université PARIS-SACLAY

Writing in two languages: Neural machine translation as an assistive bilingual writing tool

Ecrire en deux langues : la traduction automatique neuronale au service d'aide à la rédaction bilingue

Thèse de doctorat de l'université Paris-Saclay

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

Thèse préparée dans l'unité de recherche LISN (Université Paris-Saclay, CNRS), sous la direction de François YVON, directeur de recherche, du tuteur Josep CREGO, chercheur

Thèse soutenue à Paris-Saclay, le 02 décembre 2022, par

Jitao XU

Composition du jury

Pierre ZWEIGENBAUM Directeur de recherche, CNRS, LISN, Université Paris-Saclay	Président
Philippe LANGLAIS Professeur, Université de Montréal	Rapporteur & Examinateur
Qun LIU Directeur scientifique, HDR, Huawei Noah's Ark Lab	Rapporteur & Examinateur
Jan NIEHUES Professeur, Karlsruhe Institute of Technology	Rapporteur & Examinateur
Rachel BAWDEN Chargée de recherche, Inria Paris	Examinatrice
François YVON Directeur de recherche, CNRS, LISN, Université Paris-Saclay	Directeur de thèse

THESE DE DOCTORAT
NNT : 2022UPASG078

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Ecrire en deux langues : la traduction automatique neuronale au service d'aide à la rédaction bilingue

Mots clés: Rédaction bilingue, Aide à la rédaction, Traduction automatique neuronale, Traduction automatique non-autorégressive, Mémoire de traduction

Résumé: Dans un monde de plus en plus globalisé, il est de plus en plus courant d'avoir à s'exprimer dans une langue étrangère ou dans plusieurs langues. Cependant, pour de nombreuses personnes, parler ou écrire dans une langue étrangère n'est pas une tâche facile. Les outils de traduction automatique peuvent aider à générer des textes en plusieurs langues. Grâce aux progrès récents de la traduction automatique neuronale (NMT), les technologies de traduction fournissent en effet des traductions utilisables dans un nombre croissant de contextes. Pour autant, il n'est pas encore réaliste d'attendre des systèmes NMT qu'ils produisent des traductions sans erreur. En revanche, les utilisateurs ayant une bonne maîtrise d'une langue étrangère donnée peuvent trouver des aides auprès des technologies de traduction aidée par ordinateur.

Lorsqu'ils rencontrent des difficultés, les utilisateurs écrivant dans une langue étrangère peuvent accéder à des ressources externes telles que des dictionnaires, des terminologies ou des concordanciers bilingues. Cependant, la consultation de ces ressources provoque une interruption du processus de rédaction et déclenche une autre activité cognitive. Pour rendre le processus plus fluide, il est possible d'étendre les systèmes d'aide à la rédaction afin de prendre en charge la composition de textes bilingues. Cependant, les études existantes se sont principalement concentrées sur la génération de textes dans une langue étrangère. Nous suggérons que l'affichage de textes correspondants dans la langue maternelle de l'utilisateur peut également aider les utilisateurs à vérifier les textes composés à partir d'entrées bilingues. Dans cette thèse, nous étudions des techniques pour construire des systèmes d'aide à la rédaction bilingues qui permettent la composition libre dans les deux langues et affichent des textes monolingues synchronisés dans les deux langues. Nous présentons deux types de systèmes interactifs simulés.

La première solution permet aux utilisateurs de composer des textes dans un mélange de langues, qui sont ensuite traduits dans leurs équivalents monolingues. Nous étendons le modèle Transformer pour la traduction en ajoutant un décodeur duel: notre modèle comprend un encodeur partagé et deux décodeurs pour produire simultanément des textes en deux langues. Nous explorons également le modèle de décodeur duel pour plusieurs autres tâches, telles que la traduction multi-cible, la traduction bidirectionnelle, la génération de variantes de traduction et le sous-titrage multilingue.

La deuxième contribution vise à étendre les systèmes de traduction commerciaux disponibles en ligne en permettant aux utilisateurs d'alterner librement entre les deux langues, en changeant la boîte de saisie du texte à leur volonté. Dans ce scénario, le défi technique consiste à maintenir la synchronisation des deux textes d'entrée tout en tenant compte des entrées des utilisateurs, toujours dans le but de créer deux versions également bonnes du texte. Pour cela, nous introduisons une tâche générale de synchronisation bilingue et nous implémentons et expérimentons des systèmes de synchronisation auto-régressifs et non-autorégressifs. Nous étudions également l'utilisation de modèles de synchronisation bilingue pour d'autres tâches spécifiques, telles que le nettoyage de corpus parallèles et la NMT avec mémoire de traduction, afin de mieux évaluer la capacité de généralisation des modèles proposés.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title: Writing in two languages: Neural machine translation as an assistive bilingual writing tool
Keywords: Bilingual Writing, Assistive Writing Tools, Neural Machine Translation, Non-autoregressive Machine Translation, Translation Memory

Abstract: In an increasingly global world, more situations appear where people need to express themselves in a foreign language or multiple languages. However, for many people, writing in a foreign language is not an easy task. Machine translation tools can help generate texts in multiple languages. With the tangible progress in neural machine translation (NMT), translation technologies are delivering usable translations in a growing number of contexts. However, it is not yet realistic for NMT systems to produce error-free translations. Therefore, users with a good command of a given foreign language may find assistance from computer-aided translation technologies.

In case of difficulties, users writing in a foreign language can access external resources such as dictionaries, terminologies, or bilingual concordancers. However, consulting these resources causes an interruption in the writing process and starts another cognitive activity. To make the process smoother, it is possible to extend writing assistant systems to support bilingual text composition. However, existing studies mainly focused on generating texts in a foreign language. We suggest that showing corresponding texts in the user's mother tongue can also help users to verify the composed texts with synchronized bitexts. In this thesis, we study techniques to build bilingual writing assistant systems that allow free composition in both languages and display synchronized monolingual texts in the two languages. We introduce two types of simulated interactive systems.

The first solution allows users to compose mixed-language texts, which are then translated into their monolingual counterparts. We propose a dual decoder Transformer model comprising a shared encoder and two decoders to simultaneously produce texts in two languages. We also explore the dual decoder model for various other tasks, such as multi-target translation, bidirectional translation, generating translation variants, and multilingual subtitling.

The second design aims to extend commercial online translation systems by letting users freely alternate between the two languages, changing the texting input box at their will. In this scenario, the technical challenge is to keep the two input texts synchronized while taking the users' inputs into account, again with the goal of authoring two equally good versions of the text. For this, we introduce a general bilingual synchronization task and implement and experiment with autoregressive and non-autoregressive synchronization systems. We also investigate bilingual synchronization models on specific downstream tasks, such as parallel corpus cleaning and NMT with translation memories, to study the generalization ability of the proposed models.

Acknowledgments

Firstly and most importantly, I would like to express my deepest gratitude to my advisor François Yvon and my supervisor from SYSTRAN, Josep Crego. This thesis would not have been made possible without their endless support and help. It is one of the two most important things in my life until now to have François as my Ph.D. advisor. I have learned so much from him during the last three years. His guidance will have a big influence on my entire research career. He said he was lucky to have me, but I am more than lucky to have him. Words cannot express my appreciation for him. It is also my pleasure to meet Josep, not only as a supervisor but also as a very good friend. I will not forget our pleasant collaboration and the nice trip we had in Hong Kong.

I would also like to thank my committee members, Pierre Zweigenbaum, Philippe Langlais, Qun Liu, Jan Niehues, and Rachel Bawden, for their insights during the defense and for their detailed reports that helped me to improve my thesis.

I am also grateful to my friends at LISN, Shu Okabe, Maxime Bouthors, Alban Petit, Paul Lerner, MinhQuang Pham, François Buet, Léa-Marie Lam-Yee-Mui, Rémi Uro, Yajing Feng, and Dávid Javorský. I appreciate our daily coffee break and interesting discussions. A special thank goes to Alina Karakanta, who dragged me out of depressed emotions when I had bad experimental results. I would also like to thank my intern Ruiyang Zhou, who helped me a lot during my defense.

Moreover, I would like to thank my colleagues from SYSTRAN for always being nice to me, especially Dakun Zhang, who was very supportive when I wrote my thesis. My thank also goes to Jean Senellart, ex-CEO of SYSTRAN, for helping to create this collaborative project between LISN and SYSTRAN.

Last but not least, I would like to thank my parents and my girlfriend, Yunbei Zhang, for their support and love during the last three years. I will always love you.

This work was granted access to the HPC resources of IDRIS under the allocation 2022-[AD011011580R2] made by GENCI.

Contents

List of Acronyms

k-NN	k-Nearest Neighbor
AD	Average Edit Distance
APE	Automatic Post-Editing
AR	Autoregressive
ASR	Automatic Speech Recognition
AXE	Aligned Cross-Entropy
BERT	Bidirectional Encoder Representations from Transformers
Bi-sync	Bilingual Synchronization
BLEU	Bilingual Evaluation Understudy
BP	Brevity Penalty
BPE	Byte Pair Encoding
CAT	Computer Aided Translation
CE	Cross-Entropy
CLTE	Cross-Lingual Textual Entailment
CMLM	Conditional Masked Language Model
CNN	Convolutional Neural Network
COMET	Cross-lingual Optimized Metric for Evaluation of Translation
CTC	Connectionist Temporal Classification
FM	Fuzzy Match
GBS	Grid Beam Search
GLAT	Glancing Transformer
GLU	Gated Linear Units
IMT	Interactive Machine Translation
KD	Knowledge Distillation
L1	the native language
L2	a foreign language
L2R	Left-to-Right
LAT	Locally Autoregressive Translation
LCD	Lexically Constrained Decoding
LCS	Longest Common Subsequence
LevT	Levenshtein Transformer
LID	Language Identification
METEOR	Metric for Evaluation of Translation with Explicit ORdering
MLE	Maximum Likelihood Estimation
MLF	Matrix Language Frame
MLM	Masked Language Model
MT	Machine Translation
MXL	Mixed-Language

NAT	Non-Autoregressive Neural Machine Translation
NMT	Neural Machine Translation
OAXE	Order-Agnostic Cross-Entropy
R2L	Right-to-Left
RNN	Recurrent Neural Network
SAT	Semi-Autoregressive Translation
SMT	Statistical Machine Translation
TER	Translation Edit Rate
TF-IDF	Term Frequency-Inverse Document Frequency
TM	Translation Memory
WER	Word Error Rate

List of Figures

List of Tables

1 - Introduction

1.1 . Context and Motivation

In an increasingly global world, more and more situations appear where people need to write, speak, or, more generally, express themselves in a foreign language or multiple languages. For instance, researchers from different countries often write research articles or theses in English, research institutions often write emails in both the original language and English, countries or supranational bodies with multiple official languages compose official documents in all official languages, travelers fill out administrative documents in the language of the host country, audio-visual contents come with subtitles in multiple languages to widen the potential audience, etc. However, for many people, writing in a foreign language (L2) is not an easy task.

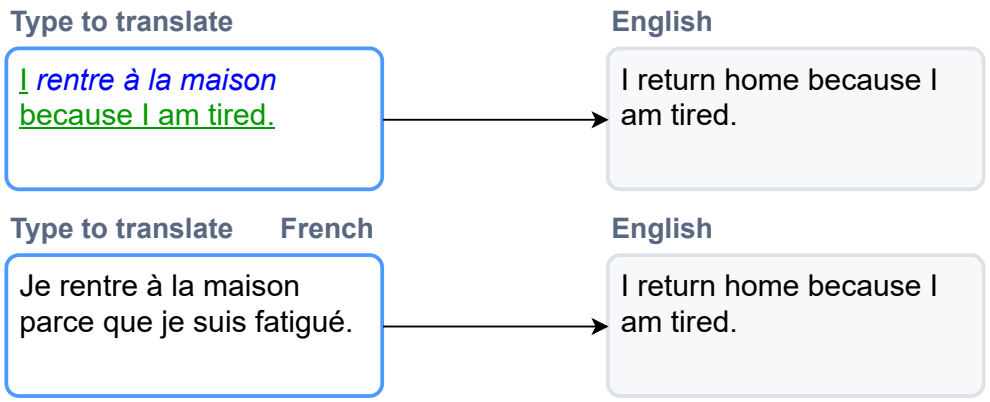
Machine translation (MT) tools, which translate texts written in one language into other languages, can help generate texts in multiple languages. With the significant advances seen in MT, especially the tangible progress in neural machine translation (NMT) in recent years, MT systems are delivering usable translations in a growing number of contexts. Nevertheless, relying on this technology to produce high-quality documents is not yet realistic since the current state-of-the-art NMT systems have not reached the level where systems can produce error-free translations, and also because fully automatic translation does not provide the necessary control over the produced translation. Therefore, users with a good command of L2 but not at a professional level may find assistance from various Computer Assisted Language Learning tools or Computer Aided Translation (CAT) systems. Users can access external resources such as dictionaries, terminologies, or bilingual concordancers (Bourdaillet et al., 2011) to help with writing. To this end, several studies have focused on developing systems to help users compose text in L2 (Koehn, 2010; Huang et al., 2012; Venkatapathy and Mirkin, 2012; Chen et al., 2012; Liu et al., 2016b).

