligned with t. Lagging difficulty (LD) is then defined as in Equation 8.7 below, with $\tau = argmin_t\{t|z_t = |x|\}$:

$$LD(X,Y) = \frac{1}{\tau}\sum_{t=1}^{\tau} z_t^{align} - \frac{|X|}{|Y|}(t-1) \tag{8.7}$$

In this subsection, we attempt to see if this metric can also be a good indicator for the speech translation task. In order to investigate this, based on LD, we extract the 100 most difficult and the 100 easiest sentences according to the metric, and report the BLEU/AL trade-off (achieved by our best setting: ULSTM overlap-and-compensate encoding strategy coupled with wait-k decoding strategy) for these sets of utterances. The results shown in Figure 8.8 indicate that LD metric could be a good tool for highlighting the most difficult utterances for simultaneous decoding. This is because the AL/BLEU curve for the hardest utterances is clearly below (in the BLEU scale) and much wider spread (in the AL scale) than the one for the easiest utterances. This suggests the possible usage of LD for building specific challenge sets for end-to-end simultaneous speech translation.

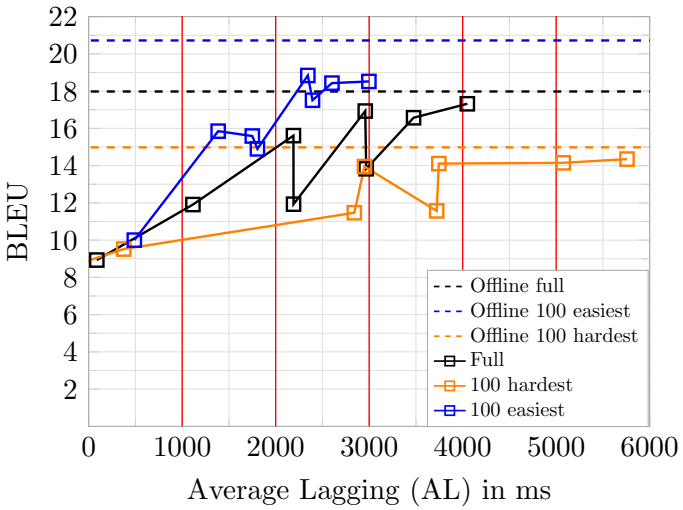


Figure 8.8: BLEU/AL trade-off scored on different subsets of En→De MuST-C tst-HE based on Lagging Difficulty (LD).

8.4 Fine-tuning online model

Earlier in this chapter, we advocated for reusing pre-trained offline models in online mode. We propose in this section to use the pre-trained offline model

to initialize the training of that same model but in a training more adapted to online translation. This can be referred to as fine-tuning even though the model is fine-tuned on the same training data. Although we acknowledge several attempts to train similar systems from scratch in Ren et al. (2020); Ma et al. (2021); Chen et al. (2021), we argue that fine-tuning our pre-trained offline translation models in online mode can further improve the BLEU/AL trade-off, while keeping an affordable computational budget. In summary, the objective of this section is to answer the following questions:

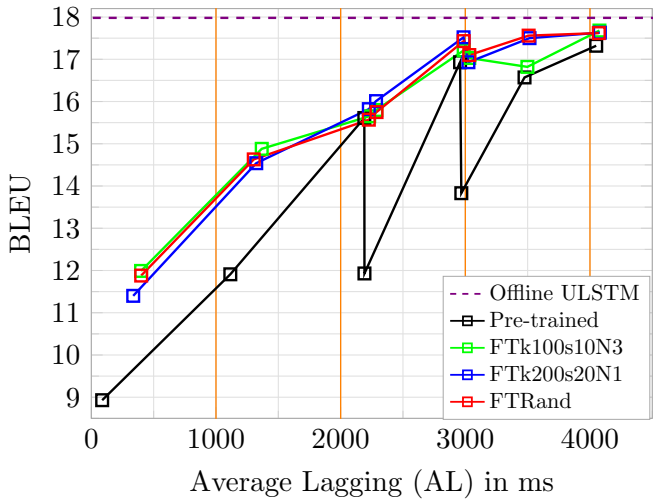
- Is fine-tuning better than using directly the pre-trained offline model?
- If yes, which fine-tuning approach is most beneficial?

8.4.1 Fine-tuned online model

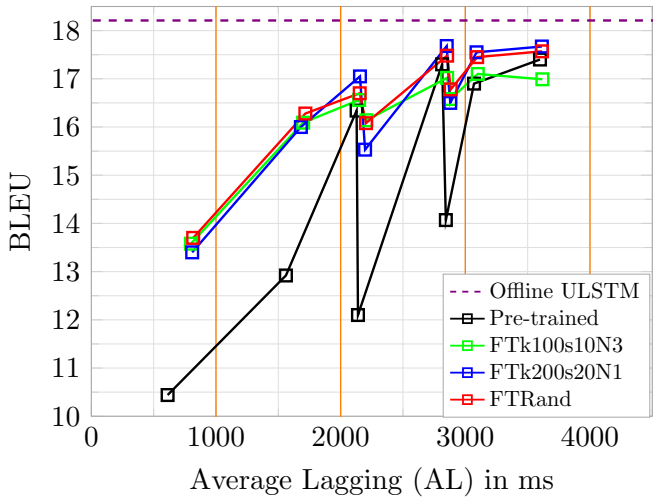
In this experiment, we take the pre-trained offline model and fine-tune it on the same training data in an online mode for several epochs more. Due to our constraint computational capacity, sentences longer than 15 *seconds* are removed from the training set. This corresponds to about 15% of training data removal. We simulate the online (*wait-k*) training fashion by gradually feeding the model a chunk of k frames at the very first step, and at each step after this, a chunk of s frames is fed into the model. As regards the encoding strategy, we implement our proposed overlap-and-compensate strategy. In detail, at each step t, the encoder has access to the current chunk of frames and several past frames ($overlap\_size = s/2$) for calculating the partial speech representation h_t. This partial speech representation is then concatenated with a memory bank of all the previous speech representations that encodes the whole history of the utterance up to step t: $H_t = concat(H_{t-1}, h_t)$. This memory bank H_t is then fed to the decoder along with the decoder's previous hidden states, and the previous target token in order for the decoder to recalculate the attention and try to emit at maximum N target tokens of step t. In this work, we fine-tune our models in the teacher-forcing mode and target tokens are characters (i.e, the previous ground-truth character is fed to the decoder at each step t). We optimize the model based on the accuracy of the hypothesis translation of the whole sequence.

8.4.2 Experiments and results

Regarding the pre-trained models, we fine-tune both the char En→De and En→Pt model presented earlier in this chapter. The following are details of our experiments and the corresponding results.



(a) MuST-C tst-HE



(b) MuST-C tst-COMMON

Figure 8.9: ULSTM Overlap-and-compensate used by pre-trained offline model and by different fine-tuned (**FT**) models, where (k, s, N) are predefined (**FTk100s10N3** and **FTk200s20N1**) or randomly set (**FTRand**).

8.4.2.1 Fine-tuned models versus pre-trained offline model

We first present our experiments for the En→De language pair, which is our main focus in this work. After that, the experiments for the En→Pt language pair are presented as to reaffirm the effectiveness of our proposed method.

En→De language pair: for fine-tuning the pre-trained model of this language pair, we take the best checkpoint of the pre-trained model, and inves-

tigate several fine-tuning scenarios where: (1) the combination of (k, s, N) is predefined or (2) randomly chosen ($k = [100, 200], s = [10, 20], N = [1, 2, 3]$) at each training batch. During inference time, (k, s, N) are fixed for the whole test set, with $k = [100, 200], s = [10, 20], N = [1, 2]$. Each point on Figure 8.9 presents the result of each (k, s, N) combination averaged on the whole test set.

We can observe in Figure 8.9 that, on both MuST-C tst-HE and tst-COMMON, all fine-tuned models outperform the pre-trained offline model, especially in low latency regimes ($< 2s$). Moreover, we observe that fine-tuned model with the predefined $(k, s, N) = (100, 10, 3)$ (**FTk100s10N3**) is slightly worse than the model fine-tuned with $(k, s, N) = (200, 20, 1)$ (**FTk200s20N1**). However, significant differences between **FTk200s20N1** and **FTRand** cannot be observed. This agrees with Elbayad et al. (2020b) on saying that setting wait-k parameters (k, s, N) (or only k in their case) randomly during training is beneficial as it frees us from carefully tuning a large possible (k, s, N) collection, while still generates reasonable BLEU/AL trade-off.

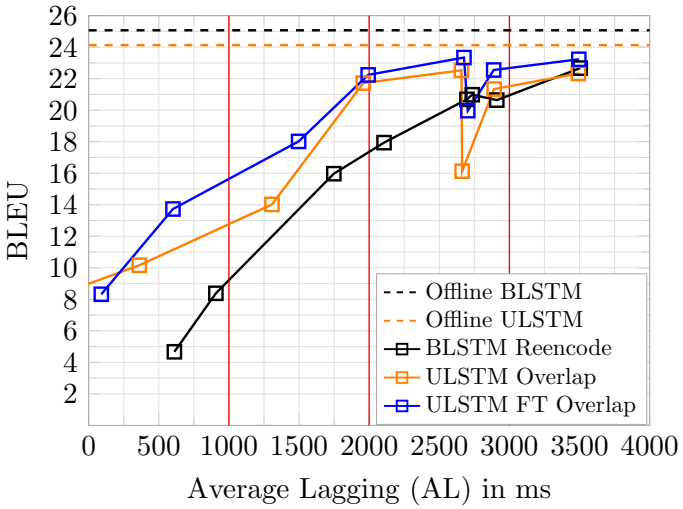


Figure 8.10: Comparing En→Pt char models with BLSTM/ULSTM Overlap encoding strategies/ULSTM Fine-tuned, evaluated on MuST-C tst-COMMON.

En→Pt language pair: we follow the same procedure for fine-tuning our char En→Pt model, except that we only experiment with fine-tuning the model on the randomly set (k, s, N) combinations. The results shown in Figure 8.10 confirm that fine-tuning (the blue curve) is beneficial especially in low-latency regimes.

Online training cost: we highlight that, even with long sentences filtered out from the training data, it is still immensely expensive for training our

char En→De model in online mode. In our constraint computational capacity (1 Nvidia Tesla V100 SXM2 32G GPU), the training takes about $60h$ for 1 epoch. When comparing with training the same model in offline mode which takes about $15h$ for 1 epoch, and in total 19 epochs for acquiring the best checkpoint, we believe that this cost needs to be taken seriously, and that fine-tuning the pre-trained offline model in online mode might be a reasonable solution for reducing our development cost.

8.4.2.2 Fine-tuning scenarios

We have just observed that fine-tuning the pre-trained offline model with (k, s, N) randomly chosen is beneficial. With that in mind, in this subsection, we investigate several fine-tuning (the char En→De model) scenarios where (k, s, N) are randomly chosen during training, but (1) either the encoder or the decoder is frozen, or (2) we fine-tune an intermediate pre-trained model rather than from the best checkpoint.

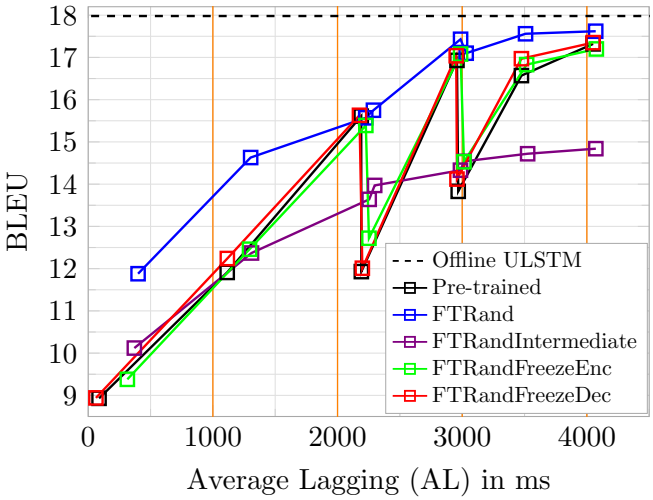
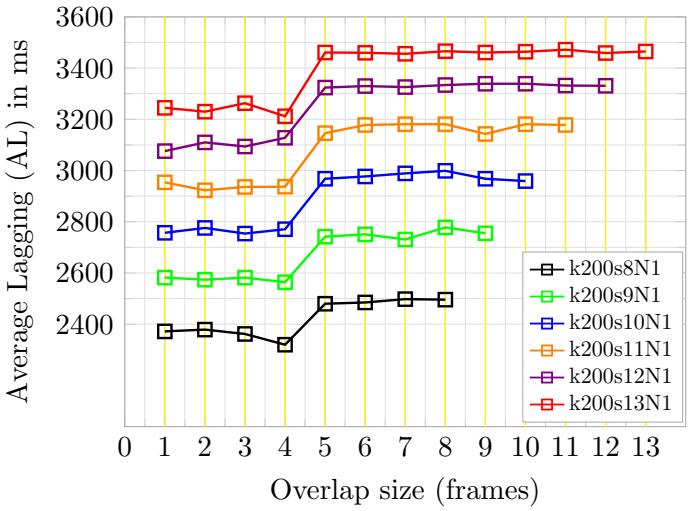


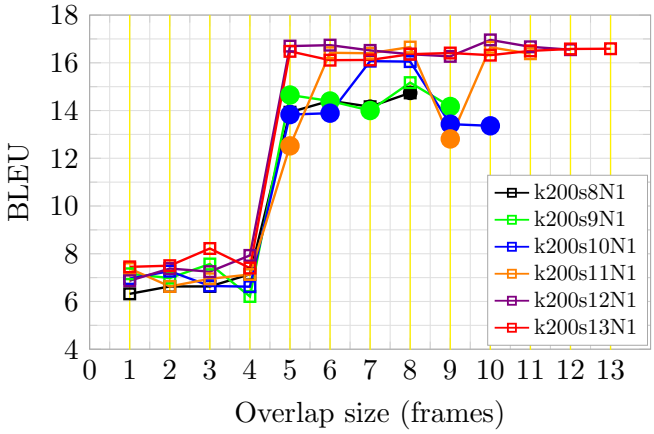
Figure 8.11: Different fine-tuning scenarios, where the best checkpoint of the pre-trained model is chosen for fine-tuning (**FTRand**) and the encoder or decoder part is frozen (**FTRandFreezeEnc** or **FTRandFreezeDec**). **FTRandIntermediate** stands for fine-tuning an intermediate checkpoint.