Nevertheless, consulting external resources causes an interruption in the writing process and starts another cognitive activity, even when writing in the users' native language (L1) (Leijten et al., 2014). Besides, L2 users tend to rely on L1 (Wolfersberger, 2003) to prevent a breakdown in the writing process (Cumming, 1989). Therefore, it has been proposed to integrate bilingual text composition into writing tools, where input in L2 can guide the writing process. For instance, the system presented by Chen et al. (2012) can be illustrated in the following example. A French writer composing the English sentence [*I* ***return home*** *because I am tired.*] may be unsure about how to formulate the notion of "return home." An input in French,

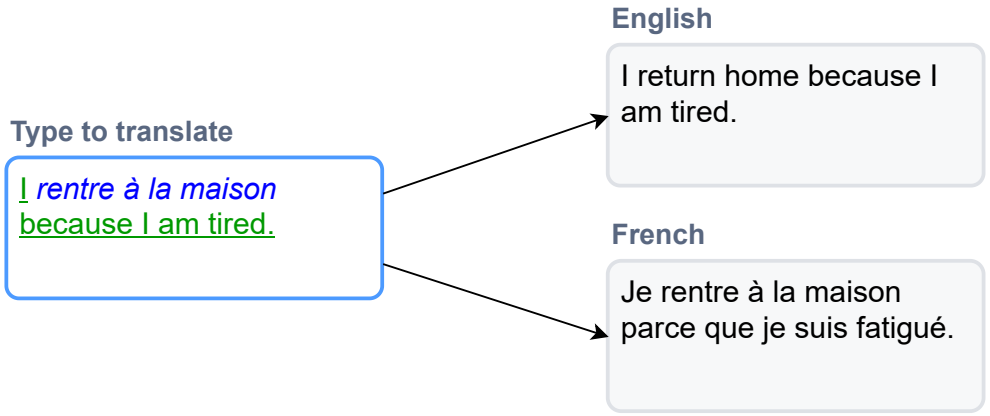
such as [*I* ***rentre à la maison*** *because I am tired.*], can be used by the system to propose alternative formulations in English. This is certainly a more acceptable solution than using fully automatic translation. First, the user actively participates in the writing process, using L2 as much as possible, without needing heuristic searches for external sources that interrupt the writing process. Second, the existing text fragments in L2 that the user is able to produce with high confidence can be used as valuable contextual clues to improve the system's suggestions. The shared task on L2 writing assistant that translates L1 fragments in an L2 context, proposed by van Gompel et al. (2014), attempts to simulate this scenario and evaluate its difficulty.

However, existing studies mainly focused on generating texts in L2, leaving the users to decide whether the provided texts precisely conveyed what they wanted to express, while it is not easy to evaluate L2 texts for users who are not at a professional level. On the other hand, researchers have explored using round-trip translation that translates the MT output written in L2 back to L1 to evaluate the quality of translation (Moon et al., 2020). Therefore, we suggest that it is also helpful to show the corresponding texts in L1 to help users verify the composed texts with synchronized bitext. Taking the previous example, when a user composes [*I* ***rentre à la maison*** *because I am tired.*], he/she can verify the French sentence [*Je rentre à la maison parce que je suis fatigué.*] displayed together with the English translation [*I return home because I am tired.*] to make sure that the English sentence is precise. Furthermore, users can obtain synchronized texts in two languages while only making an effort to compose approximately one sentence. Such systems should allow users to write freely in both languages and always provide synchronized monolingual texts in the two languages. However, existing systems do not support both functionalities at the same time. Systems like Chen et al. (2012) allow free writing in both languages but only show the final texts in L2, as illustrated in Figure 1.1a (top). Commercial translation systems like Google, DeepL, SYSTRAN, etc., always display texts in both languages. However, users can only edit the source side while the target side is computed by the systems and cannot be changed freely, as shown in Figure 1.1a (bottom). CAT systems assume the given source sentence to be fixed and only allow edits on the target side.

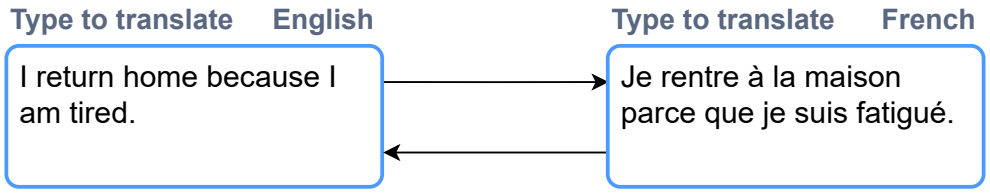
In this thesis, we study techniques to build bilingual writing assistant systems that allow free composition in both languages and simultaneously display synchronized monolingual texts in the two languages. We propose two types of simulated interactive systems as possible solutions. The first design contains three text boxes. Users can compose texts with a free mix of both languages in only one box. The other two boxes display the corresponding texts in each language generated by the system. An illustration



(a) System of Chen et al. (2012) (top) and commercial translation systems (bottom).



(b) A three-box design allowing free bilingual composition in the left box and displaying monolingual texts on the right.



(c) A two-box design allowing bilingual composition by changing input boxes. Users can freely choose the language to write, and the system automatically updates the texts in the other box.

Figure 1.1: Multiple designs of bilingual writing systems. Texts mixing two languages are distinguished using different colors and patterns.

of this design is shown in Figure 1.1b. Such interactions require the system to simultaneously translate a mixed-language sentence into both languages. The second idea extends commercial translation systems with only two text boxes, as illustrated in Figure 1.1c. Each box shows texts in one language,

and users can freely choose to write in either box at their will. Once texts in one box are changed, the system automatically updates texts in the other language so that texts in the two boxes always remain synchronized. This system requires a bilingual synchronization ability that does not specify a particular source or target language but always keeps the two texts as translations of each other while taking the users' input into account.

This thesis targets the study of new methods for generating high-quality synchronized texts in two languages in the two simulated situations described above using NMT techniques. The developed methods can be further integrated into well-designed interactive systems to build real bilingual writing assistant tools. We have not explored the interactive systems in real-world scenarios in this thesis but aim to explore this more in the future.

1.2 . Contributions

Our contributions of this thesis are as follows:

- We study bilingual writing assistance in two different scenarios. The first situation reads sentences written in a mixed language and generates monolingual sentences in both languages, while the second solution aims to synchronize texts in the two languages when either language is changed.
- We propose a dual decoder NMT model which is able to simultaneously disentangle a mixed-language sentence into both component languages, and we conduct thorough analyses of the task of translating mixed-language sentences.
- We apply the dual decoder model to various other tasks, like multi-target translation, bidirectional decoding, multi-style decoding, and multilingual subtitling, to evaluate the challenges and rewards of dual decoding.
- We propose a general bilingual synchronization task that aims to synchronize text in one language to another so that both texts are always parallel to each other. We also propose novel non-autoregressive architectures to perform this task.
- We study ways to generate various types of synthetic data that can simulate user-generated texts during bilingual writing and show that the generated artificial data is effective in training the corresponding models.
- We also apply bilingual synchronization models to more specific tasks, like parallel corpus cleaning and translation with translation memories, to study the adaptation capacity of our proposed models.

1.3 . Outline

This thesis is organized as follows:

- **Chapter 2** gives an introduction to conventional NMT systems. We discuss text tokenization methods, classical NMT model architectures used in this thesis, training and inference procedures, and evaluation metrics.
- **Chapter 3** reviews several NMT tasks incorporating augmented sources into the translation process. Conventional NMT produces the translation solely based on the source sentence, while these tasks all use some information other than the source when producing the translation. This is also the case in our bilingual writing scenario, where segments composed in a bilingual way contain information from both languages. We focus on interactive MT, automatic post-editing, NMT using translation memories, incorporating lexical constraints in NMT, and multi-source/multi-target NMT.
- In **Chapter 4**, we introduce non-autoregressive NMT, an emerging subdomain of NMT which aims to generate all target words in one single step instead of one by one from left to right. The key advantage of non-autoregressive NMT is the speedup in translation time compared to conventional autoregressive methods, which can be a good fit for real-world interactive scenarios, where the translation latency needs to be very low so that the models can quickly update translations following users' input.
- In **Chapter 5**, we study the task of simultaneously translating mixed-language sentences into both component languages. We propose a novel dual decoder Transformer model and a mixed-language data generation method for this task.
- **Chapter 6** applies our proposed dual decoder model to various tasks, including multi-target translation, bidirectional decoding, multi-style decoding, and multilingual subtitling.
- **Chapter 7** describes a new framework targeting the bilingual synchronization task, which generalizes MT. We propose autoregressive approaches to perform this task. We study various methods to generate synthetic training data in order to simulate this task. We also adapt models performing generic bilingual synchronization to two specific downstream tasks: NMT with translation memories and parallel corpus cleaning.
- **Chapter 8** focuses on studying bilingual synchronization with non-autoregressive methods. We also apply the non-autoregressive generic bilingual synchronization model to the NMT with translation memories task. We conduct an in-depth study of non-autoregressive NMT with

translation memories, which has not been explored in the literature.
- Finally, **Chapter 9** concludes this thesis with a summarization of our contributions and an outlook on future research directions.

1.4 . List of Publications

This thesis draws contributions from some of the articles published as follows:

- Jitao Xu, Josep Crego, and François Yvon. 2022b. Bilingual synchronization: Restoring translational relationships with editing operations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8016–8030, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics
- Jitao Xu, Josep Crego, and François Yvon. 2022c. Non-autoregressive machine translation with translation memories. *CoRR*, abs/2210.06020
- Jitao Xu, François Buet, Josep Crego, Elise Bertin-Lemée, and François Yvon. 2022a. Joint generation of captions and subtitles with dual decoding. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 74–82, Dublin, Ireland (in-person and online). Association for Computational Linguistics
- Jitao Xu and François Yvon. 2021b. One source, two targets: Challenges and rewards of dual decoding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8533–8546, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics
- Jitao Xu and François Yvon. 2021a. Can you traducir this? Machine translation for code-switched input. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 84–94, Online. Association for Computational Linguistics
- Minh Quang Pham, Jitao Xu, Josep Crego, François Yvon, and Jean Senellart. 2020. Priming neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 462–473, Online. Association for Computational Linguistics
- Jitao Xu, Josep Crego, and Jean Senellart. 2020. Boosting neural machine translation with similar translations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1580–1590, Online. Association for Computational Linguistics
- Jitao Xu, Minh Quang Pham, Sadaf Abdul Rauf, and François Yvon. 2021a. LISN @ WMT 2021. In *Proceedings of the Sixth Conference on Machine Translation*, pages 232–242, Online. Association for Computational Linguistics

2 - Neural Machine Translation

Neural machine translation (NMT) is currently the most widely used technique in the area of machine translation (MT). Thanks to the introduction of the first encoder-decoder architecture (Allen, 1987; Pollack, 1990; Sutskever et al., 2014; Cho et al., 2014a), then completed with an attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017), the performance of NMT systems is now good enough for a growing number of services, both for the general public and the translation industry. Given a source sentence $\mathbf{f}$ and the corresponding target sentence $\mathbf{e}$, NMT systems model the conditional probability $P(\mathbf{e}|\mathbf{f})$ of $\mathbf{e}$ given $\mathbf{f}$, with $\mathbf{f} = f_1, f_2, \ldots, f_S$ and $\mathbf{e} = e_1, e_2, \ldots, e_T$ two sequences of discrete units. This chapter provides a brief background introduction to the conventional NMT systems used in this thesis. A more detailed review of NMT can be found in (Stahlberg, 2019).

2.1 . Vocabulary and Byte Pair Encoding

2.1.1 . The Open Vocabulary Problem

When processing texts with NMT models, raw texts need to be firstly tokenized into unique tokens, which then construct the vocabularies of the source and target languages. Tokens may contain words, subwords, characters, numbers, and punctuations. NMT models generally operate with a fixed vocabulary, while MT is an open-vocabulary problem. Due to computational constraints, it is impossible to include every word in a language in the vocabulary, especially for morphologically rich languages. Early approaches (Cho et al., 2014a; Bahdanau et al., 2015) only consider the N most frequent words of each language in the vocabulary and treat all rare and unknown words as a special token <unk>. As a result, unknown words become a major challenge for word-based NMT models since the translation quality degrades vastly as the number of unknown words increases (Cho et al., 2014a).

In order to deal with the open vocabulary problem, some studies propose to perform character-based NMT (Chung et al., 2016; Luong and Manning, 2016; Costa-jussà and Fonollosa, 2016; Lee et al., 2017; Cherry et al., 2018), which considers characters instead of words as unique tokens. In this way, all words can be decomposed into sequences of characters, and unknown words only happen when an unknown character appears in the test sentence, which is relatively rare. Therefore, character-based models can handle rare words from particular domains better as they will not be considered as un-

known words. These approaches successfully reduce the vocabulary size of each language to the number of unique characters that occur in the language. However, one downside is that character-based approaches have to deal with longer sequences and largely increase the translation time, as each word is separated into a character sequence. Recent work (Libovický et al., 2022) finds that character-based models do not perform better on domain robustness or morphological generalization, despite often having this as a motivation. They only show robustness to source input noise. Besides, the segmentation choice has different effects according to different languages. Ideographic languages like Chinese and Japanese have different segmentation effects compared to alphabetic languages like English and French (Kreutzer and Sokolov, 2018; Zhang and Komachi, 2018). Languages with different morphological categories also perform differently across segmentation methods (Zhang and Komachi, 2018).

2.1.2 . Byte Pair Encoding

Both word-based and character-based NMT models have significant flaws, which has led researchers to propose an intermediary solution based on subwords. In such systems, words are segmented into subword sequences based on their occurrence frequency in the training corpus. Frequent words are often left unchanged, while rare words are generally split into a sequence of subword units. Sennrich et al. (2016c) introduce subword-level tokenization for NMT using the Byte Pair Encoding (BPE, Gage, 1994) algorithm. Each word in a training corpus is first split into a sequence of characters plus a special end-of-word symbol to mark word boundaries. All existing unique characters are considered as the initial subword unit set, also called the vocabulary of subwords. At each iteration, the most frequent pair of units that appears in the training corpus is merged into a new unit, which is then added to the vocabulary. The merge operation is recorded and will be used to apply BPE tokenization. The training corpus is also updated according to the merge operation. This merging step is performed iteratively until a predefined maximum number of merge operations is achieved. The frequency count of unit pairs does not include pairs that cross word boundaries, assuring that the merge operations only happen within original words. The final BPE model thus consists of an ordered set of unique merge operations.

BPE tokenization first splits all words into sequences of individual characters, then iteratively applies the merge operations according to their appearing order in the trained BPE model. Rare words that are not merged into their original form remain as a sequence of subword units or a sequence of characters in some extreme cases. For instance, in the WMT14 English-French corpus that is used in Chapters 5, 6, and 7, the word attribution is a common word in a training corpus, while Attribution and attribute are

relatively rarer. As a result, attribution remains unsegmented after applying BPE tokenization, while Attribution and attribute are split into [Attribu@@ tion] and [attribu@@ te], respectively, with "@@" as a special joiner to mark a split in a word.

BPE tokenization thus handles the open-vocabulary problem better by using a vocabulary built with common words, subwords, and characters. NMT systems using BPE can operate with a vocabulary size of about $30-50$k and almost covers all words, even though unknown words may still appear in some extreme cases where new characters are introduced.

BPE operations can be learned separately or jointly for the source and target language. Sennrich et al. (2016c) demonstrate that languages written in the same alphabet can benefit from a joint BPE vocabulary to increase the consistency between source and target language. Joint BPE also gives the possibility to use a shared source-target vocabulary which is necessary for parameter sharing of the word embedding matrices in NMT models (Press and Wolf, 2017; Inan et al., 2017), which we discuss in Section 2.2.1.

2.1.3 . Other Methods

As the merge operations of BPE are fixed once constructed, the resulting segmentations for given words are unique, despite multiple segmentations being possible. Provilkov et al. (2020) propose BPE-dropout, which applies stochastic corruptions in the segmentation procedure of BPE by randomly dropping merges at each merging step. Therefore, the resulting segmentations thus vary for given words, helping NMT models to learn the compositionality of words better and to be robust to segmentation errors. Kudo (2018) introduce sentence-piece tokenization, which models the segmentation of a complete sentence rather than separate words with a unigram language model. It can yield different segmentations for the same sentence by performing random subword sampling based on the unigram language model.

2.2 . Neural Machine Translation Models

State-of-the-art NMT systems generate translations in an autoregressive way, in which the probability of generating a target token is conditioned on the previously generated target tokens and the source sentence. Therefore, the probability $P(\mathbf{e}|\mathbf{f})$ is decomposed as a chain rule:

$$P(\mathbf{e}|\mathbf{f}) = \prod_{i=1}^{T} P(e_i|\mathbf{e}_{<i}, \mathbf{f}; \theta), \tag{2.1}$$

where T is the length of the target sequence, and θ represents the model parameters.

Most NMT systems follow an end-to-end encoder-decoder architecture. The encoder encodes the source sequence into high dimension vector representations. The decoder takes the source representations and the representations of previously generated target tokens computed by itself as input to generate new tokens. At each time step i, the decoder computes a probability distribution over the output target vocabulary by mapping the hidden state $\mathbf{s}_i$ for time step i to a vector space $\mathbb{R}^{|V|}$ where $|V|$ is the size of the target vocabulary:

$$P(\cdot|\mathbf{e}_{<i}, \mathbf{f}) = \text{softmax}(\mathbf{s}_i^T \mathbf{W}_p), \tag{2.2}$$

where $\mathbf{W}_p$ is a linear transformation that maps $\mathbf{s}_i$ to $\mathbb{R}^{|V|}$. The softmax function normalizes the output vector to a probability distribution, in which the probability of each element n is:

$$\text{softmax}(\mathbf{z})_n = \frac{e^{z_n}}{\sum_{j=1}^{J} e^{z_j}} \quad \text{for } n = 1, \ldots, J, \tag{2.3}$$

where J is the dimension of the vector. The generation often selects the token with the maximum probability as the predicted token at time step i.

With the advances in the field of NMT in recent years, NMT model architectures have evolved from approaches based on Recurrent Neural Network (RNN, Cho et al., 2014a; Bahdanau et al., 2015) to approaches based on Convolutional Neural Network (CNN, Gehring et al., 2017) and achieved the current state-of-the-art Transformer-based approaches (Vaswani et al., 2017). This section mainly describes the Transformer architecture, which is used for all models developed in this thesis. RNN and CNN-based approaches are also briefly introduced.

2.2.1 . Transformer Model

The Transformer model is a sequence-to-sequence model based on the self-attention mechanism proposed by Vaswani et al. (2017). It is first introduced for the NMT task and further explored as the state-of-the-art backbone architecture for large pre-trained language models like BERT (Devlin et al., 2019), XLM (Conneau and Lample, 2019), GPT-3 (Brown et al., 2020), etc. The Transformer model also follows the encoder-decoder architecture as discussed below.

Transformer Encoder

The Transformer encoder contains several identical encoder layers. Each layer consists of a multi-head self-attention sub-layer followed by a position-wise fully connected feed-forward sub-layer. The attention function maps

a query and a set of key-value pairs to an output, where the queries, keys, values, and outputs are all vector representations. By packing a set of queries, keys, and values into matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively, the attention mechanism is computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{2.4}$$

where d_k is the dimension of query and key vectors. In practice, the dimension of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are always set identical to d_k. In a self-attention sub-layer, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are identical, and all come from the output of the previous layer. The multi-head attention mechanism linearly projects $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ into different subspaces, performs attention separately in parallel, then concatenates the outputs from all subspaces back to dimension d_k. More precisely, it is computed as:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\mathbf{head}_1, \ldots, \mathbf{head}_h)\mathbf{W}^O \\ \mathbf{head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \end{aligned} \tag{2.5}$$

where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d_h \times d_k}$, $\mathbf{W}^O \in \mathbb{R}^{hd_h \times d_k}$, $h \times d_h = d_k$, with h as the number of heads, and d_h the dimension of each projected vector. The feed-forward sub-layer consists of a single hidden layer network with a ReLU activation, applied to each position separately and identically:

$$\text{FFN}(\mathbf{x}) = \text{ReLU}(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{2.6}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_k \times d_{hidden}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{hidden} \times d_k}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{hidden}}$ and $\mathbf{b}_2 \in \mathbb{R}^{d_k}$.

A residual connection (He et al., 2016) is applied to each sub-layer, followed by layer normalization (Ba et al., 2016). These two operations together are referred to as AddNorm:

$$\text{AddNorm}(\mathbf{x}) = \text{LayerNorm}(\mathbf{x} + \text{SubLayer}(\mathbf{x})), \tag{2.7}$$

with $\text{SubLayer}(\mathbf{x})$ applying a sub-layer function (MultiHead or FFN) to $\mathbf{x}$.

Transformer Decoder

The Transformer decoder also contains several identical decoder layers. Different from the encoder layer, each decoder layer consists of three sub-layers: a decoder self-attention sub-layer, an encoder-decoder cross-attention sub-layer, and a feed-forward sub-layer. In order to preserve the autoregressive property, the decoder is prevented from seeing information from future positions by applying a causal mask to the decoder self-attention sub-layer. For each position i, attention scores for all positions greater than i, in the

input of the softmax function in Equation (2.4) in the decoder self-attention layer, are masked out (setting to $-\infty$). The encoder-decoder cross-attention layer takes the output of the last encoder layer as keys and values, performing attention as $\mathrm{MultiHead}(\mathbf{Q}, \mathbf{H}, \mathbf{H})$ with $\mathbf{H}$ as the encoder output and $\mathbf{Q}$ as the output of the previous decoder self-attention sub-layer. The feed-forward sub-layer performs similarly to the encoder layers, and each sub-layer is also surrounded by an $\mathrm{AddNorm}$ operation. The final decoder layer output states are passed to a linear output transformation to compute the output probability distributions.

Embeddings

In NMT models, a token sequence is represented as a sequence of integers. Each integer refers to the index of the corresponding token in the vocabulary. Both encoder and decoder require an embedding matrix, which transforms a token integer into a high dimension real-valued vector. This vector, also called word embedding, possesses semantic properties (Mikolov et al., 2013a,b). The embedding matrix $\mathbf{E}$ contains a vector for each token in the vocabulary and is of size $|V| \times d$ with $|V|$ as the corresponding vocabulary size and d as the dimension of the embedding vector. In sequence transduction models, the embedding matrices are often randomly initialized and learned together with other parameters so that the learned embeddings fits the specific task better. When the vocabularies of the source and target language are identical, which is often the case when performing a joint BPE tokenization, the parameters of the embedding matrices $\mathbf{E}_{l_S}$ and $\mathbf{E}_{l_T}$, together with the linear output matrix $\mathbf{W}_p$ can be shared (Press and Wolf, 2017; Inan et al., 2017) without losing translation quality.

Positional Encoding

As the computations of attention and feed-forward layers are independent and identical across all positions, the Transformer model does not include the order information of sequences and considers the distance between every two positions as identical. In order to inject positional information of each token into the sequences, Vaswani et al. (2017) propose to use an absolute positional encoding, which is added element-wise to the word embedding vector of each token and is defined as:

$$\begin{aligned} \mathrm{PE}(pos, 2i) &= \sin\left(\frac{pos}{10000^{2i/d_k}}\right) \\ \mathrm{PE}(pos, 2i+1) &= \cos\left(\frac{pos}{10000^{2i/d_k}}\right), \end{aligned} \tag{2.8}$$

where pos is the position in the input sequence, and i is the dimension in a vector of dimension d_k. The resulting embedding vectors are taken as input

by the first encoder or decoder layer.

2.2.2 . RNN-based Models

Early architectures before the Transformer model mainly follow the RNN encoder-decoder architecture with attention proposed by Bahdanau et al. (2015). The encoder contains a bidirectional RNN, consisting of a forward and a backward RNN, to encode information not only from preceding tokens but also from future tokens. The forward RNN reads the input sequence from left to right and computes a sequence of forward hidden states $(\overrightarrow{\mathbf{h}}_1, \ldots, \overrightarrow{\mathbf{h}}_I)$, while the backward RNN reads the input from right to left, generating a sequence of backward hidden states $(\overleftarrow{\mathbf{h}}_1, \ldots, \overleftarrow{\mathbf{h}}_I)$. The representation of a token at position j of the bidirectional RNN is the concatenation of forward and backward hidden states at this position: $\mathbf{h}_j = [\overrightarrow{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]$, where $[\cdot;\cdot]$ denotes the concatenation operation. The RNN module can also be replaced by Long Short Term Memory units (Hochreiter and Schmidhuber, 1997) as in (Wu et al., 2016), or by Gated Recurrent Unit (Cho et al., 2014b).