We can see in Figure 8.11 that it is harmful when freezing either the encoder or the decoder part of the pre-trained model, although it takes much less time to fine-tune a model in such cases (about $44h$ and $35h$ for 1 epoch if the encoder or the decoder is frozen, respectively). Moreover, the maturity of the pre-trained model seems to affect the performance of the fine-tuned one.

Figure 8.11 illustrates that fine-tuning the intermediate model (checkpoint 12-th instead of checkpoint 19-th (the best checkpoint)) significantly decreases the performance of the system.



(a) Impact of *overlap_size* on AL



(b) Impact of *overlap_size* on BLEU

Figure 8.12: Impact of *overlap_size*. Filled circle points represent the settings where *overlap_size* > 4 and the length of the speech encoder's output $h_t = 2\ , \forall t$.

8.5 Impact of *overlap_size*

Earlier in this chapter, we presented our overlap-and-compensate encoding approach that overlaps half the number of new frames at each step (*overlap_size* =

$s/2$). In this section, we shall carefully consider the impact of *overlap_size* on the performance of the online translation system. In order to do this, we evaluate the **FTRand** char En→De model, which yields the best BLEU/AL trade-off of this language pair so far, with a fixed $(k, N) = (200, 1)$ combination, while we let s vary from 8 to 13 and the $overlap\_size = [1, 2, ..., s]$ during inference time.

We can observe notable differences from Figure 8.12(a) in terms of AL between $overlap\_size = 4$ and $overlap\_size = 5$. The system has the same performance on either sides, when $overlap\_size \leq 4$ or $overlap\_size > 4$. Similarly, the same BLEU can be found in Figure 8.12(b) but only when $overlap\_size \leq 4$, whereas great leaps happen when $overlap\_size > 4$. However, some drops are also noticeable (filled circle points in Figure 8.12(b)) on this side ($overlap\_size > 4$). We suspect that this happens because the lengths of the output representations of the VGG blocks are directly affected by a rounding operation conducted at each block. In detail, at each encoding step t, the encoder processes $(s + overlap\_size)$ speech frames. After 2 VGG blocks, the length of the output representation is $len(VGG\_output) = round(round((s + overlap\_size)/2)/2)$. We recall that in our overlap-and-compensate encoding approach, the last positions of the VGG_output are discarded: $len\_discard = round(round(overlap\_size/2)/2)$. Therefore, the final representation of step t would have the length: $len(h_t) = len(VGG\_output) - len\_discard$. Figure 8.12(b) shows that all drops in BLEU are corresponding to $len(h_t) = 2$. This indicates that when working with a speech encoder that contains CNN layers, which is a fairly common practice, one should pay attention at the way input speech is read and processed (s and *overlap_size*).

8.6 Conclusion

In this chapter, a simple yet efficient online decoding strategy, which allows pre-trained end-to-end offline AST models to decode in online mode, is presented. This strategy is inspired by wait-k decoding policy proposed for text-to-text translation, but allows the system to read more than one speech frame in order to generate more than one output token at each step. We conduct an empirical evaluation on models trained for 2 different language pairs En→De and En→Pt, with either characters or BPEs, whose results show that this strategy allows us to control the whole range of latency regimes. Moreover, it can lead to decent BLEU/AL trade-offs in a latency regime of 2 seconds, and our best settings lead to competitive results in comparison with a strong cascade baseline, without re-training the models in online mode.

Secondly, we also present in this chapter our investigation on how to encode

efficiently the continuous speech flow. Our experiments show that ULSTM speech encoders outperform BLSTM speech encoders in terms of both inference speed and BLEU/AL trade-off when using the same encoding strategy, which re-encodes from the beginning of the source sequence every time new source frames arrive. Moreover, inspired by this finding, we propose a new encoding strategy called ULSTM overlap-and-compensate which is shown to further improve inference speed and performance of ULSTM speech encoder.

Thirdly, this chapter also presents the impact of segmentation on the BLEU/AL trade-off of the ULSTM overlap-and-compensate strategy. We observe through experiments that this specific encoding method benefits most strongly when using fixed interval boundaries segmentation.

In addition, we propose in this chapter to fine-tune the pre-trained offline model in a training mode more adapted to online translation. Particularly, our experimental results indicate that this approach can further improve BLEU/AL trade-off of the online translation system while keeping a low development budget.

Finally, this chapter also highlights the crucial impact of the *overlap_size* hyperparameter on the performance of the online translation system which utilizes specifically our overlap-and-compensate encoding strategy.

Conclusion

This thesis is centered on exploring neural methods for end-to-end speech translation, specifically focusing on two types of translation systems: (1) offline speech translation and (2) online speech translation.

As regards offline speech translation, we have explored different end-to-end architectures, different data augmentation techniques and different target granularities. Furthermore, we have compared the performance of self-supervised learning from speech representations, particularly wav2vec and wav2vec2.0 features, with the conventional speech representations, for example, Mel filter-bank and MFCC features, when applied to the speech translation task.

As for online speech translation, we have adapted wait-k policy for online speech translation, and have proposed a new ULSTM overlap-and-compensate encoding strategy. These two methods are combined together and evaluated firstly on pre-trained offline end-to-end speech translation models leveraged in online mode. Moreover, we have proposed to fine-tune these models in online mode to further boost the performance of the online translation systems. In addition, we have also investigated other aspects of online speech translation, for instance, the impact of input speech segmentation, the impact of output granularity, and different fine-tuning scenarios.

This thesis shall be concluded with a summary of our contributions and a discussion about our perspective on the future work.

Summary

- **Chapter 6: Offline Neural Speech Translation.** Our experiments on two language pairs En→De and En→Pt show that LSTM-based encoder-decoder with attention architecture is the most suitable for our settings. This model consists of a speech encoder, which stacks two VGG-like CNN blocks before five layers of BLSTM, and a decoder of two LSTM layers. We exploit Bahdanau's attention mechanism to bridge the encoder and the decoder. This model is trained on different combinations of speech translation corpora, which significantly helps improve the performance of the translation system in comparison with individual corpus. Furthermore, our experimental results on both language pairs are in favor of using characters as output tokens rather than using BPE units. Finally, we observe that ensembling is remarkably beneficial, while fine-tuning the pre-trained model, which had been trained on the combina-

tion of different corpora, on the target corpus has limited or no impact on the performance of the system.

- **Chapter 7: Self-supervised Learning Speech Representation.** We investigate the impact of self-supervised learning from speech on end-to-end AST performance. Particularly, we compare speech features extracted from a pre-trained English wav2vec model, a contrastive predictive coding model pre-trained from unlabeled speech, with filter-banks features on the AST downstream task whose source language is English. Our experimental results show that self-supervised learning features significantly outperform filter-bank features in low and medium-resource settings, when the amount of training data is smaller than 100 hours. Furthermore, we observe that fine-tuning this model on the AST training data (in an unsupervised manner) further improves the performance. In addition, in higher resource settings, we see that ensembling AST models trained with filter-bank and wav2vec features leads to near state-of-the-art models without using any ASR pre-training. Our analyses indicate that self-supervised representations show better phone discrimination, source-target alignments and speaker robustness.

  Besides, we train seven wav2vec2.0 models of different model sizes on different combinations of training data of unlabelled French speech. In our open-source and reproducible framework for assessing self-supervised learning from French speech data named LeBenchmark, we show that, on a multilingual AST setting, features extracted from these self-supervised learning models marginally outperform the baselines using filter-bank features, and that fine-tuning self-supervised learning models on the task-specific data in a supervised manner can lead to great improvements.

- **Chapter 8: Online Neural Speech Translation.** We adapt wait-k decoding policy for speech translation and show that this policy allows to leverage pre-trained offline models, either character-based or BPE-based, of two different language pairs En→De and En→Pt, in online decoding mode. Our experimental results show that this strategy can control the whole range of latency regimes, and can achieve decent BLEU/AL trade-offs in a latency regime of 2 seconds. Furthermore, our best settings are comparable with a strong cascade baseline, without re-training the models in online mode.

  We also advocate for replacing BLSTM-based by ULSTM-based speech encoders when our experimental results indicate that the latter outperforms the former in terms of both inference speed and BLEU/AL

trade-off when using the same encoding strategy, which re-encodes from the beginning of the source sequence every time new source frames are read. We go further to propose a new encoding strategy called ULSTM overlap-and-compensate which is shown to further improve inference speed and performance of ULSTM speech encoders. Furthermore, our investigation on the impact of segmentation on the BLEU/AL trade-off of the ULSTM overlap-and-compensate strategy shows that this specific encoding method benefits most strongly when using fixed interval boundaries segmentation.

Finally, we propose to fine-tune the pre-trained offline model in a training more adapted to online translation. Particularly, our results indicate that this approach can further improve BLEU/AL trade-off of the online translation system while keeping a low development budget.

Future work

In this thesis, we have presented our attempt to answer our research questions, which also evoke other interesting scientific questions. We discuss some future perspectives and open questions based on the work of this thesis as the following:

- **Offline Neural Speech Translation**. So far in this thesis, we have presented AST results that are in favor of using an LSTM-based architecture instead of the Transformer-based counterpart, which is dominantly used in MT applications. This should be taken with caution because we could not fully explore Transformer-based approaches due to the time constraints imposed by the two evaluation campaigns. However, we acknowledge that gathered experimental evidence has induced mixed conclusions about Transformer-based architectures applied to the AST task. For example, Pino et al. (2019) show that their VGGTRANSFORMER AST model significantly outperforms the VGGLSTM one, but possessing substantially more parameters. Similarly, both Potapczyk and Przybysz (2020) and Bahar et al. (2020), whose results rank higher than ours in the end-to-end offline AST track of the IWSLT 2020, use different variants of Transformer architectures for their primary AST systems. However, the differences in the experimental conditions (in terms of parameter budget, amount of training data, etc.) do not allow a direct comparison between LSTM-based and Transformer-based architectures. Di Gangi et al. (2019b,c) conduct a deliberate comparison between LSTM-based and Transformer-based AST models under fairer experi-

mental conditions. Their conclusion is that even though Transformer-based AST models can be trained faster with the same computational budget, it needs additional treatments, for instance, additional 2D CNN layers and distance penalty, in order to yield comparable performance with the LSTM-based architecture. For these reasons, we are interested in conducting a more careful investigation of Transformer architectures for end-to-end AST in the future.

Earlier in this thesis, we have presented that we can leverage a larger amount of unlabelled speech data to improve the performance of the AST task by using pre-trained SSL models. In NMT, non-parallel text data can also be used to pre-train contextualized language models such as BART (Lewis et al. (2019)) and mBART (Liu et al. (2020)) which can be used to improve the performance of the NMT system. Le et al. (2021) use a pre-trained mBART model in order to initialize the decoders of their AST models whose speech encoders are initialized by pre-trained ASR models. They show promising results when fine-tuning these backbones with some adapter layers (Bapna et al. (2019)) on their multilingual AST data. We also find this idea of using pre-trained contextualized language models in the AST task appealing and would like to include it in our future work.

Another interesting future direction on that we want to embark is adapting the k-nearest-neighbor machine translation (kNN-MT) proposed by Khandelwal et al. (2021) to the speech translation task. This method consists of using a nearest retrieval mechanism to aid the decoder of a pre-trained MT model in making a better prediction about the target word at a specific step. More specifically, it relies on interpolating the target token softmax distribution from the pre-trained MT model with a multinomial outputted by the nearest neighbor search over examples cached in a "datastore". This cache is built over translation contexts (i.e, the complete source and prefix target tokens) and is indexed by hidden states calculated by the pre-trained MT model.