The RNN decoder generates target tokens from left to right, following the autoregressive property. Therefore, the RNN decoder does not use a bidirectional module. At each time step i, the hidden state of the decoder is computed as:

$$\mathbf{s}_i = g(e_{i-1}, \mathbf{s}_{i-1}, \mathbf{c}_i), \tag{2.9}$$

where $\mathbf{c}_i$ is a context vector for time step i. It is computed as a weighted sum of the encoder hidden states $\mathbf{h}_j$:

$$\mathbf{c}_i = \sum_{j=1}^{S} \alpha_{ij} \mathbf{h}_j, \tag{2.10}$$

where α_{ij} is the attention weight describing how well the input tokens around position j match the output at position i. The attention weight α_{ij} of $\mathbf{h}_j$ is computed by:

$$\begin{aligned} \epsilon_{ij} &= f_{\text{attn}}(\mathbf{s}_{i-1}, \mathbf{h}_j) \\ \alpha_{ij} &= \frac{\exp(\epsilon_{ij})}{\sum_{k=1}^{S} \exp(\epsilon_{ik})}, \end{aligned} \tag{2.11}$$

where f_{attn} is an alignment model which computes the soft alignment between the source and target token representations. The decoder output state $\mathbf{s}_i$ is used to compute the output probability distribution. The alignment model is defined as a feed-forward network as in (Bahdanau et al., 2015) and jointly trained with the other components of the RNN model. Luong et al. (2015) find that simply using dot-product as f_{attn} also yielded good performance.

2.2.3 . CNN-based Models

The CNN-based model (Gehring et al., 2017) uses a stacked convolution layer with gated linear units (GLU, Dauphin et al., 2017) to compute the hidden states of the source and target sentences. For each convolutional layer, the output hidden state $\mathbf{h}_i^l$ at position i of layer l contains information about k input elements around i. The convolution kernel is computed as:

$$Y = W^l[\mathbf{h}_{i-k/2}^{l-1}; \ldots; \mathbf{h}_{i+k/2}^{l-1}] + \mathbf{b}^l, \tag{2.12}$$

which maps k concatenated input states $[\mathbf{h}_{i-k/2}^{l-1}; \ldots; \mathbf{h}_{i+k/2}^{l-1}] \in \mathbb{R}^{kd}$ into an intermediate vector $Y \in \mathbb{R}^{2d}$ with parameter $W^l \in \mathbb{R}^{2d \times kd}$ and $\mathbf{b}^l \in \mathbb{R}^{2d}$. The intermediate vector is separated into two vectors $A, B \in \mathbb{R}^d$ where $Y = [A; B]$, and then fed into the GLU for non-linearity:

$$v([A; B]) = A \otimes \sigma(B), \tag{2.13}$$

where $\otimes$ is the element-wise multiplication and σ is the sigmoid function. Residual connections also apply to each layer so that the actual output state at position i is:

$$\mathbf{h}_i^l = v(W^l[\mathbf{h}_{i-k/2}^{l-1}; \ldots; \mathbf{h}_{i+k/2}^{l-1}] + \mathbf{b}^l) + \mathbf{h}_i^{l-1}. \tag{2.14}$$

For the decoder, the convolution kernel is shifted by $k/2$ to the left to prevent the decoder from seeing future information.

$$\mathbf{g}_i^l = v(W^l[\mathbf{g}_{i-k+1}^{l-1}; \ldots; \mathbf{g}_i^{l-1}] + \mathbf{b}^l) + \mathbf{g}_i^{l-1}. \tag{2.15}$$

Note that the parameters W^l and $\mathbf{b}^l$ are different from the encoder parameters.

Gehring et al. (2017) also use an attention mechanism in every decoder layer similar to the Transformer. They combine the current decoder state $\mathbf{g}_i^l$ with the embedding vector $\mathbf{g}_i^0$:

$$\mathbf{d}_i^l = W_d^l \mathbf{g}_i^l + \mathbf{b}_d^l + \mathbf{g}_i^0, \tag{2.16}$$

and apply dot-product to compute the attention weight between $\mathbf{d}_i^l$ and the encoder output state $\mathbf{h}_j^u$ as:

$$a_{ij}^l = \frac{\exp(\mathbf{d}_i^l \cdot \mathbf{h}_j^u)}{\sum_{t=1}^S \exp(\mathbf{d}_i^l \cdot \mathbf{h}_t^u)}. \tag{2.17}$$

The context vector $\mathbf{c}_i^l$ is computed as the weighted sum of both the encoder states and the embeddings for all positions:

$$\mathbf{c}_i^l = \sum_{j=1}^S a_{ij}^l (\mathbf{h}_j^u + \mathbf{h}_j^0), \tag{2.18}$$

which is directly added to the decoder state $\mathbf{g}_i^l$ to make the actual output state of the attention mechanism $\mathbf{s}_i^l = \mathbf{g}_i^l + \mathbf{c}_i^l$. The final probability distribution is computed similarly to the Transformer and RNN models.

The CNN model is similar to Transformer as it cannot capture the distance information very well and requires a positional embedding to be added to the word embedding vector.

2.3 . Training Neural Machine Translation Models

NMT models can be trained with Maximum Likelihood Estimation (MLE). Given a training set $D = \{(\mathbf{f}, \mathbf{e})_i, i = 1, \ldots, N\}$, the set of all parameters θ of an NMT model is estimated by solving the following optimization problem:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \log P(\mathbf{e}^{(i)}|\mathbf{f}^{(i)}; \theta), \tag{2.19}$$

which is equivalent to minimizing the cross-entropy (CE) loss:

$$\theta^* = \arg\min_{\theta} - \sum_{n=1}^{N} \sum_{i=1}^{T} \log P(e_i^{(n)}|\mathbf{e}_{<i}^{(n)}, \mathbf{f}^{(n)}; \theta). \tag{2.20}$$

For each training example, the CE loss in formally defined as:

$$\mathcal{L}_{CE} = -\sum_{i=1}^{T} \sum_{k \in V} \log(p(k|\mathbf{e}_{<i}, \mathbf{f})) \cdot q(k|\mathbf{f}), \tag{2.21}$$

where the ground truth label distribution $q(k|\mathbf{f}) = \delta_{k,e}$ is a Dirac delta function, with $q(k = e|\mathbf{f}) = 1$ and $q(k|\mathbf{f}) = 0$ for all $k \neq e$. However, training directly on the standard CE loss may result in over-fitting, as the model becomes too confident and learns to give full probability to the ground truth label. To alleviate this problem, Szegedy et al. (2016) propose the *label smoothing* strategy, which uniformly assigns small probabilities to tokens other than the ground truth label to make the model less confident. With a smoothing parameter ϵ, the ground truth distribution is modified as:

$$q'(k|\mathbf{f}) = (1 - \epsilon) \cdot \delta_{k,e} + \frac{\epsilon}{|V|}. \tag{2.22}$$

The label-smoothed CE loss is thus defined as:

$$\mathcal{L}_{LSCE} = -\sum_{i=1}^{T} \left[(1 - \epsilon) \cdot \log P(e_i|\mathbf{e}_{<i}, \mathbf{f}) + \frac{\epsilon}{|V|} \cdot \sum_{k \in V} \log P(k|\mathbf{e}_{<i}, \mathbf{f}) \right]. \tag{2.23}$$

In practice, label smoothing is often applied with $\epsilon = 0.1$.

The parameters can be optimized using the gradient descent method with backpropagation (Rumelhart et al., 1986). As NMT models are often very large and require a great amount of parallel data to train, gradient descent is thus computationally infeasible. The common practice uses stochastic gradient descent with mini-batches. The most widely used gradient-based optimization algorithm is Adam (Kingma and Ba, 2015), which is based on adaptive estimates of lower-order moments of the gradients.

Standard NMT models only perform bilingual translation from one source language into one target language, with the training dataset D containing only one pair of languages. Thanks to the sequence-to-sequence architecture of NMT models, the dataset D can be easily extended to have multiple language pairs without significant modifications to the model architecture, training one single NMT model on several language pairs to perform multilingual translation. The encoder side of a multilingual NMT model often remains similar to a bilingual model. However, the multilingual decoder requires an extra signal to specify the desired target language. The most commonly used solution is to prepend an extra token tag indicating the output language either to the source or target sentence during training (Johnson et al., 2017). This tagging mechanism is very simple yet effective. It can provide additional information to the model from various aspects, such as distinguishing real and synthetic parallel training data (Caswell et al., 2019; Marie et al., 2020), controlling translation style (Sennrich et al., 2016a), providing specific domain indications (Kobus et al., 2017), etc.

2.3.1 . Teacher Forcing and Scheduled Sampling

During training, the ground truth prefix $\mathbf{e}_{<i}$ is used for every time step i to compute the output probability distribution. NMT models often do not make predictions at the training stage. The distribution vector is only used to compute the CE loss. This procedure is also called *teacher forcing*. For the Transformer decoder, the training through time steps can be parallelized by preparing the prefix $\mathbf{e}_{<i}$ for each time step and batch all steps into one matrix. This is implemented by adding a begin-of-sentence token to the beginning of each sentence, thus shifting the reference target sequence by 1 step to the right so that each position can only attend to preceding tokens thanks to the decoder self-attention causal mask described in Section 2.2.1. However, NMT models generate tokens from scratch during inference and can only condition on their own previous predictions, which can be possibly incorrect. NMT models thus need to generate words based on noisy prefixes containing previous decoding errors (also known as cascading errors). Therefore, errors may accumulate as the decoding continues. This issue is known as the *exposure bias* problem, meaning that the inference stage is exposed to wrong

predictions which have never been encountered during training.

In order to mitigate the exposure bias problem, Daumé et al. (2009) first advocate the idea of incorporating the model predictions at training time for structured prediction problems. Bengio et al. (2015) introduce the *scheduled sampling* strategy, which is a curriculum learning strategy. During training, for each time step, the sampling strategy decides with probability ε to use either the ground truth or previously predicted tokens. The latter can be a token randomly sampled according to the probability distribution $P(\cdot|\mathbf{e}_{<i}, \mathbf{f})$ or taken as the $\arg\max P(\cdot|\mathbf{e}_{<i}, \mathbf{f})$. It can also be a weighted average of all word embeddings, with $P(\cdot|\mathbf{e}_{<i}, \mathbf{f})$ as weights (Goyal et al., 2017). To increase robustness, one can also add noise to the probability distribution (Zhang et al., 2019). When $\varepsilon = 1$, the model always uses ground truth tokens as in conventional training, while when $\varepsilon = 0$, the model is trained with only predictions as in inference. Bengio et al. (2015) propose using a decay schedule of ε to go from the former to the latter during the training procedure. The model uses ground-truth tokens more frequently at the beginning of training as it is not well-trained to make reasonable predictions and samples more from its own prediction at the end of training.

Scheduled sampling suits the RNN-based model well as the RNN decoder always proceeds step by step. However, this is not straightforward for the Transformer model as the training procedure is highly parallelized. Forcing the Transformer model to proceed step by step as RNN models do yields significant computation overheads. Mihaylova and Martins (2019) successfully apply scheduled sampling to the Transformer models by performing a two-pass decoding strategy. The first decoding pass is like the conventional training of the Transformer, generating the output probability distributions for all positions. The second pass takes a mixture of the reference target sequence and the model predictions from the first pass as input to the decoder, in which each position decides to use the reference token with a probability ε. Liu et al. (2021a,b) further improve scheduled sampling for Transformers by taking into account model prediction confidence and by applying schedules not only based on the number of training steps but also on decoding steps, as errors are likely to appear as the decoding step accumulates.

2.4 . Neural Machine Translation Inference

The ultimate objective of NMT models is to generate a target sentence given the source. Therefore, during inference, NMT models aim to search for the optimal target sequence $\hat{\mathbf{e}}$, which delivers the highest conditional probability given the source $\mathbf{f}$ as:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} P(\mathbf{e}|\mathbf{f}; \theta). \tag{2.24}$$

However, the theoretical search space grows exponentially with respect to the target sequence length as it is possible to select all tokens in the vocabulary as output at each time step, making the exact search intractable. For instance, a target sequence with 10 tokens yields a space of $|V|^{10}$ possible sequences. An intuitive alternative is *greedy search*, which simply takes the most probable token at each time step:

$$\hat{e}_i = \arg\max_{e_i \in V} P(e_i | \mathbf{e}_{<i}, \mathbf{f}; \theta). \tag{2.25}$$

Nevertheless, greedy search is a single path with no way back. It cannot make any changes if errors are made at certain steps and have to continue searching with wrong predictions.

The most practical approach for standard NMT inference is *beam search*. Instead of always taking the most probable token, beam search keeps the top k most probable candidates at each time step. For time steps $i > 1$, each of the top k hypotheses generates a probability distribution over the vocabulary. Beam search only keeps the top-k hypotheses with the highest cumulative probability among $k \cdot |V|$ candidates. After each time step, the k hypotheses are re-ranked according to their cumulated probability. The searching procedure stops when the end of sentence symbol <eos> is generated or when the decoding reaches a predefined maximum step, and the hypothesis with the highest probability is considered as the final translation.

However, beam search tends to generate translations that lack diversity (Ott et al., 2018), which is sometimes problematic for data augmentation methods like back-translation (Sennrich et al., 2016b). Edunov et al. (2018) propose an alternative top-k sampling strategy to increase translation diversity. Unlike beam search, which keeps the top k candidates at each time step, the sampling strategy randomly samples a candidate from the k most probable candidates following their probabilities to increase the diversity of final translation. Even though the quality of translation generated via top-k sampling decreases compared to beam search, the increased diversity is helpful for applications like back-translation (Edunov et al., 2018).