Last but not least, most works in the literature make a claim that one of the reasons they advocate for end-to-end AST approaches instead of the cascaded ones is that they are cheaper in terms of development cost taking into account that we need to optimize only one objective. However, we have been seeing that it is not easy for end-to-end models to reach the performance of cascaded models if not for using additional methods such as data augmentation and pre-training, which also come with a cost. For fairer comparisons between these two types of speech translation systems, we believe that all costs should be considered and how

to make these costs comparable is also an interesting scientific question that we are eager to answer.

- **Self-supervised Learning Speech Representation**. So far in this thesis, we have shown promising results when using pre-trained SSL models as features extractors for the AST task. If we view a pre-trained SSL model as an additional component of the speech encoder, we can say that its parameters are frozen during the training of the whole model. However, we cannot think of any particular reason that prevents us from training this component with the AST model. Note that this is somewhat similar to the way Baevski et al. (2020b) fine-tune their wav2vec2.0 model on the ASR downstream task, stacking a randomly initialized output layer on top of the wav2vec2.0 model for predicting target tokens. We can expand this idea to the AST task to exploit a wide collection of decoders and additional adapter layers. For example, we can use a wav2vec2.0 model pre-trained on unlabelled source speech to initialize a speech encoder and a BART model pre-trained on the target text to initialize the decoder of an end-to-end AST model, and train this model on the parallel AST dataset. We acknowledge the similar idea in very recent works such as Gállego et al. (2021); Li et al. (2021) and Babu et al. (2021).

  Another interesting research direction that we want to discover is learning a shared latent space of both acoustic and textual representations. This idea is similar to those of Agrawal et al. (2020) and Han et al. (2021). By projecting textual representations onto a shared space with speech representations, we can guide the speech representations to be closer to the textual ones. Intuitively, this helps us leverage both unlabelled speech data and monolingual text data or massive MT corpora, which are more abundantly available than the parallel data of speech paired with the corresponding translation text.

- **Online Neural Speech Translation**. One of our short-term plans is to explain the surprising results shown Chapter 8, which regard to the inferior performance of the oracle word boundaries segmentation in comparison with the fixed interval one. Even though more deliberate experiments need to be conducted in order to explain this, our hypothesis is that our translation model heavily under-generates target tokens because a word segment can be arbitrarily long. This subsequently lengthens the delay incurred by the system when translating the same sentence. For this reason, a more dynamic way of using the write parameter N might need to be investigated.

  We are also interested in developing more dynamic online speech trans-

lation systems which, instead of functioning on fixed intervals of speech frames, can detect word boundaries and make READ/WRITE decisions based on the seen context. We have seen that CTC can be exploited for word boundaries detection in Ren et al. (2020) whose similar idea can be found in Zeng et al. (2021). Also attempting to detect word boundaries automatically, Wang et al. (2020c) train a separate Scout Network which performs word boundary detection. This information is then used to guide the streaming ASR system. We would like to adapt this idea to the online speech translation task.

In terms of latency metrics, as pointed out in Ma et al. (2021), most currently used latency metrics fail to represent the time spent on generating the translation when analyzing the latency. This weak assumption allows a system to have good latency quality trade-offs, while being inapplicable in real-time scenarios. Furthermore, although the computation-aware version of AL considers the real-time needed to generate each output token, it does not allow to directly compare different systems running on different hardware configurations. Therefore, designing a new latency metric which informatively reflexes the real time delays incurred by online speech translation systems and, at the same time, taking into account the differences in terms of hardware configurations of these systems is well considered for our future work.

French translation

Annexe A

Introduction (française)

En passant de la traduction automatique texte-texte à la traduction automatique parole-texte, on se rapproche un peu plus du rêve séculaire de l'humanité, qui est de supprimer la barrière de la langue entre les personnes de communautés différentes.

Les efforts de traduction automatique de la parole remontent aux années 1980, lorsque NEC a réalisé la première preuve de concept lors de l'ITU Telecom World de 1983 (Nakamura (2009)) et ont continué à prospérer dans les années 1990. Pendant des décennies, le domaine de la traduction vocale (de la parole au texte en particulier) a été le témoin de la prédominance des approches complexes en cascade à deux étapes, qui couplent un système de reconnaissance automatique de la parole suivi d'un système de traduction automatique du texte au texte. Pour cette raison, on dit généralement que ce domaine est beaucoup plus difficile que la reconnaissance vocale et la traduction de texte seules, car il doit résoudre des problèmes provenant de ces deux fronts, qui, à leur tour, sont loin d'être résolus.

À partir de 2016, des approches neuronales de bout en bout, qui cherchaient à remédier aux lacunes des méthodes en cascade, ont émergé et ont directement remis en question la domination des approches en cascade, vieille de plusieurs décennies. Les pionniers de cette branche de la recherche (Duong et al. (2016); Bérard et al. (2016); Weiss et al. (2017); Bérard et al. (2018)) soutiennent que les modèles en cascade à deux étages sont sujets à la propagation d'erreurs puisque les deux principaux composants du système sont entraînés pour optimiser deux fonctions objectives distinctes. Par conséquent, ils proposent d'entraîner des systèmes de traduction de la parole unique qui prédisent directement la traduction à partir de la séquence d'entrée, en optimisant une fonction objectif unique. Cette façon de faire a progressivement suscité l'intérêt de la communauté, grâce à l'efficacité prouvée des modèles de séquence à séquence pour la traduction automatique, et les tâches de reconnaissance de la parole (Chorowski et al. (2015); Chan et al. (2016); Zhang et al. (2017); Chorowski and Jaitly (2016)), et aux efforts pour construire des corpus de traduction de la parole (c'est-à-dire des données parallèles de parole enregistrée couplées à du texte de traduction) qui permettent d'entraîner des modèles de traduction de la parole de bout en bout sans utiliser la transcription source. Depuis 2016, ce domaine de recherche connaît un essor vigoureux.

Cependant, en raison des énormes défis imposés par cette tâche, il est encore loin d'être résolu Sperber and Paulik (2020).

Les modèles de traduction vocale de bout en bout susmentionnés, qui ont l'avantage de disposer de la séquence d'entrée complète pour conditionner la génération de la traduction de sortie, sont appelés traduction vocale hors ligne. On dit qu'elle est moins difficile que la traduction vocale en ligne (parfois appelée *"simultanée"*) de bout en bout (Ren et al. (2020); Ma et al. (2020b); Han et al. (2020); Ma et al. (2021)), qui doit générer une hypothèse de sortie de manière incrémentielle à partir de la parole partielle d'entrée. Cette branche de recherche plutôt naissante attire de plus en plus l'attention de la communauté grâce aux résultats préliminaires mais encourageants présentés lors de campagnes d'évaluation telles que les IWSLT 2020 et 2021. Cependant, le problème de la traduction en ligne de bout en bout est loin d'être résolu, et les tentatives d'amélioration du statu quo dans ce domaine sont donc les bienvenues.

Avant tout, cette thèse est centrée sur l'exploration des méthodes neuronales pour la traduction de la parole de bout en bout. Il s'agit d'un large éventail d'aspects, tels que la recherche de modèles de bout en bout efficaces, la manière de gérer la rareté des données d'entraînement à la traduction de la parole, la manière de segmenter efficacement l'entrée de la parole, le type d'unités token cibles le plus avantageux, etc. Il s'agit également de répondre à une question de recherche sur la manière de représenter efficacement les données d'entrée de la parole, ce qui constitue le second objectif de cette thèse. En particulier, une étude de la nouvelle proposition d'apprentissage auto-supervisé à partir de représentations de la parole (Schneider et al. (2019a); Baevski et al. (2020b))) est menée, ce qui permettra de remplacer les approches conventionnelles de représentation de la parole, telles que les représentations Mel fitler-bank et les représentations MFCC, etc. par ces représentations d'apprentissage auto-supervisé.

Enfin, une autre contribution de cette thèse concerne la traduction vocale en ligne. En particulier, nous visons à adapter la stratégie wait-k (Ma et al. (2019)), une stratégie de décodage proposée pour la traduction de texte à texte en ligne, à la traduction de parole en ligne. Nous couplons cette stratégie de décodage adaptée avec une nouvelle stratégie d'encodage ULSTM overlap-and-compensate afin, d'une part, d'exploiter en mode en ligne des modèles de traduction vocale de bout en bout hors ligne pré-entraînés et, d'autre part, d'affiner ces modèles en mode en ligne pour améliorer encore les performances des systèmes de traduction en ligne. D'autres aspects différents de la traduction vocale en ligne, par exemple, l'impact de la segmentation de la parole en entrée et l'impact de la granularité en sortie, sont également étudiés dans cette thèse.

Aperçu des contributions

Cette thèse étudie le problème de la modélisation des systèmes neuronaux de traduction de la parole, en se concentrant sur les aspects suivants:

- Modélisation de bout en bout pour la traduction vocale hors ligne: nous nous concentrons sur l'exploration de différentes architectures de bout en bout pour la traduction vocale, ainsi que de différents types de jetons cibles. Nous soutiendrons, par le biais d'expériences, que l'architecture d'encodeur-décodeur attentionnel basée sur LSTM est la plus avantageuse, et que les modèles basés sur les caractères sont plus performants que leurs homologues basés sur BPE dans nos paramètres. Cette quête de modèles de bout en bout pour la traduction vocale hors ligne donne lieu à deux publications pour les campagnes d'évaluation IWSLT 2019 et IWSLT 2020.

- Représentations de la parole pour la traduction: nous comparons les approches conventionnelles de représentation de la parole, telles que les coefficients en bancs de filtres ou MFCC, avec un nouveau type de représentation de la parole basé sur l'apprentissage auto-supervisé. Nous montrons que les représentations wav2vec, qui sont basées sur le codage prédictif contrastif (CPC), surpassent largement les représentations conventionnelles dans des conditions de faibles ressources où les données d'entraînement à la traduction de la parole ne sont pas suffisamment disponibles. Cette étude de l'apprentissage auto-supervisé à partir des représentations de la parole donne lieu à plusieurs publications dans des conférences internationales prestigieuses telles que Interspeech 2020, Interspeech 2021 et NeurIPS 2021.

- Système de traduction en ligne de bout en bout: nous nous concentrons sur l'équilibre entre la qualité de la traduction et la latence des systèmes de traduction de bout en bout. Nous proposons une stratégie de décodage de type wait-k, qui s'avère suffisante pour exploiter des modèles hors ligne pré-entraînés de bout en bout pour la traduction de la parole en ligne. Afin d'améliorer encore les performances du système de traduction en ligne, nous proposons également une stratégie d'encodage, à savoir le ULSTM overlap-and-compensate, qui permet aux encodeurs de parole de type VGG avec des couches ULSTM de encoder plus efficacement l'entrée de parole partielle. En outre, nous préconisons également un réglage fin des modèles hors ligne dans le cadre d'un entraînement plus adaptée à la traduction en ligne, et montrons que cela peut contribuer à améliorer les performances du système de traduction en ligne

tout en conservant un coût de développement raisonnable. Ces travaux ont donné lieu à 2 publications dans 2 conférences internationales de haut niveau, dont l'ICASSP 2021 et l'Interspeech 2021.

Plan de la thèse

Cette thèse est organisée en deux parties principales. Dans la première partie de la thèse, nous discutons des connaissances de base qui sont étroitement liées au travail effectué dans cette thèse. Il s'agit de:

- **Chapitre 1: Traduction automatique neuronale.** Dans ce chapitre, nous présentons un aperçu des anciennes méthodes qui étaient dominantes avant l'ère de la traduction automatique neuronale de bout en bout, par exemple, la traduction automatique statistique basée sur les mots et les phrases. Nous présenterons ensuite l'état de l'art des méthodes neuronales, en particulier la traduction automatique de bout en bout. En outre, ce chapitre décrit également les métriques couramment utilisées pour évaluer un système de traduction automatique.

- **Chapitre 2: Traduction vocale neuronale.** Ce chapitre est consacré à la présentation de l'état de l'art de la traduction vocale neuronale de bout en bout. Avant cela, nous abordons brièvement les systèmes en cascade, qui restent des bases solides que les modèles de bout en bout doivent dépasser. Ensuite, nous mettons l'accent sur certaines architectures de bout en bout importantes, largement utilisées dans la traduction vocale, ainsi que sur les défis et leurs solutions correspondantes. Ce chapitre se termine par un aperçu des corpus de traduction de la parole.

- **Chapitre 3: Traduction automatique neuronale en ligne.** Ce chapitre vise à présenter l'état de l'art de la traduction automatique neuronale en ligne. Dans ce chapitre, nous nous concentrons sur la discussion des différentes stratégies de décodage en ligne de nature déterministe et dynamique. En outre, nous mentionnons également comment un système de traduction automatique en ligne peut être évalué automatiquement.