2.5 . Evaluation Metrics

Evaluating translation quality is not a simple task, even for humans and much more so for machines. Besides, human evaluations are always expensive and time-consuming. Therefore, automatic evaluation metrics, which correlate with human evaluations and are inexpensive, are required to quickly evaluate translation hypotheses generated by different systems. In this section, we briefly discuss three metrics based on word-level matching and two other metrics based on sentence representation similarity comparison using recently proposed large pre-trained language models.

2.5.1 . BLEU

The most commonly used MT evaluation metric is the Bilingual Evaluation Understudy (BLEU, Papineni et al., 2002). It measures the corpus-level n-gram match precision, factored by a brevity penalty for short hypotheses. BLEU computes a modified n-gram precision for a candidate corpus by counting n-gram matches over the corpus:

$$p_n = \frac{\sum_{\hat{\mathbf{e}} \in C} \sum_{w_n} \min(\text{count}(w_n|\mathbf{e}), \text{count}(w_n|\hat{\mathbf{e}}))}{\sum_{\hat{\mathbf{e}} \in C} \sum_{w_n} \text{count}(w_n|\hat{\mathbf{e}})}, \tag{2.26}$$

where w_n is an n-gram, C is the candidate corpus, and $\hat{\mathbf{e}}$ and $\mathbf{e}$ refer to a hypothesis and a reference translation, respectively. $\text{count}(w_n|\mathbf{e})$ computes the number of times n-gram w_n appears in a sequence $\mathbf{e}$.

The n-gram precision penalizes translations that are longer than the references. However, it cannot reflect hypotheses with shorter lengths well. Papineni et al. (2002) introduce a *brevity penalty* (BP) score, which is a factor that penalizes short hypotheses:

$$\text{BP} = \begin{cases} 1 & \text{if } |\hat{C}| > |C| \\ \exp(1 - \frac{|C|}{|\hat{C}|}) & \text{if } |\hat{C}| \leq |C|, \end{cases} \tag{2.27}$$

where $|\hat{C}|$ and $|C|$ refer to the number of tokens in the hypothesis and reference corpus. In practice, BLEU is usually computed with $N = 4$, thus counting precisions from uni-gram to 4-grams. The exact BLEU score takes the geometric average of n-gram precisions factored by BP:

$$\text{BLEU}(\hat{\mathbf{e}}, \mathbf{e}) = \text{BP} \cdot \exp(\frac{1}{4} \sum_{n=1}^{4} \log p_n). \tag{2.28}$$

BLEU score thus ensures that a high-scoring translation must match the reference in length, word choice, and word order.

However, BLEU score and other exact word matching metrics are highly influenced by the tokenization of sentences. For example, [*Thank you.*] and [*Thank you* .] are semantically identical but are considered as two different sentences with different lengths. Besides, "*you.*" is considered as one single word, which is neither matched to "." nor to "*you*". Post (2018) propose SacreBLEU, a standard implementation of the BLEU metric, which always requires translations and references to be of raw text as input and applies internal tokenization to both texts so that the tokenization will not influence the final BLEU score anymore. The use of a standardized tool as SacreBLEU is very important but is still far from being widely adopted by the MT community (Marie et al., 2021).

2.5.2 . METEOR

The METEOR score (Banerjee and Lavie, 2005; Denkowski and Lavie, 2014), standing for the Metric for Evaluation of Translation with Explicit ORdering, considers not only exact word matches but also matches of stems and synonyms. It first computes a unigram matching alignment between the translation and reference. The alignment here is defined as each unigram in one sequence being matched to at most one unigram in the other sequence. Unigram matches take into account not only exact word matches but also when the two words are the same after being stemmed and if they are synonyms. With the unigram alignment, it then obtains the number of matched unigrams m, the number of unigrams in the translation t, and the reference r. The unigram precision $P = m/t$ and recall $R = m/r$ are used to compute the F_{mean} as:

$$F_{mean} = \frac{P \cdot R}{\alpha P + (1-\alpha)R}. \tag{2.29}$$

METEOR also incorporates a fragmentation penalty. It computes n-grams of unigram matches so that it can obtain a minimum number of matched n-grams ck. The penalty is then computed as:

$$Pen = \gamma \cdot (\frac{ck}{m})^{\beta}, \tag{2.30}$$

and the final METEOR score is therefore:

$$\text{METEOR}(\hat{\mathbf{e}}, \mathbf{e}) = (1 - Pen) \cdot F_{mean}. \tag{2.31}$$

The hyperparameters α, β, and γ need to be tuned for different languages so that METEOR can maximize the correlation with human judgments. The stemming and synonyms are also language-specific, which limits the ability to compute and compare METEOR scores across languages.

2.5.3 . TER

Snover et al. (2006) propose Translation Edit Rate (TER), which is a metric that measures the amount of editing required to transform a translation into an exact reference. The TER score computes the number of edits normalized by the reference length:

$$\text{TER}(\hat{\mathbf{e}}, \mathbf{e}) = \frac{\#\text{Edits}}{\#\text{Reference words}}, \tag{2.32}$$

where the number of edits computes the number of insertions, substitutions, deletions, and segment shifts. The segment shift is defined as moving a contiguous sequence of words within the hypothesis to another location. The number of shifts is obtained by repeatedly selecting the shift that reduces the number of insertions, substitutions, and deletions at most until no more beneficial shifts remain within the hypothesis.

2.5.4 . Metrics Based on Pre-trained Language Models

With the advance of large pre-trained language models, several metrics incorporate pre-trained models to obtain high-quality sentence embeddings that are used to compute the similarity between the hypotheses and the reference. Pre-trained model-based metrics show a better correlation with human judgments than word-level matching-based metrics.

Rei et al. (2020) propose the COMET score, standing for the Crosslingual Optimized Metric for Evaluation of Translation. COMET uses a pre-trained cross-lingual language model to encode the hypothesis translation, the reference, and also the source sentence. It obtains a vector representation of a sentence with a mean pooling operation to compute the averaged embedding over tokens. The states are computed as the weighted sum of the pre-trained model's output at each encoder layer. With the obtained sentence embeddings, Rei et al. (2020) further train a small network on quality estimation data to either directly regress to a quality score or measure the distance between the hypothesis and both source and reference.

Zhang et al. (2020b) propose BERTScore. They compute the similarities between all tokens in the hypothesis and all tokens in the reference. The similarity is defined as the cosine similarity between contextualized token embeddings. They use pre-trained BERT models to encode the hypothesis and the reference to obtain the output states of each token as the token embedding. They further compute the token match precision and recall by taking the token with the highest similarity score as a match and the F1 score as the harmonic mean of precision and recall.

2.6 . Conclusion

In this chapter, we introduced the conventional NMT systems used in this thesis. We have reviewed well-known text tokenization methods, various NMT model architectures, conventional training and inference procedures, and several evaluation metrics. This chapter has not covered all works for each topic we discussed. We refer to Mielke et al. (2021) for a more complete review of text tokenization methods; Dabre et al. (2020) for more developments of multilingual NMT; and Wiher et al. (2022) for more discussions about decoding strategies.

There are also many aspects of NMT that are not covered in this chapter. We will review NMT tasks using augmented sources other than only the source sentence in Chapter 3. These tasks are highly related to building bilingual writing systems, which incorporate bilingual information in the translation process. We will introduce non-autoregressive NMT, which generates multiple tokens in one decoding step, in Chapter 4. Non-autoregressive

models greatly accelerate the inference speed, which is suitable for interactive systems that need to update translations frequently. We refer to further readings for some subdomains of NMT research that are not discussed in this thesis, like data augmentation methods to enlarge the training corpus (Wang et al., 2018b; Edunov et al., 2020); low-resource language translation (Sennrich and Zhang, 2019; Wang et al., 2021; Haddow et al., 2022); parallel data cleaning and selection (van der Wees et al., 2017; Koehn et al., 2018, 2019, 2020); unsupervised NMT (Artetxe et al., 2018; Lample et al., 2018a,b); speech translation and simultaneous translation (Anastasopoulos and Chiang, 2018; Dalvi et al., 2018; Arivazhagan et al., 2019; Elbayad et al., 2020); domain adaptation of NMT (Chu and Wang, 2018; Pham et al., 2021; Saunders, 2021); document-level NMT (Abdul Rauf and Yvon, 2020; Maruf et al., 2021); robust NMT (Cheng et al., 2018, 2019), etc.

3 - Neural Machine Translation with Augmented Sources

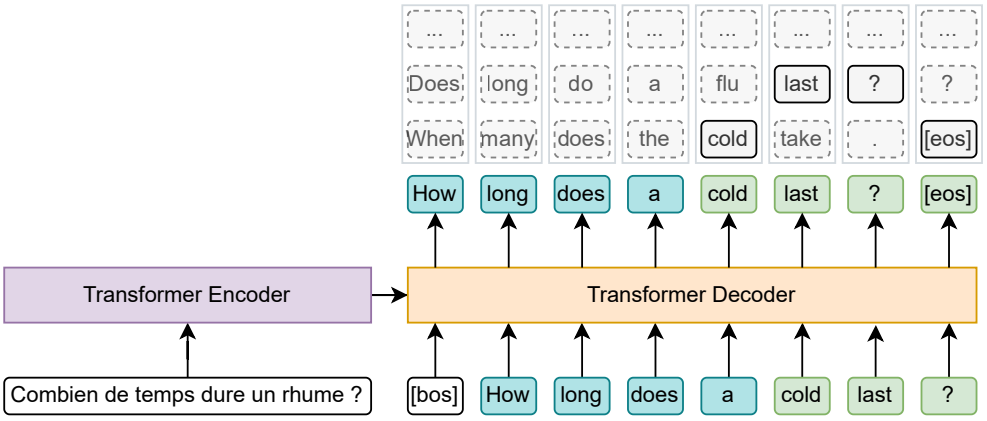
Standard neural machine translation (NMT) systems take only a source sentence as input and generate output translations from scratch on the target side. However, NMT systems can incorporate augmented sources other than the source sentence to help improve the translation quality. This is also the case of bilingual writing, where text segments are composed in both languages, therefore containing bilingual information that may help both text segments when generating monolingual texts. In this chapter, we describe several applications integrating augmented sources.

3.1 . Interactive Machine Translation

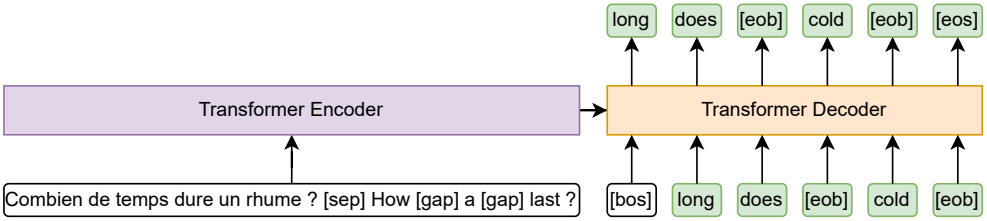
Interactive machine translation (IMT), also known as interactive translation prediction (Knowles and Koehn, 2016; Santy et al., 2019), text prediction (Foster et al., 2002), etc., is an important branch of Computer Aided Translation (CAT). IMT systems are designed to help human translators produce high-quality translations. They repeatedly produce translation suggestions or completions while the human translators type a translation (Langlais et al., 2000). Human translators can decide to accept the suggestion provided by IMT systems. Otherwise, they can type a translation if the suggestion is not satisfactory. IMT systems will then generate new suggestions based on the translation segments entered by human translators.

3.1.1 . Prefix Decoding

In IMT, human translators are always in control of the entire translation process. Moreover, IMT systems should respect the texts typed or accepted by human translators when generating new suggestions (Langlais et al., 2000). Therefore, the core task for IMT systems is to provide translation suggestions based on specific constraints imposed by human translators. Most works consider a left-to-right translation mode, where human translators generate or accept translations step by step from the beginning. In this situation, the constraint is a *prefix* of a translation. IMT systems thus perform a prefix-constrained prediction to generate suffix translations by taking both the source sentence and the constrained prefix as input (Langlais et al., 2000; Koehn et al., 2014; Alabau et al., 2014; Green et al., 2014; Wuebker et al., 2016; Knowles and Koehn, 2016; Santy et al., 2019). Thanks to the autoregressive nature of NMT systems, prefix decoding can be straightfor-



(a) Prefix decoding. Prefix tokens are in blue, while model predictions are in green.



(b) Text infilling model of Xiao et al. (2022a). The model tends to fill each `[gap]` in the initial target sequence and only produces translation segments. Segments are separated by a `[eob]` token.

Figure 3.1: Two methods to incorporate human inputs for IMT.

wardly implemented by conditioning on the given prefix $\mathbf{e}^*_{\leq t}$ and the source sentence to generate the target tokens: $p(e_i|\mathbf{e}^*_{\leq t}, \hat{\mathbf{e}}_{[t+1,i-1]}, \mathbf{f}; \theta)$ (Knowles and Koehn, 2016; Santy et al., 2019). As illustrated in Figure 3.1a, tokens predicted by IMT systems at the initial positions $i \leq t$ are ignored, and the beam search strategy only starts to consider various candidates from the position $t + 1$.

3.1.2 . Relaxing Constraint Order

The strict left-to-right translation may not be an ideal writing mode for human translators, as some may prefer to first translate difficult words and then easy ones (Huang et al., 2021). Some studies have explored several alternatives to the strict prefix-constrained decoding.

One of the directions is to allow human modifications or indications of a complete initial translation. Note that this approach is more related to post-editing, which is another important branch of CAT tools. Post-editing requires human translators to edit a translation produced by a machine translation (MT) system instead of starting to translate from scratch with generated

suggestions. However, since this approach involves more substantial interaction between humans and systems than post-editing and proposes translation suggestions, we also consider it as another type of IMT. Marie and Max (2015) initially attempt to revise an initial translation in a touch-based scenario. In their proposed system, users only need to select usable translation segments, which do not require to be a prefix, while the IMT system automatically uses the selected spans to retranslate the source sentence. This process can be iteratively repeated until there are no more improvements. Grangier and Auli (2018) study a similar approach by crossing-out undesired words in a proposed translation. They use a dual-encoder sequence-to-sequence architecture where the second encoder encodes the marked translation as input. Using a dual encoder to integrate the augmented sources is a common practice. More details about the dual-encoder architecture are discussed in Sections 3.2 and 3.5. Wang et al. (2020) further explore this idea by marking a sequence of actions to a given translation and then using a dual-encoder model to perform translation by taking the marked translation as a second input.

Another direction focuses on giving more freedom to human translators when performing translations. Lee et al. (2021), Li et al. (2021), Huang et al. (2021), and Yang et al. (2022) allow users to perform edits at any position in a hypothesis sequence and provide translation suggestions or autocompletion based on context information from both prefixes and suffixes. Xiao et al. (2022a) extend these approaches by considering a text infilling task to generate multiple missing segments from a source sentence concatenated with pre-translated segments. Figure 3.1b illustrates this approach. Their method allowed the translation segments approved by human translators to be interspersed among gaps. They use a special token to mark the gaps between two pre-translated segments to construct an initial target sequence. Instead of using dual-encoder architectures, they concatenate the source sequence with the initial target sequence, with a separator token in between, to make a long input sequence on the source side. The approach to concatenate the extended sources with the source sentence has been widely explored across various applications, which we also discuss in the following sections. On the target side, instead of producing the entire target sentences, Xiao et al. (2022a) only output segments that are predicted to infill the gaps in the initial sequence. The final translation is constructed by replacing gaps in the initial target sequence with predicted segments.

3.2 . Automatic Post-Editing

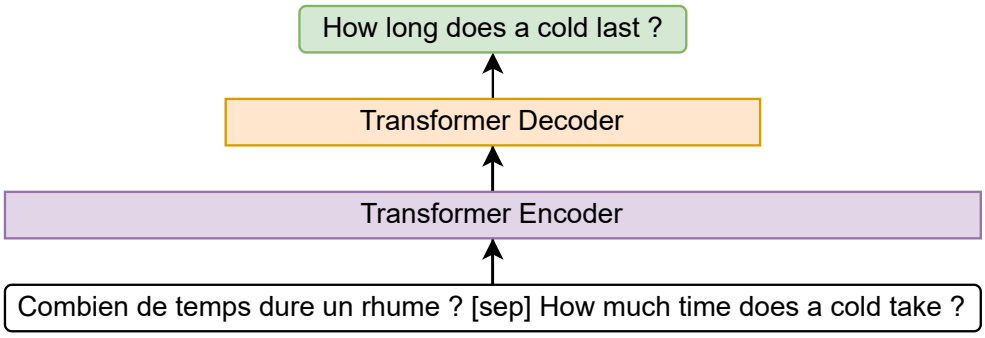
The automatic post-editing (APE) task focuses on automatically editing a machine-generated translation to improve the translation quality and

correct translation errors to reach a good translation (Simard et al., 2007). Therefore, unlike MT systems which only require source-target parallel data, APE systems need triplets of examples consisting of a source sentence, a target sentence generated by an MT system, and a human post-edited version of the machine-generated target sentence. The last element should not be a reference translation, which is different from regular MT tasks, as APE systems need to learn from human post-editing to perform edits to the translation (do Carmo et al., 2021). Using reference translations may misrepresent the editing patterns between MT outputs and human post-editing results. However, such triplets are challenging to collect as human post-edited translations are often proprietary and difficult to get access to. To this end, several works studied exploiting artificial APE triplets to alleviate the data scarcity problem. Junczys-Dowmunt and Grundkiewicz (2016) use round-trip translation to create synthetic source and MT output by first translating monolingual target language sentences into the source language, then retranslating them back to the target language. Negri et al. (2018) approximate the actual APE triplets by treating parallel corpora from MT tasks as the source and artificial human post-edits and generating MT outputs from the source sentences with separate MT systems.

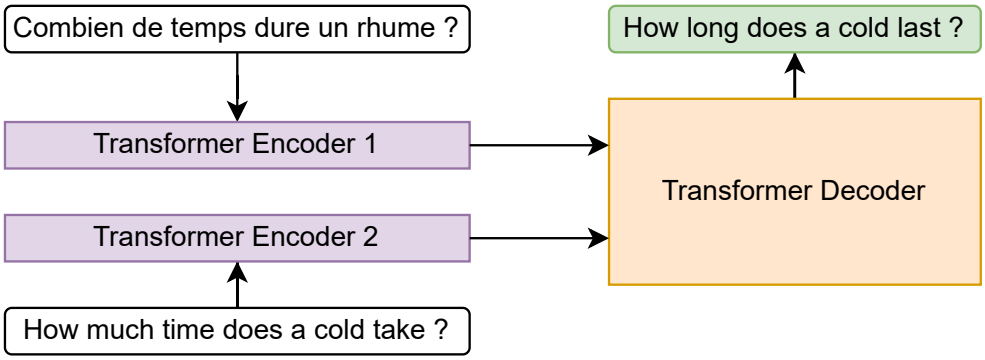
Neural APE methods mainly focus on two general directions: dual-encoder models and single-encoder models with a concatenated long input sequence. As we mentioned in Section 3.1 and will also discuss in the following sections, these two directions are also widely used in other applications integrating augmented sources, like IMT, NMT with translation memories and lexical constraints, multi-source and bidirectional translation, etc.

3.2.1 . Single Encoder Model

Crego et al. (2016) and Hokamp (2017) perform APE by concatenating the source sentence and the MT output into a long sequence. They separate the two sentences with a special token, as illustrated in Figure 3.2a. Niehues et al. (2016) propose a similar approach, which concatenates a pre-translation generated by a statistical machine translation (SMT) system to the source. They mark tokens from the source and target languages with different characters to distinguish the two languages. In this single-encoder approach, the encoder can process inputs with a mixture of language. The advantage of this approach is that it does not require modifications to the model architecture, meaning that it can be easily applied to various implementations. Correia and Martins (2019) and Lopes et al. (2019) incorporate pre-trained multilingual BERT models (Devlin et al., 2019) into an encoder-decoder architecture. BERT, referring to Bidirectional Encoder Representations from Transformers, is a large-scale language model trained on a very large dataset. It is able to output high-quality sentence representations that could be di-



(a) Single encoder model with a concatenated input.



(b) Dual encoder model.

Figure 3.2: Single and dual-encoder models incorporating augmented sources.

rectly used or fine-tuned for downstream tasks. Correia and Martins (2019) and Lopes et al. (2019) initialize their encoder and decoder with the multilingual BERT parameters and used the concatenated sequence as input to perform APE.

Despite various approaches developed for APE, with the great and rapid advance in NMT, it has become more difficult for APE models to improve NMT model outputs which are already of high quality (Chollampatt et al., 2020).

3.2.2 . Dual Encoder Model

The dual-encoder model involves two encoders, one for the source sentence and the other for the MT output. Figure 3.2b shows the dual-encoder approach. We mainly focus on Transformer-based approaches. A typical dual-encoder Transformer was proposed by Junczys-Dowmunt and Grundkiewicz (2018). They use two separate encoders with the same architecture to map the source sentence and the MT output. In their decoder, they use

two stacked encoder-decoder cross-attention layers in each decoder block, with one layer attending to the representation of the source encoder and the other attending to the hidden state generated from the MT output encoder. Shin and Lee (2018) study a similar approach, with an additional cross-attention layer in the decoder to compute the attention between the output of the two encoders. Tebbifakhr et al. (2018) also apply two encoders. However, they concatenate the outputs from the two encoders as one sequence to perform cross-attention on the decoder side. The decoder side of their approach remains unchanged compared to the basic Transformer architecture. Pal et al. (2018) further incorporate a third encoder which maps the concatenated output sequence from the two encoders to a new representation for the decoder.

3.3 . Neural Machine Translation with Translation Memories

A Translation Memory (TM) is a database that contains segments in a source language with the corresponding translations in a target language, which are recorded during the regular activities of human translators. It is mainly used to match up previously generated translations to new content similar to the contents translated in the past. Many domains, like technical domains, contain highly repeated segments, terms, and phraseology, which are more efficient for reusing existing translations rather than translating from scratch. We assume that this pair of sentences exists in the TM, which translates the following English sentence into French: *[How long does the flight last?]* $\rightsquigarrow$ *[Combien de temps dure le vol?]*. If the same source sentence appears in a future document (an *exact match*), the TM will suggest reusing the translation that has been saved. In addition to exact matches, TMs are also useful with *fuzzy matches*, which can help to reduce translation effort and to increase consistency when a new sentence is similar to a previously translated sentence but not identical. For example, when translating the input sentence: *[How long does a cold last?]*, the TM may also suggest the previous translation since only two replacements (*un rhume (a cold)* $\rightarrow$ *le vol (the flight)*) are needed to achieve a correct translation.

For decades, the localization industry has proposed TM technologies in CAT tools to allow human translators to visualize one or several similar or usable translations extracted from the TM when translating a sentence, leading to higher productivity and consistency (Yamada, 2011). Hence, even though the retrieval methods of TM differ among CAT tools (Bloodgood and Strauss, 2014), human translators generally accept discounted translation

rates for sentences with high fuzzy matches.[1]

The idea of using TMs to help translation is also explored in the MT area, where extracted translations are used to help improve the translation quality. In the context of MT, TM simply refers to parallel bilingual data. In earlier SMT approaches, the TM retrieval shares the same idea with SMT about managing and retrieving the optimal combination of longest translated n-grams. This property leads to the development of several techniques like the use of TM in SMT decoding (Koehn and Senellart, 2010; Wang et al., 2013), adaptive MT (Zaretskaya et al., 2015) or fuzzy match repairing (Ortega et al., 2016; Knowles et al., 2018).

In the context of NMT, methods using TMs to help improve translation quality are also known as retrieval-based approaches. They mainly contain two stages: the retrieval of similar translation segments and the integration of similar translations into NMT systems. Integrating TMs into NMT is structurally identical to the APE task. However, as Sánchez-Gijón et al. (2019) indicate, processing TMs and MT outputs requires different efforts in human post-editing. The edition of automatic (noisy) translation is quite different from the edition of a good translation of a similar source sentence.

3.3.1 . Retrieving Similar Translations

Consider a TM as a set of N sentence pairs $\{(\mathbf{f}, \mathbf{e})_i, i = 1, \ldots, N\}$ where $\mathbf{f}_i$ and $\mathbf{e}_i$ are mutual translations. A TM must be conveniently stored to allow fast access to the pair $(\mathbf{f}_i, \mathbf{e}_i)$ that shows the highest similarity between $\mathbf{f}_i$ and a given new sentence $\mathbf{f}$. Many methods to compute sentence similarity have been explored, mainly falling into two broad categories: lexical match (*i.e.,* fuzzy match) and distributed representation match. The former relies on the ratio of overlaps between the source sentences taken into account, while the latter counts on the generalization power of neural networks when building sentence representations.