- **Chapitre 4: Traduction vocale neuronale en ligne.** Dans ce chapitre, nous discutons brièvement des premières tentatives de systèmes de traduction vocale en ligne, avant de donner un aperçu des modèles de traduction vocale en ligne en cascade. Ce chapitre sera suivi d'une discussion sur les modèles de bout en bout pour la traduction vocale en ligne, qui sont étroitement liés à l'objectif de cette thèse.

- **Chapitre 5: Apprentissage auto-supervisé de la représentation de la parole.** Dans ce chapitre, nous donnons d'abord un aperçu des approches conventionnelles de représentation des données de parole. Ensuite, une discussion détaillée centrée sur les méthodes d'apprentissage auto-supervisé de la parole, qui sont prometteuses pour remplacer les approches conventionnelles de représentation de la parole, sera donnée.

La deuxième partie de la thèse vise à détailler les contributions scientifiques de cette thèse, notamment dans les aspects suivants:

- **Chapitre 6: Traduction vocale neuronale hors ligne.** Nous explorons différentes architectures de bout en bout pour la traduction automatique de la parole, La discussion sera confinée dans le contexte de notre participation aux campagnes d'évaluation IWSLT 2019 et IWSLT 2020, où les traductions automatiques générées par ces modèles de bout en bout sont évaluées et comparées avec les méthodes d'autres groupes de recherche.

- **Chapitre 7: Apprentissage auto-supervisé de la représentation de la parole.** L'objectif de ce chapitre est de discuter de nos contributions à l'utilisation du modèle d'apprentissage auto-supervisé pré-entraîné en anglais pour générer des caractéristiques de parole pour la tâche de traduction de la parole, ainsi que de nos efforts pour entraîner nos propres modèles SSL à partir de la parole française. Dans ce chapitre, nous montrons des résultats qui soutiennent l'utilisation de représentations de la parole par apprentissage auto-supervisé au lieu de représentations conventionnelles.

- **Chapitre 8: Traduction vocale neuronale en ligne.** Nous discutons dans ce chapitre de nos contributions en termes d'approches de décodage et d'encodage pour la traduction vocale en ligne, qui utilisent efficacement des modèles de traduction vocale hors ligne pré-entraînés dans la tâche de traduction en ligne. Nous montrons également dans ce chapitre comment ces modèles pré-entraînés peuvent être affinés dans un entraînement plus adaptée à la traduction en ligne afin d'améliorer encore les performances du système de traduction en ligne.

Nous concluons cette thèse dans la **Conclusion**, où nous résumons nos contributions ainsi que notre perspective sur les travaux futurs.

Annexe B

Résumé des Chapitres

B.1 Chapitre 1: Traduction automatique neuronale

Dans ce chapitre, nous présentons brièvement les méthodes traditionnelles de traduction automatique, notamment la traduction automatique à base de règles, qui repose essentiellement sur des règles linguistiques, et la traduction automatique statistique, qui exploite des méthodes statistiques sans se soucier des règles linguistiques. La traduction automatique statistique, qui se compose d'un modèle de langue et d'un modèle de traduction, a été l'état de l'art dans le domaine pendant des décennies, jusqu'à l'émergence de la traduction automatique neuronale, qui est présentée dans un volume plus important de ce chapitre. La méthode dominante dans la traduction automatique neuronale est la modélisation de séquence à séquence de bout en bout. L'exemple le plus représentatif de ce type de modèle est constitué d'un encodeur et d'un décodeur, qui sont reliés par un mécanisme d'attention. En plus de passer en revue plusieurs extensions de réseau pour la traduction automatique neuronale, par exemple, les encodeurs à Long Short-Term Memory (LSTM) et l'architecture Transformer, nous passons également en revue différentes approches visant à améliorer la qualité de la traduction automatique, par exemple, l'utilisation de la beam search, de l'ensemble et des unités de sous-mots. La qualité de la traduction est le plus souvent mesurée par le BLEU, une métrique automatique, qui est présentée dans ce chapitre avec le METEOR, le Word Error Rate (WER) et le Translation Error Rate (TER).

B.2 Chapitre 2: Traduction vocale neuronale

Dans ce chapitre, nous passons en revue les méthodes en cascade qui ont constitué pendant longtemps l'état de l'art en matière de traduction automatique de la parole. Cependant, ces méthodes, qui couplent un système de reconnaissance automatique de la parole avant un système de traduction automatique, attirent moins l'attention récemment en raison de l'apparition de la traduction neuronale de la parole de bout en bout. Cette méthode nouvellement proposée est plus attrayante car elle promet de remédier aux défauts des méthodes en cascade, par exemple, le problème de la propagation des erreurs et le coût

de développement élevé. Cependant, les modèles de bout en bout nécessitent d'être entraînés sur une grande quantité de données de traduction de la parole (c'est-à-dire des données parallèles de la parole alignées sur le texte de traduction correspondant), ce qui n'est pas facile à satisfaire dans la plupart des cas. Parallèlement à l'introduction de nouveaux corpus de traduction vocale, diverses méthodes, dont l'augmentation des données, l'apprentissage multi-tâches et la pré-entraînement, ont été introduites pour atténuer ce problème de disponibilité des données. Nous consacrons une grande partie de ce chapitre à l'examen de différents modèles de bout en bout pour la traduction automatique de la parole, dont la plupart sont basés sur le modèle Listen, Attend, and Spell (LAS). Le modèle LAS est un modèle de séquence à séquence similaire à ceux utilisés en traduction automatique neuronale (de text), mais les encodeurs de la parole sont adaptés pour mieux traiter les représentations de la parole en entrée.

B.3 Chapitre 3: Traduction automatique neuronale en ligne

Dans ce chapitre, nous présentons l'état de l'art de la traduction automatique neuronale en ligne, qui se distingue de la tâche de traduction hors ligne qui a l'avantage de disposer de la séquence d'entrée complète pour conditionner la génération des tokens de sortie. La traduction automatique en ligne, en revanche, doit commencer à générer des hypothèses de sortie partielles sur la base d'une entrée partielle afin d'équilibrer le compromis entre la qualité de la traduction et la latence. Nous nous concentrons sur la discussion des différentes stratégies de décodage en ligne de nature déterministe et dynamique. L'approche dominante des stratégies déterministes est la stratégie wait-k, qui est un cadre préfixe à préfixe consistant en un agent qui lit k tokens source à la première étape, puis alterne les opérations WRITE/READ uniques jusqu'à ce que tous les tokens source soient lus. En revanche, les stratégies dynamiques visent à entraîner l'agent à effectuer des actions de READ/WRITE plus dynamiques ou à apprendre une attention plus monotone au lieu de l'attention douce traditionnelle qui nécessite l'accès à la séquence source complète. Outre la qualité de la traduction, la traduction en ligne est également évaluée en fonction de la latence, qui est le plus souvent mesurée par l'Average Lagging (la latence moyenne). D'autres métriques automatiques de mesure de la latence présentées dans ce chapitre incluent l'Average Proportion (proportion moyenne) et la Differentiable Average Lagging (latence moyenne différentiable).

B.4 Chapitre 4: Traduction vocale neuronale en ligne

Dans ce chapitre, nous présentons l'état de l'art de la traduction vocale en ligne. La plupart des efforts déployés jusqu'à présent dans ce domaine se concentrent sur la construction de systèmes en cascade, qui exploitent des modules de traduction simultanée de texte à texte pour traduire la transcription partielle produite par un module ASR en ligne. En revanche, les tentatives d'exploitation de modèles de bout en bout pour traduire directement et simultanément la parole source en texte cible sont apparues très récemment, principalement en adaptant à la tâche de la parole le modèle wait-k proposé pour la traduction texte-texte. Les premiers modèles de bout en bout effectuent des opérations de READ/WRITE de manière déterministe (c'est-à-dire qu'ils prennent des décisions de READ et WRITE sur des blocs d'un nombre fixe de trames sources), ou sont entraînés conjointement avec un module supplémentaire, par exemple un segmenteur de parole ou un module de prédécision basé sur la CTC, qui évoque des décisions de READ et WRITE dynamiques.

B.5 Chapitre 5: Apprentissage auto-supervisé de la représentation de la parole

Nous passons en revue dans ce chapitre deux approches conventionnelles de représentation de la parole les plus couramment utilisées, à savoir le Log Mel Filterbank et les MFCCs. Ces méthodes utilisent sélectivement des outils mathématiques pour transformer directement les signaux vocaux numérisés en une représentation plus dense. Dans ce chapitre, nous nous concentrons toutefois sur l'apprentissage auto-supervisé à partir de représentations de la parole, qui est une approche plus récente pour l'extraction de caractéristiques de la parole. Plus précisément, nous présentons le codage prédictif contrastif (CPC) qui vise à apprendre des représentations utiles à partir de données non annotées de haute dimension en prédisant le futur dans l'espace latent. Cette approche permet d'exploiter d'énormes quantités de données non annotées pour apprendre des représentations utiles de la parole. Ces représentations sont ensuite utilisées dans des tâches en aval, par exemple la traduction automatique de la parole, qui souffre généralement d'un manque de données. Les différentes variantes du CPC sont présentées dans ce chapitre, à savoir wav2vec et deux de ses extensions, notamment vq-wav2vec, qui apprend des représentations vectorielles quantifiées, et wav2vec2.0, qui masque la parole brute en entrée dans l'espace latent et résout une tâche contrastive définie sur

des représentations vocales quantifiées.

B.6 Chapitre 6: Contribution - Traduction vocale neuronale hors ligne

Dans ce chapitre, nous avons présenté notre quête pour trouver les modèles AST de bout en bout les mieux adaptés à notre condition. Une série d'expériences, réalisées dans le cadre de notre participation à deux campagnes d'évaluation de la traduction vocale, à savoir IWSLT 2019 et IWSLT 2020, montre que l'architecture d'encodeur-décodeur basé sur LSTM avec attention donne de bons résultats dans nos conditions. Nous observons également, par le biais d'expériences, que la combinaison de corpus de traduction vocale permet d'améliorer de manière significative les performances du système de traduction. Les expériences sur les deux paires de langues En→De et En→Pt montrent que les résultats sont en faveur de l'utilisation de caractères comme jetons de sortie. De plus, l'assemblage est considérablement bénéfique, tandis que réglage fin ne montre qu'une amélioration limitée ou nulle des performances du système.

D'après les résultats présentés, les modèles d'encodeur-décodeur avec attention à base de caractères sera systématiquement réutilisé dans nos expériences suivantes, qui seront présentées dans la suite de cette thèse. Cependant, les lecteurs de cette thèse doivent être avertis que, puisque ce domaine de recherche prolifère à grande vitesse, ces résultats peuvent ne pas correspondre à ceux plus récents d'autres groupes de recherche.

B.7 Chapitre 7: Contribution - Apprentissage auto-supervisé de la représentation de la parole

Dans ce chapitre, nous avons présenté notre étude sur l'impact de l'apprentissage auto-supervisé (SSL) à partir de la parole sur les performances de la traduction automatique de la parole de bout en bout. Il s'agit, à notre connaissance, de l'un des premiers efforts d'utilisation du SSL pour la tâche AST. Plus précisément, nous utilisons un modèle wav2vec anglais pré-entraîné, un modèle CPC pré-entraîné à partir de la parole non étiquetée, comme extracteur de caractéristiques pour une tâche AST en aval qui concerne l'anglais comme langue source. Nos résultats expérimentaux montrent que le pré-entraînement auto-supervisé est particulièrement efficace dans les contextes de ressources faibles et moyennes, lorsque la quantité de données d'entraînement à la traduction vocale est inférieure à 100 heures, et que réglage fin des modèles CPC sur

les données d'entraînement AST améliore encore les performances. En outre, dans des contextes de ressources plus élevées, nous observons par le biais d'expériences que l'assemblage de modèles AST entraînés avec des représentations de filterbanks et de CPC conduit à des modèles proches de l'état de l'art sans utiliser de pré-entraînement ASR. Nos analyses montrent que les représentations auto-supervisées présentent une meilleure discrimination du phonème, des alignements source-cible et une meilleure robustesse à la variabilité des orateurs, ce qui pourrait être responsable de cette amélioration significative par rapport au base de référence de filterbanks. Ceci pourrait être particulièrement bénéfique dans la situation où nous devons développer un système qui traduit à partir de la parole dans une langue dont l'orthographe est peu standardisée ou même à partir de la parole dans une langue non écrite.

En outre, ce chapitre a également présenté nos contributions à l'entraînement de plusieurs modèles SSL à partir de la parole française. En particulier, nous entraînons 7 modèles wav2vec2.0 de différentes tailles sur différentes combinaisons de données d'entraînement. Ces modèles sont utilisés dans notre cadre open-source et reproductible pour évaluer le SSL à partir de données vocales françaises, à savoir ***LeBenchmark***. Nous montrons des résultats sur un cadre AST multilingue, qui réaffirment que les représentations SSL surpassent marginalement les bases de référence de filterbanks, et que le réglage fin des modèles SSL sur les données spécifiques à la tâche d'une manière supervisée peut conduire à de grandes améliorations.