Fuzzy Matching

Fuzzy matching is a lexicalized matching method aimed at identifying sentences that are similar to a given sentence $\mathbf{f}$. The fuzzy matching score between two sentences $\mathbf{f}_i$ and $\mathbf{f}_j$ is defined as:

$$\mathrm{FM}(\mathbf{f}_i, \mathbf{f}_j) = 1 - \frac{\mathrm{ED}(\mathbf{f}_i, \mathbf{f}_j)}{\max(|\mathbf{f}_i|, |\mathbf{f}_j|)}, \tag{3.1}$$

where $\mathrm{ED}(\mathbf{f}_i, \mathbf{f}_j)$ is the edit distance between $\mathbf{f}_i$ and $\mathbf{f}_j$, and $|\mathbf{f}|$ is the length of $\mathbf{f}$. Many variants have been proposed to compute the edit distance, generally

[1] https://signsandsymptomsoftranslation.com/2015/03/06/fuzzy-matches/

performed on normalized sentences (ignoring, for instance, case, number, punctuation, spaces, or inline tag differences that are typically handled at a later stage). Also, inverse document frequency and stemming techniques are used to give more weight to significant words or less weight to morphological variants (Bloodgood and Strauss, 2014; Vanallemeersch and Vandeghinste, 2015).

Fuzzy matching delivers excellent performance under large overlapping conditions. However, it requires two sentences to be similar in terms of length. When it is not the case, sentences with large overlaps may receive low FM scores. Consider, for instance, the input: *[How long does the flight arriving in Paris from Barcelona last?]* and the TM entry of the previous example: *[How long does the flight last?]* ⇝ *[Combien de temps dure le vol?]*. Even though the TM entry may be of great help when translating the input sentence, it receives a low FM score ($1 - 5/12 = 0.583$) because of multiple insertion/deletion operations needed. Therefore, a second lexicalized similarity measure that focuses on finding the longest n-gram overlap between sentences is also used.

N-gram Matching

N-gram matching is also called subsequence matching or chunk matching in CAT tools and is usually combined with source-target alignment to help human translators find translation fragments. The n-gram matching score between $\mathbf{f}_i$ and $\mathbf{f}_j$ is defined as:

$$\mathrm{NM}(\mathbf{f}_i, \mathbf{f}_j) = \left| \max\Big(\{\mathcal{N}(\mathbf{f}_i) \cap \mathcal{N}(\mathbf{f}_j)\}\Big) \right|, \tag{3.2}$$

where $\mathcal{N}(\mathbf{f})$ denotes the set of n-grams in sentence $\mathbf{f}$, $\max(q)$ returns the longest n-gram in the set q, and $|r|$ is the length of an n-gram r.

Sentence Representation Matching

The current research on sentence-level similarity measures has made tremendous advances thanks to distributed word representations computed by neural networks (Mikolov et al., 2013a,b; Pennington et al., 2014; Joulin et al., 2017; Bojanowski et al., 2017) and large pre-trained language models. These methods are used to compute the vector representation of a given sentence, also denoted as sentence embedding. Well-known sentence embedding methods include LASER (Artetxe and Schwenk, 2019) and LaBSE (Feng et al., 2022). The similarity score between sentences $\mathbf{f}_i$ and $\mathbf{f}_j$ is computed as the cosine similarity of their distributed representations $\mathbf{h}_i$ and

$\mathbf{h}_j$:

$$\mathrm{EM}(\mathbf{f}_i, \mathbf{f}_j) = \frac{\mathbf{h}_i \cdot \mathbf{h}_j}{||\mathbf{h}_i|| \times ||\mathbf{h}_j||}, \tag{3.3}$$

where $||\mathbf{h}||$ denotes the magnitude of vector $\mathbf{h}$. There exists fast retrieval implementations like the `faiss`[2] toolkit (Johnson et al., 2021), which can quickly compute the similarities between a query vector and a set of vectors, providing fast retrieval of similar translations.

Sentence similarity methods based on distributional representations typically outperform lexicalized methods in capturing semantic similarity, particularly under small lexical overlap conditions. To this end, Cai et al. (2021) and Khandelwal et al. (2021) use the distributed representation computed by the NMT model itself to incorporate similar monolingual target sentences or contextualized target tokens instead of always retrieving sentences based on source side similarities.

3.3.2 . Integrating Translation Memories

NMT models incorporating TMs can be clustered into three main groups of approaches: single-encoder model with concatenated input, dual-encoder model, and nearest neighbor-based approaches (Khandelwal et al., 2021). The first two groups are similar to the APE models described in Section 3.2, using complete TM sentences. The third group tends to extract and integrate similar translation segments or tokens in a similar context instead of the entire sentence.

Bulte and Tezcan (2019) experiment with concatenating retrieved similar translations to the source sentence as a long input sequence without modifying the model architecture. Therefore, the model can learn to directly copy useful segments from the concatenated similar translations. They explore using up to 3 similar translations and concatenate them all together with the source sentence. We extend this approach in (Xu et al., 2020) by distinguishing tokens in a similar translation that are aligned to source tokens from those "unrelated" tokens that cannot find counterparts in the source. We add a second input feature to the encoder embedding that marks source tokens, aligned target tokens, and unrelated tokens. More details about detecting unrelated tokens are presented in Chapter 8. We also propose "Priming NMT" in (Pham et al., 2020), in which we use both the source $\tilde{\mathbf{f}}$ and the target $\tilde{\mathbf{e}}$ of the retrieved examples. Different from Bulte and Tezcan (2019), we concatenate the similar source $\tilde{\mathbf{f}}$ and the input $\mathbf{f}$ to make the source sequence and train the model to translate the corresponding target sequence by concatenating the similar translation $\tilde{\mathbf{e}}$ and the reference $\mathbf{e}$. In Priming NMT, the encoder does not contain target sequences. The model

[2] https://github.com/facebookresearch/faiss

learns to use $\tilde{\mathbf{f}}$ from the source side and $\tilde{\mathbf{e}}$ as a previously generated sequence on the target side to produce the translation. During inference, the source side remains the same as training, and we prefix the target side with $\tilde{\mathbf{e}}$ using prefix decoding to only output the desired target translation.

Gu et al. (2018b) use an external black-box search engine to search for similar translations and integrate TM by rerunning the NMT model to obtain context vectors of similar translations. To some extent, this approach is equivalent to using a second encoder with a shared parameter. Xia et al. (2019) extend this idea by incorporating TMs into a compact graph as a second encoder. Cai et al. (2021) jointly train the retrieval module together with the NMT model to extract translation-specific monolingual target sentence and incorporate it with an extra encoder. This dual-encoder approach is similar to the approach of Junczys-Dowmunt and Grundkiewicz (2018) for APE. He et al. (2021b) encode retrieved similar translations using the decoder embedding matrix and integrate them on the decoder side by performing cross-attention between the decoder input sentence and the similar translation. This approach is more like a dual "decoder" architecture.

Zhang et al. (2018a) incorporate cross-lingual alignment information to retrieved TMs and extract only similar target segments whose aligned source pieces were present in the input source sentence. The extracted similar pieces are used to guide NMT in terms of extra rewards during decoding. Khandelwal et al. (2021) further extend this idea by performing k-*nearest neighbor* (kNN) MT. For each decoding step, they search for k target tokens with similar contexts in the TM and use them to help generate the token prediction probability. Zheng et al. (2021) relax using fixed kNN for each target token by dynamically deciding the number of neighbors to use among the k neighbors for each decoding step. However, the kNN-based approaches require extensive computing efforts as it needs to compute and store the contextualized vector representation of each token in each target sentence of the TM and perform a kNN search of vectors in a very large data store for each beam at each decoding step.

Retrieval-based approaches can improve translation quality by using TMs. Therefore, they are often applied to perform domain adaptations by searching for similar translations in specific domains (Xu et al., 2020; Khandelwal et al., 2021). Sentence-level retrieval is often very fast and only yields negligible time overheads. However, it often requires a good match to help the translation. For some domains, especially more general domains, it is difficult to find a proper translation similar to the source at the sentence-level, and using poor matches even degrades the translation quality (Xu et al., 2020). On the contrary, token-level retrieval can search for more accurate matches for each token that cannot be found only by searching for entire sentences but with a significant retrieval time overhead. However, the performance

gains of kNN MT on more general domains are still much smaller than those on specific domains.

3.4 . Lexical Constraints for Neural Machine Translation

NMT models typically operate in an end-to-end fashion. Therefore, it is not obvious to introduce additional knowledge or add controls during the translation process. However, it is useful to introduce external sources of knowledge in some scenarios, like incorporating domain-specific terminologies or respecting human-entered translation segments like in IMT to generate higher quality translations. The extra knowledge often appears as terminology words or pieces of translation segments that are considered as lexical constraints and are required to be present in the output translation. Studies on NMT with lexical constraints focus on generating a good translation that properly includes given constraints.

3.4.1 . Lexically Constrained Decoding

Lexically constrained decoding (LCD) aims to force NMT models to produce all required constraints in their exact form in the translation. Hokamp and Liu (2017) propose using grid beam search (GBS) for LCD. Instead of maintaining only one beam at each time step, they explore a grid of beams, in which the hypotheses of each beam were filled with both the token generated by the model and force-predicted constrained tokens. The GBS procedure only stops if all constrained tokens are predicted, and the model predicts the end-of-sentence symbol. The hypothesis sequence that covers all constraints with the highest probability in the beam set is selected as the final translation. In this way, GBS assures that all constrained tokens are present in the output translation. However, the expanded beams also cost a large runtime overhead. The complexity increase is linear to the number of token constraints compared to standard beam search decoding.

Post and Vilar (2018) and Hu et al. (2019) improve the GBS by a dynamic beam allocation. They expand candidates with constrained tokens and keep only one single beam of hypotheses at each time step as regular beam search decoding to make the complexity independent to the number of constraints. Hasler et al. (2018) also study similar approaches to incorporate terminology constraints via constrained beam search (Anderson et al., 2017). However, even with the improvements, LCD still encounters a large decoding overhead compared to standard beam search. Additionally, LCD sometimes hurts the overall quality of generated translations, as it breaks the semantic assessment of NMT models by forcing constraints to occur in the final translation, even

when they do not fit in the context or when a synonym is selected by the NMT model.

3.4.2 . Soft Lexical Constraints

LCD methods mainly focus on the decoding step by injecting hard constraints with a significant efficiency loss. On the contrary, other approaches have focused on training the NMT model to use additional terminology translations. Crego et al. (2016) first propose to replace domain-specific source terminologies with abstract placeholder tokens. They train NMT models with pre-processed data containing the placeholder on both the source and target sides so that the model learns to predict the corresponding placeholder token on the target side. The final translation is obtained by replacing target placeholders with the translated terminologies.

Dinu et al. (2019) first propose to inject the terminology translation into the source sentence. They experiment with two approaches. The first one appends the terminology translation to the source word. Taking the example from Section 3.3, the source sentence is augmented as: *[How long does the **flight vol** last?]*, making it a sentence mixing two languages. The second approach directly replaces the source terminology by its translation: *[How long does the **vol** last?]*. In both settings, the target side remains unchanged. Dinu et al. (2019) train NMT models with the augmented source sentence to learn a *copy behavior* of the terminology translation. This is somewhat similar to Bulte and Tezcan (2019), even though the source sequence structures are different. Song et al. (2019) also apply the "replace" setting of Dinu et al. (2019) and incorporate a pointer network (Gulcehre et al., 2016) to enhance the copy behavior.

All these approaches do not involve any modifications to the decoding stage and use a standard beam search strategy. Therefore, these approaches do not incur any overhead during decoding. The terminology translations injected on the source side are soft constraints, as there is no guarantee that they will always appear on the target side during decoding. Dinu et al. (2019) show that this soft terminology constraint obtained a terminology usage rate greater than 90% for both settings.

3.4.3 . Morphological Inflection

Crego et al. (2016) and Dinu et al. (2019) focus on direct copies of the terminology constraints, while LCD methods involve hard constraints that might not suit the grammatical context of the translation. The above approaches lack the flexibility to accommodate terminological variation, *e.g.*, due to morphological inflection, as terms may not always be in the correct form and are even provided as lemmas in some real-world scenarios.

To address this issue, Michon et al. (2020) extend Crego et al. (2016) by replacing terms with placeholders indicating part-of-speech and morphological information on both source and target sides. Their model can learn to translate dedicated source placeholders into proper target ones with the correct morphological change. This helps the translation by generating placeholders with morphological information and facilitates the post-processing to use the correct form of target terminologies.

Bergmanis and Pinnis (2021) and Niehues (2021) extend Dinu et al. (2019) by annotating the source side word with the target lemma instead of an exact translation with correct morphological inflection. Their models do not learn to simply copy the target lemma but to perform *copy-and-inflect* behavior instead. Xu and Carpuat (2021b) introduce a cross-lingual rule-based inflection module to predict the target inflected form of each lemma constraint solely based on the source sentence. The predicted inflected target constraints can be integrated into NMT models as in (Dinu et al., 2019) or in other models.

3.5 . Multi-source/Multi-target Translation

Another common situation for MT applications that involve auxiliary resources is multi-source/multi-target translation. Multi-source translation (Och and Ney, 2001; Crego et al., 2010) aims to translate multiple source languages into a single target language, while multi-target translation (Dong et al., 2015) does the opposite, translating one source language into multiple target languages.