Enfin, nous devons préciser que même si ce type de représentations de la parole s'est avéré très efficace pour la tâche d'AST, dans la suite de cette thèse, nous reviendrons à l'utilisation de filterbanks dans les expériences concernant l'AST en ligne. Ceci est dû à la contrainte de calcul que nous avons rencontrée au cours du processus. L'AST en ligne s'avérera plus coûteuse en termes de calcul, et nous n'avons donc pas été en mesure d'appliquer les fonctionnalités SSL à cette tâche.

B.8 Chapitre 8: Contribution - Traduction vocale neuronale en ligne

Dans ce chapitre, nous présentons une stratégie de décodage en ligne simple mais efficace, qui permet aux modèles hors ligne AST pré-entraînés de bout en bout de décoder en mode en ligne. Cette stratégie est inspirée de la stratégie de décodage wait-k proposée pour la traduction de texte à texte, mais permet au système de lire plus d'une trame de parole afin de générer plus d'un jeton de sortie à chaque étape. Nous effectuons une évaluation empirique sur des modèles pre-entraînés pour 2 paires de langues différentes En→De et En→Pt, avec

des caractères ou des BPEs, dont les résultats montrent que cette stratégie nous permet de contrôler toute la gamme des régimes de latence. De plus, elle peut conduire à des compromis BLEU/AL décents dans un régime de latence de 2 secondes, et nos meilleurs paramètres conduisent à des résultats compétitifs par rapport à une base de référence de cascade forte, sans ré-entraîner les modèles en mode en ligne.

Deuxièmement, nous présentons également dans ce chapitre notre étude sur la manière d'encoder efficacement le flux de parole continu. Nos expériences montrent que les encodeurs de parole ULSTM sont plus performants que les encodeurs de parole BLSTM en termes de vitesse d'inférence et de compromis BLEU/AL lorsqu'ils utilisent la même stratégie d'encodage, qui réencode depuis le début de la séquence source à chaque fois que de nouvelles trames sources arrivent. De plus, inspirés par cette découverte, nous proposons une nouvelle stratégie d'encodage appelée ULSTM overlap-and-compensate qui permet d'améliorer encore la vitesse d'inférence et les performances du encodeur vocal ULSTM.

Troisièmement, ce chapitre présente également l'impact de la segmentation de la parole sur le compromis BLEU/AL de la stratégie overlap-and-compensate ULSTM. Nous observons, par le biais d'expériences, que cette méthode d'encodage spécifique est plus avantageuse lorsqu'elle utilise une segmentation des frontières (de mots) à intervalle fixe.

En outre, nous proposons dans ce chapitre d'affiner le modèle hors ligne pré-entraîné dans un mode d'apprentissage plus adapté à la traduction en ligne. En particulier, nos résultats expérimentaux indiquent que cette approche peut améliorer davantage le compromis BLEU/AL du système de traduction en ligne tout en conservant un faible budget de développement.

Enfin, ce chapitre met en évidence l'impact crucial de l'hyperparamètre *overlap_size* sur les performances du système de traduction en ligne qui utilise spécifiquement notre stratégie d'encodage overlap-and-compensate.

Annexe C

Conclusion (française)

Cette thèse est centrée sur l'exploration des méthodes neuronales pour la traduction vocale de bout en bout, en se concentrant spécifiquement sur deux types de systèmes de traduction : (1) la traduction vocale hors ligne et (2) la traduction vocale en ligne.

En ce qui concerne la traduction vocale hors ligne, nous avons exploré différentes architectures de bout en bout, différentes techniques d'augmentation des données et différentes granularités de cible. En outre, nous avons comparé les performances de l'apprentissage auto-supervisé à partir de représentations de la parole, en particulier les représentations wav2vec et wav2vec2.0, avec les représentations conventionnelles de la parole, par exemple, les Mel filter-bank et MFCCs, lorsqu'elles sont appliquées à la tâche de traduction de la parole.

En ce qui concerne la traduction de la parole en ligne, nous avons adapté la stratégie wait-k à la traduction de la parole en ligne, et nous avons proposé une nouvelle stratégie d'encodage ULSTM overlap-and-compensate. Ces deux méthodes sont combinées ensemble et évaluées d'abord sur des modèles de traduction vocale hors ligne pré-entraînés de bout en bout exploités en mode en ligne. De plus, nous avons proposé d'affiner ces modèles en mode en ligne afin d'améliorer les performances des systèmes de traduction en ligne. En outre, nous avons également étudié d'autres aspects de la traduction vocale en ligne, par exemple, l'impact de la segmentation de la parole en entrée, l'impact de la granularité en sortie, et différents scénarios de réglage fin.

Cette thèse sera conclue par un résumé de nos contributions et une discussion sur notre perspective sur le travail futur.

Sommaire

- **Chapter 6: Traduction vocale neuronale hors ligne.** Nos expériences sur deux paires de langues En→De et En→Pt montrent que l'encodeur-décodeur basé sur un LSTM avec architecture d'attention est le plus adapté à nos paramètres. Ce modèle se compose d'un encodeur vocal, qui empile deux blocs CNN de type VGG avant cinq couches de BLSTM, et d'un décodeur composé de deux couches LSTM. Nous exploitons le mécanisme d'attention de Bahdanau pour faire le lien entre l'encodeur

et le décodeur. Ce modèle est entraîné sur différentes combinaisons de corpus de traduction vocale, ce qui permet d'améliorer considérablement les performances du système de traduction par rapport à un corpus individuel. De plus, nos résultats expérimentaux sur les deux paires de langues sont en faveur de l'utilisation de caractères comme jetons de sortie plutôt que d'utiliser des unités BPE. Enfin, nous observons que l'assemblage est remarquablement bénéfique, tandis que le réglage fin du modèle pré-entraîné, qui avait été entraîné sur la combinaison de différents corpus, sur le corpus cible a un impact limité ou nul sur les performances du système.

- **Chapter 7: Apprentissage auto-supervisé de la représentation de la parole.** Nous étudions l'impact de l'apprentissage auto-supervisé à partir de la parole sur les performances de l'AST de bout en bout. En particulier, nous comparons les présentations de la parole extraites d'un modèle wav2vec anglais pré-entraîné, un modèle de codage prédictif contrastif pré-entraîné à partir de la parole non étiquetée, avec les présentations des bancs de filtres sur la tâche AST en aval dont la langue source est l'anglais. Nos résultats expérimentaux montrent que les présentations d'apprentissage auto-supervisé surpassent de manière significative les filterbanks dans des contextes de ressources faibles et moyennes, lorsque la quantité de données d'entraînement est inférieure à 100 heures. De plus, nous observons que le réglage fin de ce modèle sur les données d'entraînement AST (de manière non supervisée) améliore encore les performances. En outre, dans des paramètres de ressources plus élevés, nous constatons que l'assemblage de modèles AST entraînés avec des filterbanks et présentations wav2vec conduit à des modèles proches de l'état de l'art sans utiliser de pré-entraînement ASR. Nos analyses indiquent que les représentations auto-supervisées présentent une meilleure discrimination des phonèmes, des alignements source-cible et une meilleure robustesse à la variabilité du locuteur.

  En outre, nous entraînons sept modèles wav2vec2.0 de tailles différentes sur différentes combinaisons de données d'entraînement de parole française non étiquetée. Dans notre cadre ouvert et reproductible d'évaluation de l'apprentissage auto-supervisé à partir de données vocales françaises, appelé LeBenchmark, nous montrons que, dans un contexte d'AST multilingue, les caractéristiques de parole extraites de ces modèles d'apprentissage auto-supervisé surpassent marginalement les bases de référence utilisant des filterbanks, et que le réglage fin des modèles d'apprentissage auto-supervisé sur les données spécifiques à la tâche d'une manière supervisée peut conduire à de grandes améliorations.

- **Chapter 8: Traduction vocale neuronale en ligne.** Nous adaptons la stratégie de décodage wait-k à la traduction de la parole et montrons que cette stratégie permet d'exploiter des modèles hors ligne pré-entraînés, basés sur les caractères ou sur le BPE, de deux paires de langues différentes, En→De et En→Pt, en mode de décodage en ligne. Nos résultats expérimentaux montrent que cette stratégie peut contrôler toute la gamme des régimes de latence, et peut obtenir des compromis BLEU/AL décents dans un régime de latence de 2 secondes. De plus, nos meilleurs paramètres sont comparables à une base de référence de cascade forte, sans ré-entraîner les modèles en mode en ligne.

  Nous préconisons également le remplacement des encodeurs vocaux basés sur les BLSTM par des encodeurs basés sur les ULSTM lorsque nos résultats expérimentaux indiquent que ces derniers sont plus performants que les premiers en termes de vitesse d'inférence et de compromis BLEU/AL lorsque l'on utilise la même stratégie d'encodage, qui ré-encode depuis le début de la séquence source à chaque fois que de nouvelles trames sont lues. Nous allons plus loin en proposant une nouvelle stratégie d'encodage appelée ULSTM overlap-and-compensate qui permet d'améliorer la vitesse d'inférence et les performances des encodeurs vocaux ULSTM. En outre, notre étude de l'impact de la segmentation sur le compromis BLEU/AL de la stratégie de ULSTM overlap-and-compensate montre que cette méthode d'encodage spécifique est plus avantageuse lorsqu'elle utilise une segmentation à intervalles fixes.

  Enfin, nous proposons d'affiner le modèle hors ligne pré-entraîné dans un entraînement plus adapté à la traduction en ligne. En particulier, nos résultats indiquent que cette approche peut améliorer le compromis BLEU/AL du système de traduction en ligne tout en conservant un faible budget de développement.

Travaux futurs

Dans cette thèse, nous avons présenté notre tentative de réponse à nos questions de recherche, qui évoquent également d'autres questions scientifiques intéressantes. Nous discutons des perspectives futures et des questions ouvertes basées sur le travail de cette thèse comme suit:

- **Traduction vocale neuronale hors ligne**. Jusqu'à présent dans cette thèse, nous avons présenté des résultats d'AST qui sont en faveur de l'utilisation d'une architecture basée sur LSTM au lieu de la contrepartie

basée sur Transformer, qui est principalement utilisée dans les applications de MT. Ceci doit être pris avec précaution car nous n'avons pas pu explorer complètement les approches basées sur Transformer en raison des contraintes de temps imposées par les deux campagnes d'évaluation. Cependant, nous reconnaissons que les preuves expérimentales recueillies ont induit des conclusions mitigées sur les architectures basées sur Transformer appliquées à la tâche AST. Par exemple, Pino et al. (2019) montrent que leur modèle VGGTRANSFORMER AST surpasse significativement le modèle VGGLSTM, mais en possédant beaucoup plus de paramètres. De même, Potapczyk and Przybysz (2020) et Bahar et al. (2020), dont les résultats sont supérieurs aux nôtres dans la piste AST hors ligne de bout en bout de l'IWSLT 2020, utilisent différentes variantes de l'architecture Transformer pour leurs systèmes AST primaires. Cependant, les différences dans les conditions expérimentales (en termes de budget de paramètres, de quantité de données d'entraînement, etc.) ne permettent pas une comparaison directe entre l'architecture basée sur LSTM et celle basée sur Transformer. Di Gangi et al. (2019b,c) effectuent une comparaison délibérée entre les modèles AST basés sur LSTM et ceux basés sur Transformer dans des conditions expérimentales plus justes. Leur conclusion est que même si les modèles AST basés sur Transformer peuvent être entraînés plus rapidement avec le même budget de calcul, ils nécessitent des traitements supplémentaires, par exemple, des couches 2D CNN supplémentaires et une pénalité de distance, afin d'obtenir des performances comparables à celles de l'architecture basée sur LSTM. Pour ces raisons, nous souhaitons mener à l'avenir une étude plus approfondie des architectures Transformer pour l'AST de bout en bout.

Plus tôt dans cette thèse, nous avons présenté que nous pouvons exploiter une plus grande quantité de données vocales non étiquetées pour améliorer les performances de la tâche AST en utilisant des modèles SSL pré-entraînés. En NMT, les données textuelles non parallèles peuvent également être utilisées pour pré-entraîner des modèles de langage contextualisés tels que BART (Lewis et al. (2019)) et mBART (Liu et al. (2020)) qui peuvent être utilisés pour améliorer les performances du système NMT. Le et al. (2021) utilisent un modèle mBART pré-entraîné afin d'initialiser les décodeurs de leurs modèles AST dont les encodeurs de parole sont initialisés par des modèles ASR pré-entraînés. Ils montrent des résultats prometteurs en affinant ces dorsales avec quelques couches d'adaptation (Bapna et al. (2019)) sur leurs données AST multilingues. Nous trouvons également attrayante l'idée d'utiliser des mod-

èles de langage contextualisés pré-entraînés dans la tâche AST et nous aimerions l'inclure dans nos travaux futurs.

Une autre direction future intéressante sur laquelle nous voulons nous engager est l'adaptation de la traduction automatique par k-nearest-neighbor (kNN-MT) proposée par Khandelwal et al. (2021) à la tâche de traduction de la parole. Cette méthode consiste à utiliser un mécanisme de récupération des plus proches pour aider le décodeur d'un modèle de MT pré-entraîné à faire une meilleure prédiction du mot cible à une étape spécifique. Plus précisément, il s'appuie sur l'interpolation de la distribution de la softmax du token cible à partir du modèle de MT pré-entraîné avec une multinomiale produite par la recherche du plus proche voisin sur des exemples mis en cache dans un "datastore". Ce cache est construit sur des contextes de traduction (c'est-à-dire, la source complète et le préfixe des tokens cibles) et est indexé par des états cachés calculés par le modèle de MT pré-entraîné.

Enfin, la plupart des travaux dans la littérature affirment que l'une des raisons pour lesquelles ils préconisent les approches AST de bout en bout plutôt que les approches en cascade est qu'elles sont moins coûteuses en termes de développement, compte tenu du fait que nous devons optimiser un seul objectif. Cependant, nous avons constaté qu'il n'est pas facile pour les modèles de bout en bout d'atteindre les performances des modèles en cascade si ce n'est en utilisant des méthodes supplémentaires telles que l'augmentation des données et la pré-entraînement, qui ont également un coût. Pour des comparaisons plus justes entre ces deux types de systèmes de traduction de la parole, nous pensons que tous les coûts devraient être pris en compte et la manière de rendre ces coûts comparables est également une question scientifique intéressante à laquelle nous sommes impatients de répondre.

- **Apprentissage auto-supervisé de la représentation de la parole**. Jusqu'à présent dans cette thèse, nous avons montré des résultats prometteurs en utilisant des modèles SSL pré-entraînés comme extracteurs de caractéristiques pour la tâche AST. Si nous considérons un modèle SSL pré-entraîné comme un composant supplémentaire de l'encodeur vocal, nous pouvons dire que ses paramètres sont gelés pendant l'entraînement du modèle entier. Cependant, nous ne pouvons pas penser à une raison particulière qui nous empêche d'entraîner ce composant avec le modèle AST. Nous notons que cela est quelque peu similaire à la manière dont Baevski et al. (2020b) affine son modèle wav2vec2.0 sur la tâche ASR en aval, en empilant une couche de sortie initialisée de manière aléatoire sur le modèle wav2vec2.0 pour prédire les tokens cibles. Nous pouvons

étendre cette idée à la tâche AST pour exploiter une large collection de décodeurs et de couches d'adaptation supplémentaires. Par exemple, nous pouvons utiliser un modèle wav2vec2.0 pré-entraîné sur la parole source non étiquetée pour initialiser un encodeur de parole et un modèle BART pré-entraîné sur le texte cible pour initialiser le décodeur d'un modèle AST de bout en bout, et entraîner ce modèle sur le jeu de données AST parallèle. Nous reconnaissons l'idée similaire dans des travaux très récents tels que Gállego et al. (2021); Li et al. (2021) et Babu et al. (2021).

Une autre direction de recherche intéressante que nous voulons découvrir est l'apprentissage d'un espace latent partagé des représentations acoustiques et textuelles. Cette idée est similaire à celles de Agrawal et al. (2020) et Han et al. (2021). En projetant les représentations textuelles sur un espace partagé avec les représentations vocales, nous pouvons guider les représentations vocales pour les rapprocher des représentations textuelles. Intuitivement, cela nous permet d'exploiter à la fois des données vocales non étiquetées et des données textuelles monolingues ou des corpus massifs de MT, qui sont plus abondamment disponibles que les données parallèles de la parole couplée au texte de traduction correspondant.

- **Traduction vocale neuronale en ligne**. L'un de nos projets à court terme est d'expliquer les résultats surprenants présentés dans le chapitre 8, qui concernent les performances inférieures de la segmentation des frontières de mots de l'oracle par rapport à celle de l'intervalle fixe. Même si des expériences plus approfondies doivent être menées pour expliquer ce résultat, notre hypothèse est que notre modèle de traduction sous-génère fortement les tokens cibles car un segment de mot peut être arbitrairement long. Cela allonge par la suite le délai encouru par le système lors de la traduction de la même phrase. Pour cette raison, il faudrait peut-être étudier une manière plus dynamique d'utiliser le paramètre d'écriture N.

  Nous sommes également intéressés par le développement de systèmes de traduction vocale en ligne plus dynamiques qui, au lieu de fonctionner sur des intervalles fixes de trames vocales, peuvent détecter les frontières des mots et prendre des décisions de READ/WRITE en fonction du contexte observé. Nous avons vu que CTC peut être exploité pour la détection des frontières de mots dans Ren et al. (2020) dont l'idée similaire peut être trouvée dans Zeng et al. (2021). Pour tenter de détecter automatiquement les frontières des mots, Wang et al. (2020c) entraînent un réseau de scouts (Scout Network) distinct qui effectue la détection

des frontières des mots. Ces informations sont ensuite utilisées pour guider le système ASR en ligne. Nous aimerions adapter cette idée à la tâche de traduction vocale en ligne.

En ce qui concerne les mesures de latence, comme indiqué dans Ma et al. (2021), la plupart des mesures de latence actuellement utilisées ne tiennent pas compte du temps passé à générer la traduction lors de l'analyse de la latence. Cette hypothèse faible permet à un système d'avoir un bon compromis entre latence et qualité, tout en étant inapplicable dans des scénarios en temps réel. En outre, bien que la version d'AL tenant compte des calculs prenne en considération le temps réel nécessaire pour générer chaque jeton de sortie, elle ne permet pas de comparer directement différents systèmes fonctionnant sur différentes configurations matérielles. Par conséquent, la conception d'une nouvelle mesure de latence qui reflète de manière informative les latences en temps réel encourus par les systèmes de traduction vocale en ligne et, en même temps, la prise en compte des différences en termes de configurations matérielles de ces systèmes est bien envisagée pour nos travaux futurs.

Bibliography

(2003). African Accented French. Type: dataset.

(2019). Mpf. https://hdl.handle.net/11403/mpf/v3, ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.

Agrawal, B., Müller, M., Radfar, M., Choudhary, S., Mouchtaris, A., and Kunzmann, S. (2020). Tie your embeddings down: Cross-modal latent spaces for end-to-end spoken language understanding. *arXiv preprint arXiv:2011.09044*.

Aguero, P., Adell, J., and Bonafonte, A. (2006). Prosody generation for speech-to-speech translation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE.

Alinejad, A., Siahbani, M., and Sarkar, A. (2018). Prediction improves simultaneous neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3022–3027.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR.

Ansari, E., Axelrod, A., Bach, N., Bojar, O., Cattoni, R., Dalvi, F., Durrani, N., Federico, M., Federmann, C., Gu, J., et al. (2020). Findings of the iwslt 2020 evaluation campaign. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34.

Arivazhagan, N., Cherry, C., Macherey, W., Chiu, C., Yavuz, S., Pang, R., Li, W., and Raffel, C. (2019). Monotonic infinite lookback attention for simultaneous machine translation. *CoRR*, abs/1906.05218.

Arivazhagan, N., Cherry, C., Macherey, W., and Foster, G. F. (2020a). Re-translation versus streaming for simultaneous translation. In Federico, M., Waibel, A., Knight, K., Nakamura, S., Ney, H., Niehues, J., Stüker, S., Wu, D., Mariani, J., and Yvon, F., editors, *Proceedings of the 17th International Conference on Spoken Language Translation, IWSLT 2020, Online, July 9 - 10, 2020*, pages 220–227. Association for Computational Linguistics.

Arivazhagan, N., Cherry, C., Te, I., Macherey, W., Baljekar, P., and Foster, G. (2020b). Re-translation strategies for long form, simultaneous, spoken language translation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7919–7923. IEEE.

Ataman, D., Firat, O., Gangi, M. A. D., Federico, M., and Birch, A. (2019). On the importance of word boundaries in character-level neural machine translation. In Birch, A., Finch, A. M., Hayashi, H., Konstas, I., Luong, T., Neubig, G., Oda, Y., and Sudoh, K., editors, *Proceedings of the 3rd Workshop on Neural Generation and Translation@EMNLP-IJCNLP 2019, Hong Kong, November 4, 2019*, pages 187–193. Association for Computational Linguistics.

ATILF (2020). TCOF : Traitement de corpus oraux en français. https://hdl.handle.net/11403/tcof/v2.1, ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., et al. (2021). Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*.

Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*.

Baevski, A., Auli, M., and Mohamed, A. (2019). Effectiveness of self-supervised pre-training for speech recognition. *arXiv preprint arXiv:1911.03912*.

Baevski, A., Schneider, S., and Auli, M. (2020a). vq-wav2vec: Self-supervised learning of discrete speech representations. In *International Conference on Learning Representations (ICLR)*.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020b). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33.

Bahar, P., Wilken, P., Alkhouli, T., Guta, A., Golik, P., Matusov, E., and Herold, C. (2020). Start-before-end and end-to-end: Neural speech translation by AppTek and RWTH Aachen University. In *Proceedings of the 17th*

International Conference on Spoken Language Translation, pages 44–54, Online. Association for Computational Linguistics.

Bahar, P., Zeyer, A., Schlüter, R., and Ney, H. (2019). On using specaugment for end-to-end speech translation. *arXiv preprint arXiv:1911.08876*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR 2015*, pages 3104–3112, San Diego, California, USA.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE.

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Bangalore, S., Sridhar, V. K. R., Kolan, P., Golipour, L., and Jimenez, A. (2012). Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 437–445.

Bansal, S., Kamper, H., Livescu, K., Lopez, A., and Goldwater, S. (2019). Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 58–68, Minneapolis, Minnesota. Association for Computational Linguistics.

Bapna, A., Arivazhagan, N., and Firat, O. (2019). Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*.

Beilharz, B., Sun, X., Karimova, S., and Riezler, S. (2019). Librivoxdeen: A corpus for german-to-english speech translation and german speech recognition. *arXiv preprint arXiv:1910.07924*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Bérard, A., Besacier, L., Kocabiyikoğlu, A. C., and Pietquin, O. (2018). End-to-end automatic speech translation of audiobooks. In *ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 6224–6228.

Bérard, A., Pietquin, O., Servan, C., and Besacier, L. (2016). Listen and translate: A proof of concept for end-to-end speech-to-text translation. In *NIPS Workshop on End-to-End Learning for Speech and Audio Processing*.

Bertoldi, N. and Federico, M. (2005). A new decoder for spoken language translation based on confusion networks. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 86–91. IEEE.

Boito, M. Z., Havard, W., Garnerin, M., Le Ferrand, É., and Besacier, L. (2020). MaSS: A large and clean multilingual corpus of sentence-aligned spoken utterances extracted from the Bible. In *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseille, France. European Language Resources Association.

Boito, M. Z., Havard, W. N., Garnerin, M., Ferrand, É. L., and Besacier, L. (2019). Mass: A large and clean multilingual corpus of sentence-aligned spoken utterances extracted from the bible. *arXiv preprint arXiv:1907.12895*.

Branca-Rosoff, S., Fleury, S., Lefeuvre, F., and Pires, M. (2012). Discours sur la ville. Présentation du Corpus de Français parlé Parisien des années 2000 (CFPP2000).

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Bänziger, T., Mortillaro, M., and Scherer, K. (2012). Introducing the Geneva Multimodal Expression Corpus for Experimental Research on Emotion Perception. *Emotion (Washington, D.C.)*, 12(5):1161–79.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE.

Chen, J., Ma, M., Zheng, R., and Huang, L. (2021). Direct simultaneous speech-to-text translation assisted by synchronized streaming asr. *arXiv preprint arXiv:2106.06636*.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

Cherry, C. and Foster, G. (2019). Thinking slow about latency evaluation for simultaneous machine translation. *arXiv preprint arXiv:1906.00048*.

Cherry, C., Foster, G. F., Bapna, A., Firat, O., and Macherey, W. (2018). Revisiting character-based neural machine translation with capacity and compression. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J., editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4295–4305. Association for Computational Linguistics.

Chiu, C. and Raffel, C. (2017). Monotonic chunkwise attention. *CoRR*, abs/1712.05382.

Cho, K. and Esipova, M. (2016). Can neural machine translation do simultaneous translation? *arXiv preprint arXiv:1606.02012*.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Cho, W. I., Kim, S. M., Cho, H., and Kim, N. S. (2021). Kosp2e: Korean speech to english translation corpus. *arXiv preprint arXiv:2107.02875*.

Chorowski, J. and Jaitly, N. (2016). Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Chung, Y.-A. and Glass, J. (2020a). Generative pre-training for speech with autoregressive predictive coding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3497–3501. IEEE.

Chung, Y.-A. and Glass, J. (2020b). Improved speech representations with multi-target autoregressive predictive coding. *arXiv preprint arXiv:2004.05274*.

Chung, Y.-A., Hsu, W.-N., Tang, H., and Glass, J. (2019). An unsupervised autoregressive model for speech representation learning. *arXiv preprint arXiv:1904.03240*.

Conneau, A., Baevski, A., Collobert, R., Mohamed, A., and Auli, M. (2020). Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*.

Dalvi, F., Durrani, N., Sajjad, H., and Vogel, S. (2018). Incremental decoding and training methods for simultaneous translation in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 493–499, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel, P., Arnab, G., Gilles, B., Lukas, B., and Ondrej, G. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584.

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences.

IEEE transactions on acoustics, speech, and signal processing, 28(4):357–366.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Di Gangi, M. A., Cattoni, R., Bentivogli, L., Negri, M., and Turchi, M. (2019a). Must-c: a multilingual speech translation corpus. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2012–2017. Association for Computational Linguistics.

Di Gangi, M. A., Negri, M., Cattoni, R., Dessi, R., and Turchi, M. (2019b). Enhancing transformer for end-to-end speech-to-text translation. In *Proceedings of Machine Translation Summit XVII: Research Track*, pages 21–31.

Di Gangi, M. A., Negri, M., and Turchi, M. (2019c). Adapting transformer to end-to-end spoken language translation. In *INTERSPEECH 2019*, pages 1133–1137. International Speech Communication Association (ISCA).

Dieleman, S., Oord, A. v. d., and Simonyan, K. (2018). The challenge of realistic music generation: modelling raw audio at scale. *arXiv preprint arXiv:1806.10474*.

Dong, L., Xu, S., and Xu, B. (2018). Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE.

Duong, L., Anastasopoulos, A., Chiang, D., Bird, S., and Cohn, T. (2016). An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 949–959, San Diego, California. Association for Computational Linguistics.

Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Elbayad, M. (2020). *Rethinking the Design of Sequence-to-Sequence Models for Efficient Machine Translation*. Theses, Université Grenoble Alpes [2020-....].

Elbayad, M., Besacier, L., and Verbeek, J. (2018). Pervasive attention: 2D convolutional neural networks for sequence-to-sequence prediction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 97–107, Brussels, Belgium. Association for Computational Linguistics.

Elbayad, M., Besacier, L., and Verbeek, J. (2020a). Efficient wait-k models for simultaneous machine translation. *arXiv preprint arXiv:2005.08595*.

Elbayad, M., Nguyen, H., Bougares, F., Tomashenko, N., Caubrière, A., Lecouteux, B., Estève, Y., and Besacier, L. (2020b). ON-TRAC consortium for end-to-end and simultaneous speech translation challenge tasks at IWSLT 2020. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 35–43, Online. Association for Computational Linguistics.

Elbayad, M., Ustaszewski, M., Esperança-Rodier, E., Manquat, F. B., Verbeek, J., and Besacier, L. (2020c). Online versus offline nmt quality: An in-depth analysis on english-german and german-english. *arXiv preprint arXiv:2006.00814*.

Eshkol-Taravella, I., Baude, O., Maurel, D., Hriba, L., Dugua, C., and Tellier, I. (2011). Un grand corpus oral "disponible" : le corpus d'Orléans 1968-2012. *Ressources Linguistiques Libres - Traitement Automatique des Langues*, 53(2):17–46.

Estève, Y., Bazillon, T., Antoine, J.-Y., Béchet, F., and Farinas, J. (2010). The EPAC Corpus: Manual and Automatic Annotations of Conversational Speech in French Broadcast News. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Evain, S., Nguyen, H., Le, H., Boito, M. Z., Mdhaffar, S., Alisamir, S., Tong, Z., Tomashenko, N., Dinarelli, M., Parcollet, T., et al. (2021a). Lebenchmark: A reproducible framework for assessing self-supervised representation learning from speech. *arXiv preprint arXiv:2104.11462*.

Evain, S., Nguyen, H., Le, H., Boito, M. Z., Mdhaffar, S., Alisamir, S., Tong, Z., Tomashenko, N., Dinarelli, M., Parcollet, T., et al. (2021b). Task agnostic and task specific self-supervised learning from speech with lebenchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Fitzgerald, E., Hall, K. B., and Jelinek, F. (2009). Reconstructing false start errors in spontaneous speech text.

Françoise, G. (2017). Les parlers jeunes dans l'île-de-France multiculturelle. *Paris and Gap, Ophrys*.

Fügen, C., Waibel, A., and Kolss, M. (2007). Simultaneous translation of lectures and speeches. *Machine translation*, 21(4):209–252.

Fujita, T., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2013). Simple, lexicalized choice of translation timing for simultaneous speech translation. In *INTERSPEECH*, pages 3487–3491.

Gállego, G. I., Tsiamas, I., Escolano, C., Fonollosa, J. A. R., and Costa-jussà, M. R. (2021). End-to-end speech translation with pre-trained models and adapters: UPC at IWSLT 2021. In Federico, M., Waibel, A., Costa-jussà, M. R., Niehues, J., Stüker, S., and Salesky, E., editors, *Proceedings of the 18th International Conference on Spoken Language Translation, IWSLT 2021, Bangkok, Thailand (online), August 5-6, 2021*, pages 110–119. Association for Computational Linguistics.

Garofalo, J., Graff, D., Paul, D., and Pallett, D. (2007). Csr-i (wsj0) complete. *Linguistic Data Consortium, Philadelphia*.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., and Dahlgren, N. L. (1993). DARPA TIMIT acoustic phonetic continuous speech corpus cdrom.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *Proceedings of the 34th International Conference on Machine Learning*, pages 1243–1252.

Ghahremani, P., BabaAli, B., Povey, D., Riedhammer, K., Trmal, J., and Khudanpur, S. (2014). A pitch extraction algorithm tuned for automatic speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2494–2498. IEEE.

Gournay, P., Lahaie, O., and Lefebvre, R. (2018). A canadian french emotional speech dataset. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 399–402, Amsterdam Netherlands. ACM.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee.

Grissom II, A., He, H., Boyd-Graber, J., Morgan, J., and Daumé III, H. (2014). Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*, pages 1342–1352.

Grissom II, A., Orita, N., and Boyd-Graber, J. (2016). Incremental prediction of sentence-final verbs: Humans versus machines. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 95–104.

Gu, J., Neubig, G., Cho, K., and Li, V. O. (2016). Learning to translate in real-time with neural machine translation. *arXiv preprint arXiv:1610.00388*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

Han, C., Wang, M., Ji, H., and Li, L. (2021). Learning shared semantic space for speech-to-text translation. *arXiv preprint arXiv:2105.03095*.

Han, H. J., Zaidi, M. A., Indurthi, S. R., Lakumarapu, N. K., Lee, B., and Kim, S. (2020). End-to-end simultaneous translation system for IWSLT2020 using modality agnostic meta-learning. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 62–68, Online. Association for Computational Linguistics.

He, H., Grissom II, A., Morgan, J., Boyd-Graber, J., and Daumé III, H. (2015). Syntax-based rewriting for simultaneous machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 55–64.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hernandez, F., Nguyen, V., Ghannay, S., Tomashenko, N., and Estève, Y. (2018). Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation. In *International Conference on Speech and Computer*, pages 198–208. Springer.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Hutchins, J. (2007). Machine translation: A concise history. *Computer aided translation: Theory and practice*, 13(29-70):11.

Inaguma, H., Kiyono, S., Duh, K., Karita, S., Soplin, N. E. Y., Hayashi, T., and Watanabe, S. (2020). ESPnet-ST: All-in-one speech translation toolkit. *arXiv preprint arXiv:2004.10234*.

Indurthi, S., Han, H., Lakumarapu, N. K., Lee, B., Chung, I., Kim, S., and Kim, C. (2020). End-end speech-to-text translation with modality agnostic meta-learning. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7904–7908. IEEE.

Iranzo-Sánchez, J., Silvestre-Cerdà, J. A., Jorge, J., Roselló, N., Giménez, A., Sanchis, A., Civera, J., and Juan, A. (2020). Europarl-st: A multilingual corpus for speech translation of parliamentary debates. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8229–8233. IEEE.

Jan, N., Cattoni, R., Sebastian, S., Cettolo, M., Turchi, M., and Federico, M. (2018). The iwslt 2018 evaluation campaign.

Jan, N., Cattoni, R., Sebastian, S., Negri, M., Turchi, M., Elizabeth, S., Ramon, S., Loic, B., Lucia, S., and Federico, M. (2019). The iwslt 2019 evaluation campaign. In *16th International Workshop on Spoken Language Translation 2019*.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015a). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1–10. The Association for Computer Linguistics.

Jean, S., Firat, O., Cho, K., Memisevic, R., and Bengio, Y. (2015b). Montreal neural machine translation systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Jegou, H., Douze, M., and Schmid, C. (2010). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

Jia, Y., Johnson, M., Macherey, W., Weiss, R. J., Cao, Y., Chiu, C.-C., Ari, N., Laurenzo, S., and Wu, Y. (2019). Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE.

Kahn, J., Rivière, M., Zheng, W., Kharitonov, E., Xu, Q., Mazaré, P.-E., Karadayi, J., Liptchinsky, V., Collobert, R., Fuegen, C., et al. (2020). Libri-light: A benchmark for asr with limited or no supervision. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7669–7673. IEEE.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. v. d., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Kawakami, K., Wang, L., Dyer, C., Blunsom, P., and Oord, A. v. d. (2020). Learning robust and multilingual speech representations. *arXiv preprint arXiv:2001.11128*.

Khandelwal, U., Fan, A., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2021). Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. (2015). Audio augmentation for speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Kocabiyikoglu, A. C., Besacier, L., and Kraif, O. (2018). Augmenting librispeech with French translations: A multimodal corpus for direct speech translation evaluation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. ACL.

Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.

Koehn, P. (2010). Moses, statistical machine translation system, user manual and code guide. *University of Edinburgh*.

Koehn, P. (2020). *Neural machine translation*. Cambridge University Press.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Kreutzer, J. and Sokolov, A. (2018). Learning to segment inputs for nmt favors character-level processing. In *15th International Workshop on Spoken Language Translation (IWSLT)*.

Kroll, J. and De Groot, A. (2005). Simultaneous interpreting: a cognitive perspective. In *Handbook of Bilingualism*, pages 454–479. Oxford University Press.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Lavie, A., Gates, D., Gavalda, M., Tomokiyo, L. M., Waibel, A., and Levin, L. (1996). Multi-lingual translation of spontaneously spoken language in a limited domain. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Le, H., Pino, J., Wang, C., Gu, J., Schwab, D., and Besacier, L. (2021). Lightweight adapter tuning for multilingual speech translation. *arXiv preprint arXiv:2106.01463*.

Le Moine, C. and Obin, N. (2020). Att-HACK: An Expressive Speech Database with Social Attitudes.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2.

Lee, J., Cho, K., and Hofmann, T. (2017). Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Lefèvre, F., Mostefa, D., Besacier, L., Estève, Y., Quignard, M., Camelin, N., Favre, B., Jabaian, B., and Rojas-Barahona, L. (2012). Robustesse et portabilités multilingue et multi-domaines des systèmes de compréhension de la parole : le projet PortMedia. In *Actes de la conférence conjointe JEP-TALN-RECITAL 2012*, volume 1:JEP, pages 779–786, Grenoble, France.

Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, X., Wang, C., Tang, Y., Tran, C., Tang, Y., Pino, J., Baevski, A., Conneau, A., and Auli, M. (2021). Multilingual speech translation from efficient finetuning of pretrained models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 827–838, Online. Association for Computational Linguistics.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. (2020). Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Luong, M.-T. and Manning, C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*.

Luong, T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Ma, M., Huang, L., Xiong, H., Zheng, R., Liu, K., Zheng, B., Zhang, C., He, Z., Liu, H., Li, X., Wu, H., and Wang, H. (2019). STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Ma, X., Dousti, M. J., Wang, C., Gu, J., and Pino, J. M. (2020a). SIMULEVAL: an evaluation toolkit for simultaneous translation. In Liu, Q. and Schlangen, D., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 144–150. Association for Computational Linguistics.

Ma, X., Pino, J., and Koehn, P. (2020b). Simulmt to simulst: Adapting simultaneous text translation to end-to-end simultaneous speech translation. *arXiv preprint arXiv:2011.02048*.

Ma, X., Pino, J., and Koehn, P. (2020c). SimulMT to SimulST: Adapting simultaneous text translation to end-to-end simultaneous speech translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 582–587, Suzhou, China. Association for Computational Linguistics.

Ma, X., Pino, J. M., Cross, J., Puzon, L., and Gu, J. (2020d). Monotonic multihead attention. In *International Conference on Learning Representations*.

Ma, X., Wang, Y., Dousti, M. J., Koehn, P., and Pino, J. (2021). Streaming simultaneous speech translation with augmented memory transformer. In

ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7523–7527. IEEE.

Matsubara, S., Iwashima, K., Kawaguchi, N., Toyama, K., and Inagaki, Y. (2000). Simultaneous japanese-english interpretation based on early prediction of english verb. *Proc. of SNLP*, pages 268–273.

Matusov, E., Mauser, A., and Ney, H. (2006). Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proceedings of the Third International Workshop on Spoken Language Translation: Papers*.

Matusov, E., Ney, H., and Schluter, R. (2005). Phrase-based translation of speech recognizer word lattices using loglinear model combination. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 110–115. IEEE.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. (2017). Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, volume 2017, pages 498–502.

Meignier, S. and Merlin, T. (2010). Lium spkdiarization: An open source toolkit for diarization. In *CMU SPUD Workshop*.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.

Moser-Mercer, B., Künzli, A., and Korac, M. (1998). Prolonged turns in interpreting: Effects on quality, physiological and psychological stress (pilot study). *Interpreting*, 3(1):47–64.

Nagrani, A., Chung, J. S., and Zisserman, A. (2017). VoxCeleb: a large-scale speaker identification dataset. In *Interspeech*, pages 2616–2620.

Nakamura, S. (2009). Overcoming the language barrier with speech translation technology. Technical report, Citeseer.

Nguyen, H., Bougares, F., Tomashenko, N., Estève, Y., and Besacier, L. (2020a). Investigating self-supervised pre-training for end-to-end speech translation. In *Interspeech 2020*.

Nguyen, H., Estève, Y., and Besacier, L. (2021). An empirical study of end-to-end simultaneous speech translation decoding strategies. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7528–7532. IEEE.

Nguyen, H., Tomashenko, N., Boito, M. Z., Caubriere, A., Bougares, F., Rouvier, M., Besacier, L., and Esteve, Y. (2019). ON-TRAC consortium end-to-end speech translation systems for the IWSLT 2019 shared task.

Nguyen, T. S., Niehues, J., Cho, E., Ha, T.-L., Kilgour, K., Muller, M., Sperber, M., Stueker, S., and Waibel, A. (2020b). Low latency asr for simultaneous speech translation. *arXiv preprint arXiv:2003.09891*.

Niehues, J. (2019). Survey Talk: A Survey on Speech Translation. In *Proc. Interspeech 2019*.

Niehues, J., Nguyen, T. S., Cho, E., Ha, T.-L., Kilgour, K., Müller, M., Sperber, M., Stüker, S., and Waibel, A. (2016a). Dynamic transcription for low-latency speech translation. In *Interspeech*, pages 2513–2517.

Niehues, J., Nguyen, T. S., Cho, E., Ha, T.-L., Kilgour, K., Müller, M., Sperber, M., Stüker, S., and Waibel, A. (2016b). Dynamic Transcription for Low-Latency Speech Translation. In *Proc. Interspeech 2016*, pages 2513–2517.

Niehues, J., Salesky, E., Turchi, M., and Negri, M. (2021). Tutorial proposal: End-to-end speech translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, pages 10–13, online. Association for Computational Linguistics.

Oda, Y., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2014). Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 551–556.

Oda, Y., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2015). Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 198–207.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Oord, A. v. d., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5206–5210.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 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.

Pino, J., Puzon, L., Gu, J., Ma, X., McCarthy, A. D., and Gopinath, D. (2019). Harnessing indirect training data for end-to-end automatic speech translation: Tricks of the trade. *arXiv preprint arXiv:1909.06515*.

Post, M., Kumar, G., Lopez, A., Karakos, D., Callison-Burch, C., and Khudanpur, S. (2013). Improved speech-to-text translation with the fisher and callhome spanish–english speech translation corpus. In *Proc. IWSLT*.

Potapczyk, T. and Przybysz, P. (2020). SRPOL's system for the IWSLT 2020 end-to-end speech translation task. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 89–94, Online. Association for Computational Linguistics.

Pratap, V., Xu, Q., Sriram, A., Synnaeve, G., and Collobert, R. (2020). Mls: A large-scale multilingual dataset for speech research. In *INTERSPEECH*, Shanghai, China.

Raffel, C., Luong, M.-T., Liu, P. J., Weiss, R. J., and Eck, D. (2017). Online and linear-time attention by enforcing monotonic alignments. In *International Conference on Machine Learning*, pages 2837–2846. PMLR.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., Mori, R. D., and Bengio, Y. (2021). SpeechBrain: A general-purpose speech toolkit. arXiv:2106.04624.

Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. (2020). COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Ren, Y., Liu, J., Tan, X., Zhang, C., Qin, T., Zhao, Z., and Liu, T.-Y. (2020). Simulspeech: End-to-end simultaneous speech to text translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3787–3796.

Riviere, M., Joulin, A., Mazaré, P.-E., and Dupoux, E. (2020). Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7414–7418. IEEE.

Ruiz, N., Gao, Q., Lewis, W., and Federico, M. (2015). Adapting machine translation models toward misrecognized speech with text-to-speech pronunciation rules and acoustic confusability. In *Interspeech*. ISCA-International Speech Communication Association.

Sainath, T. N., Kingsbury, B., Mohamed, A.-r., Dahl, G. E., Saon, G., Soltau, H., Beran, T., Aravkin, A. Y., and Ramabhadran, B. (2013a). Improvements to deep convolutional neural networks for lvcsr. In *2013 IEEE workshop on automatic speech recognition and understanding*, pages 315–320. IEEE.

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. (2013b). Deep convolutional neural networks for lvcsr. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8614–8618.

Salesky, E., Wiesner, M., Bremerman, J., Cattoni, R., Negri, M., Turchi, M., Oard, D. W., and Post, M. (2021). The multilingual tedx corpus for speech recognition and translation. *arXiv preprint arXiv:2102.01757*.

Sanabria, R., Caglayan, O., Palaskar, S., Elliott, D., Barrault, L., Specia, L., and Metze, F. (2018). How2: a large-scale dataset for multimodal language understanding. *arXiv preprint arXiv:1811.00347*.

Schafer, R. W. and Rabiner, L. R. (1975). Digital representations of speech signals. *Proceedings of the IEEE*, 63(4):662–677.

Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019a). wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019b). wav2vec: Unsupervised Pre-Training for Speech Recognition. In *Proc. Interspeech 2019*, pages 3465–3469.

Schultz, T., Jou, S.-C., Vogel, S., and Saleem, S. (2004). Using word latice information for a tighter coupling in speech translation systems. In *Eighth International Conference on Spoken Language Processing*.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Schwenk, H. (2012). Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of COLING 2012: Posters*, pages 1071–1080, Mumbai, India. The COLING 2012 Organizing Committee.

Schwenk, H., Déchelotte, D., and Gauvain, J.-L. (2006). Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Shapira, D. and Storer, J. A. (2002). Edit distance with move operations. In *Annual Symposium on Combinatorial Pattern Matching*, pages 85–98. Springer.

Shimizu, H., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2014). Collection of a simultaneous translation corpus for comparative analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 670–673, Reykjavik, Iceland. European Language Resources Association (ELRA).

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018). X-vectors: Robust DNN embeddings for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333.

Sperber, M., Niehues, J., and Waibel, A. (2017). Toward robust neural machine translation for noisy input sequences. In *International Workshop on Spoken Language Translation (IWSLT)*, page 18.

Sperber, M. and Paulik, M. (2020). Speech translation and the end-to-end promise: Taking stock of where we are. *arXiv preprint arXiv:2004.06358*.

Sridhar, V. K. R., Chen, J., Bangalore, S., Ljolje, A., and Chengalvarayan, R. (2013). Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238.

Stolcke, A. (2002). Srilm – an extensible language modeling toolkit. In *IN PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING (ICSLP 2002*, pages 901–904.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tan, X., Ren, Y., He, D., Qin, T., Zhao, Z., and Liu, T.-Y. (2019). Multilingual neural machine translation with knowledge distillation. *arXiv preprint arXiv:1902.10461*.

Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge university press.

Tomashenko, N., Srivastava, B. M. L., Wang, X., Vincent, E., Nautsch, A., Yamagishi, J., Evans, N., Patino, J., Bonastre, J.-F., Noé, P.-G., and Todisco, M. (2020). Introducing the VoicePrivacy initiative. In *Interspeech*.

Torreira, F., Adda-Decker, M., and Ernestus, M. (2010). The Nijmegen Corpus of Casual French. *Speech Communication*, 52(3):201. Publisher: Elsevier : North-Holland.

Tsvetkov, Y., Metze, F., and Dyer, C. (2014). Augmenting translation models with simulated acoustic confusions for improved spoken language translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 616–625, Gothenburg, Sweden. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Vauquois, B. (1968). Structures profondes et traduction automatique: le système du ceta.

Wang, C., Pino, J., Wu, A., and Gu, J. (2020a). Covost: A diverse multilingual speech-to-text translation corpus. *arXiv preprint arXiv:2002.01320*.

Wang, C., Rivière, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., and Dupoux, E. (2021). Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. *arXiv preprint arXiv:2101.00390*.

Wang, C., Wu, A., and Pino, J. (2020b). Covost 2 and massively multilingual speech-to-text translation. *arXiv preprint arXiv:2007.10310*.

Wang, C., Wu, Y., Liu, S., Li, J., Lu, L., Ye, G., and Zhou, M. (2020c). Low latency end-to-end streaming speech recognition with a scout network. In *Interspeech 2020*.

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y., Heymann, J., Wiesner, M., Chen, N., et al. (2018). Espnet: End-to-end speech processing toolkit. *arXiv preprint arXiv:1804.00015*.

Weiss, R. J., Chorowski, J., Jaitly, N., Wu, Y., and Chen, Z. (2017). Sequence-to-sequence models can directly translate foreign speech. In Lacerda, F., editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2625–2629. ISCA.

Wilken, P., Alkhouli, T., Matusov, E., and Golik, P. (2020). Neural simultaneous speech translation using alignment-based chunking. *arXiv preprint arXiv:2005.14489*.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Woszczyna, M., Coccaro, N., Eisele, A., Lavie, A., McNair, A., Polzin, T., Rogina, I., Rose, C. P., Sloboda, T., Tomita, M., Tsutsumi, J., Aoki-Waibel, N., Waibel, A., and Ward, W. (1993). Recent advances in janus: A speech translation system. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, page 211–216, USA. Association for Computational Linguistics.

Wu, C., Wang, Y., Shi, Y., Yeh, C.-F., and Zhang, F. (2020). Streaming Transformer-Based Acoustic Models Using Self-Attention with Augmented Memory. In *Proc. Interspeech 2020*, pages 2132–2136.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xiong, H., Zhang, R., Zhang, C., He, Z., Wu, H., and Wang, H. (2019). Dutongchuan: Context-aware translation model for simultaneous interpreting. *arXiv preprint arXiv:1907.12984*.

Yarmohammadi, M., Rangarajan Sridhar, V. K., Bangalore, S., and Sankaran, B. (2013). Incremental segmentation and decoding strategies for simultaneous translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1032–1036, Nagoya, Japan. Asian Federation of Natural Language Processing.

Zeng, X., Li, L., and Liu, Q. (2021). RealTranS: End-to-end simultaneous speech translation with convolutional weighted-shrinking transformer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2461–2474, Online. Association for Computational Linguistics.

Zhang, R., Kikui, G., Yamamoto, H., and Lo, W.-K. (2005). A decoding algorithm for word lattice translation in speech translation. In *Proceedings of the Second International Workshop on Spoken Language Translation*.

Zhang, R., Wang, X., Zhang, C., He, Z., Wu, H., Li, Z., Wang, H., Chen, Y., and Li, Q. (2021). Bstc: A large-scale chinese-english speech translation dataset. *arXiv preprint arXiv:2104.03575*.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhang, Y., Chan, W., and Jaitly, N. (2017). Very deep convolutional networks for end-to-end speech recognition. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4845–4849. IEEE.

Zheng, B., Liu, K., Zheng, R., Ma, M., Liu, H., and Huang, L. (2020a). Simultaneous translation policies: From fixed to adaptive. *arXiv preprint arXiv:2004.13169*.

Zheng, B., Zheng, R., Ma, M., and Huang, L. (2019a). Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Zheng, B., Zheng, R., Ma, M., and Huang, L. (2019b). Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.

Zheng, R., Ma, M., Zheng, B., and Huang, L. (2019c). Speculative beam search for simultaneous translation. *arXiv preprint arXiv:1909.05421*.

Zheng, R., Ma, M., Zheng, B., Liu, K., and Huang, L. (2020b). Opportunistic decoding with timely correction for simultaneous translation. *arXiv preprint arXiv:2005.00675*.

Zheng, R., Ma, M., Zheng, B., Liu, K., and Huang, L. (2020c). Opportunistic decoding with timely correction for simultaneous translation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 437–442. Association for Computational Linguistics.

Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. (2016). Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

Zhou, S., Dong, L., Xu, S., and Xu, B. (2018a). Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin chinese. *arXiv preprint arXiv:1804.10752*.

Zhou, S., Xu, S., and Xu, B. (2018b). Multilingual end-to-end speech recognition with a single transformer on low-resource languages. *arXiv preprint arXiv:1806.05059*.