3.5.1 . Multi-source Translation

The principal purpose of multi-source translation is to use multi-parallel corpora to improve the translation quality. Zoph and Knight (2016) use a multi-encoder NMT model to encode two source languages at the same time. They combine the hidden states from both encoders into a single hidden state before passing to the decoder. Firat et al. (2016b) and Garmash and Monz (2016) explore a *mixture of NMT experts* ensemble approach. They adopt an encoder-decoder NMT model for each source-target language pair and combine the probability distribution of separate NMT models at each time step to make the prediction. Probability distributions are combined as a weighted sum where a gated network computes the weights to control the contributions of each model. Dabre et al. (2017) propose a different way to concatenate parallel sentences in multiple source languages into a long input sentence, similar to Bulte and Tezcan (2019). The only difference is that they do not use any separator tokens to indicate individual source sentences

and rely on the model to learn to distinguish sentence boundaries.

All the above methods require multi-parallel corpora, in which all languages are aligned among each other. Even though the separate models in the ensemble approach can be pre-trained on different bilingual data for each language pair, they still need multi-parallel data to train the gated network. However, multi-parallel data is relatively scarce compared to bilingual data, thus limiting the possible use cases of large-scale multi-source NMT to restricted domains (*e.g.*, multilingual institutions or bodies). Nishimura et al. (2018) extend the above two approaches to perform translation when some multi-source inputs are missing. They use a special token to replace the missing input sentence and train multi-source models to ignore the input when the missing token is present. This approach makes a step towards exploring incomplete multi-parallel data. Choi et al. (2018) create synthetic multi-parallel data by translating sentences from a pivot language into missing languages to improve translation quality for low-resource language pairs. Xu et al. (2021c) further study using auxiliary languages in a multi-source setting. Similarly, they create synthetic sentences in the auxiliary languages by translating from bilingual training data using a pre-trained multilingual NMT model. This idea is also similar to Niehues et al. (2016), as discussed in Section 3.2, in which the SMT output acts like an artificial auxiliary source in the target language. Xu et al. (2021c) enable a multi-source NMT model to handle single-source situations by randomly dropping the auxiliary language during training. They study the concatenated multi-source and multi-encoder approaches. They also randomly drop the auxiliary language during training so that the model can perform standard translation without auxiliary sources. This is similar to Bulte and Tezcan (2019), who combine original parallel data and TM augmented data to train NMT models, which can translate with and without a TM.

3.5.2 . Multi-target Translation

As Och and Ney (2001) point out, the ultimate goal of multi-source translation is to use manually translated texts to help translate them into more languages. Another variant considers translating multiple target languages simultaneously, assuming weak target languages can benefit from the strong target languages. Although the multi-target concept can be generalized to translating into more than two target languages, most work has focused on generating two languages.

Neubig et al. (2015) first study simultaneously generating two targets under the SMT scenario, with the objective that a strong target language model can help the other language. Dong et al. (2015) explore multi-target NMT by performing multi-task learning. Their model contains a shared encoder and a separate decoder for each target language. The source sentence

representation is identical across translation directions so that source language data across language pairs can all be used. Even though Dong et al. (2015) aim to perform multi-target translation, their model is trained only with bilingual data for different language pairs. The multi-task objective is achieved in a general term by jointly optimizing the loss from different language pairs across mini-batches. In other words, this multi-task training scheme only handles one language pair at a time, even though it can perform simultaneous decoding by only encoding the source sentence once and translating with multiple decoders at the same time.

Wang et al. (2019) propose a synchronous self-attention framework to handle two target languages simultaneously. In this framework, the generation of a target language relies not only on the source sentence but also on the previously generated tokens from the other target language. The synchronous self-attention model consists of a shared encoder and two decoders. The two decoders interact with each other at the decoder self-attention layer of the Transformer model. Wang et al. (2019) use an additional decoder cross-attention module so that each decoder attends not only to itself but also to the other decoder. The hidden states from the two attention modules are then combined into one to pass through other layers.

The multi-target approach has also been applied to simultaneously generate a transcript and a translation from spoken input. Anastasopoulos and Chiang (2018) propose a triangle model to perform simultaneous automatic speech recognition (ASR) and translation. The speech translation task aims at translating an audio input into a target text in a different language. In this triangle model, the encoder computes vector representations of the audio source. The ASR decoder performs as a regular decoder, attending only to source representations, while the translation decoder attends to the hidden states of both the encoder and the ASR decoder to generate translations. Le et al. (2020) study the same task with a dual-decoder model similar to Wang et al. (2019). They additionally perform a decoder cross-attention in the encoder-decoder cross-attention sublayer, which enhances the interaction between the two decoders. The multi-target translation is also applied to the subtitling task to jointly generate consistent subtitles in various languages from an audio source (Karakanta et al., 2021). Our own work with dual decoding and its applications to multi-target translation is in Chapters 5 and 6.

3.5.3 . Bidirectional Decoding Translation

The concept of multi-target translation generalizes to produce several outputs at the same time, which also suits the bidirectional decoding translation setting. Regular MT tasks generate a target sentence from left to right, while bidirectional decoding MT aims to generate another target sequence in

the inverse direction from right to left. As the left-to-right generation cannot see predicted information in the future, bidirectional decoding MT tends to make use of information generated from the other direction to improve the translation quality.

Under the NMT scenario, Liu et al. (2016a) explore jointly training two separate NMT models, one for each direction, with a combined loss from both models. During decoding, the two models generate k-best hypotheses each, which are then re-scored to select the best translation. Zhang et al. (2018b) and Su et al. (2019) propose making use of the target side's future information by performing asynchronous bidirectional decoding. This architecture resembles the triangle model of Anastasopoulos and Chiang (2018). The backward right-to-left decoder performs as regular NMT decoders, only in a reverse direction, generating hidden states from right to left. The forward left-to-right decoder, which performs similarly to the translation decoder of Anastasopoulos and Chiang (2018), attends both to the source hidden states and the backward hidden states during decoding. Therefore, the forward decoder can make use of future context on the target side to generate better translations.

Zhou et al. (2019) and Zhang et al. (2020a) explore synchronous bidirectional decoding NMT. They use the same model as Wang et al. (2019) to generate in both directions at the same time. The final translation is the direction with a higher probability. The backward translation is reversed to make a proper sentence from left to right. More recently, He et al. (2021a) combine bidirectional decoding MT and multi-target MT together and managed to generate two languages in two directions simultaneously. For each target among the four, the prediction depends on the source as well as on all four targets. They extend Zhou et al. (2019), Wang et al. (2019), and Zhou et al. (2019) by making each decoder attend to all four decoders.

3.6 . Conclusion

In this chapter, we reviewed several NMT tasks incorporating augmented sources besides the source sentence. The augmented sources can be of various types, such as human-corrected translation segments, pre-translated MT outputs, similar translations extracted from TMs, lexical constraints and terminologies, translations from auxiliary languages, etc. Despite all these varieties, the internal methods to incorporate these augmented sources do not vary much. They can be clustered into two general approaches: the single-encoder approach that mixes the augmented sources with the source sentence in the same sequence, and the dual-encoder approach that uses an additional encoder to encode the augmented sources. Among these two approaches, the single-encoder approach does not require specific modifications

to model architectures and allows different types of language mix in the input sequence. Augmented sources can be concatenated to the source sentence or interspersed among source tokens, providing more flexibility for language mixing. Our works in Chapters 5, 7, and 8 are all based on a single-encoder that incorporates bilingual information.

4 - Non-autoregressive Neural Machine Translation

Conventional neural machine translation (NMT) systems perform translations in an autoregressive (AR) way, as described in Chapter 2. The generation of one token is conditioned on the source sentence and the previously generated target sequences, as in Equation (2.1). Therefore, the decoding procedure only generates one token at each decoding step. The AR characteristic suits Recurrent Neural Network (RNN)-based NMT models well, as RNN models cannot be parallelized and must proceed step by step. However, the AR decoding is slow as the decoding time increases linearly with the length of the target translation, and the computationally expensive NMT decoder is used at each time step to generate only one token. Besides, AR generation also suffers from the exposure bias problem, as introduced in Section 2.3.1. When performing offline translations, AR decoding can be largely accelerated with batch decoding to translate many sentences simultaneously (Helcl et al., 2022). However, in the case of interactive translation scenarios, NMT systems often only have a single sequence to deal with that cannot be batched. For interactive systems like bilingual writing tools, it is important to generate translations within a low latency so that users will not wait long to see the translation suggestions once they make changes to the texts.

Thanks to the Transformer model (Vaswani et al., 2017), which allows more parallelizable architectures, an important variant of NMT, namely non-autoregressive neural machine translation (NAT), has emerged (Gu et al., 2018a). NAT is able to generate multiple tokens in parallel in one single decoder step, as it removes the AR connection by making a conditional independence assumption that all tokens on the target side are independent of each other given the source sentence:

$$P_{\text{NAT}}(\mathbf{e}|\mathbf{f}) = P_L(T|\mathbf{f};\theta) \cdot \prod_{i=1}^{T} P(e_i|\mathbf{f};\theta), \tag{4.1}$$

where $P_L(T|\mathbf{f};\theta)$ measures the probability that the target sentence is of length T given the source. Compared to Equation (2.1), the conditioning history tokens $\mathbf{e}_{<i}$ are removed for NAT models. Unlike AR models, which gradually increase the output sequence step by step, NAT models aim to generate all tokens at the same time. Therefore, NAT models do not have any target information in the translation process and generate each token independently and solely based on the source. The independence assumption and missing target information make NAT a very challenging task, as NAT models *fail to capture the dependency within target tokens* (Xiao et al.,

2022b). The first NAT model of Gu et al. (2018a) performs far behind AR Transformer baselines.

Translations are multi-modal, meaning that a source sentence can be accurately translated into multiple correct target sentences. Such multi-modal translations possibly exist in the training data, and NAT models are trained to cover all possible modes. During inference, without explicit indications, NAT models may generate tokens that belong to different translations due to the independence assumption. For example, a French sentence [*Merci* .] can be translated into English sentences [*Thanks* .] or [*Thank you* .]. However, NAT models may generate sequences like [*Thanks you* .] or [*Thank* .]. This phenomenon is called the multimodality problem of NAT (Gu et al., 2018a). From another aspect, this is similar to a one-to-many multilingual NMT model, in which the model is trained to translate the source language into multiple target languages, as also experimented with by Zhou et al. (2020). If no indication about the desired target language is given to the model, it may generate sentences in arbitrary target languages. However, the indications in multilingual NMT models like using a tag token to claim the desired target language (Johnson et al., 2017) is a relatively strong constraint in AR approaches, while it is more difficult for NAT models to respect such constraints like target sentence length.

In this chapter, similar to Chapter 2, we briefly introduce methods in NAT and consider the following aspects:

- Model architecture
- Objective function
- Learning paradigm
- Knowledge distillation of training data

A more detailed review can be found in (Xiao et al., 2022b) and Gu and Tan (2022),[1] from which we borrow inspirations for this chapter. We also introduce methods using NAT with augmented sources, as in Chapter 3 for AR models.

4.1 . Non-autoregressive Translation Architectures

Many NAT model architectures have been proposed in recent years. We only briefly introduce some examples in this section, with a particular focus on iterative refinement-based models.

4.1.1 . Fully Non-autoregressive Model

The definition of NAT indicates that it is fully non-autoregressive, generating all target tokens of a translation at once with a time complexity of

[1] https://nar-tutorial.github.io/acl2022/

$O(1)$. Fully NAT models mainly follow the Transformer model. In order to predict multiple tokens, the decoder needs to be initialized with an input sequence to indicate the number of tokens it is required to generate. Therefore, most NAT models incorporate a length predictor, which aims to estimate the target sentence length based on the source. The length predictor also helps NAT models to alleviate the multimodality problem by providing an estimated length that includes certain contextual target information. For instance, it is much less likely that a NAT model will generate [*Thank* .] if given a length of 2 tokens, as *Thank* comes from [*Thank you* .], which is of length 3.

Gu et al. (2018a) implement the length predictor as a *fertility predictor*, which takes the encoder output hidden states as input and predicts a fertility value for each input position, indicating how many target tokens should be generated from each source token. The decoder input length is thus the sum of all fertility values. The encoder of a fully NAT model is identical to the Transformer encoder. For the decoder, since the AR property is removed, the causal mask in the decoder self-attention layer (*cf.*, Section 2.2.1) is no longer necessary. However, contrary to AR models, which use teacher forcing during training (*cf.*, Section 2.3.1), the decoder input for NAT models requires an initial input sequence. Gu et al. (2018a) propose to copy the source token embeddings as decoder input by following the fertility information or uniformly, where decoder input t is the copy of the $[St/T]$-th encoder input. To enhance the positional information, Gu et al. (2018a) add a positional attention layer in each decoder layer, in which the query and key of Equation (2.5) are the positional encoding vectors, and the value is the decoder states, respectively.

During inference, Gu et al. (2018a) propose noisy parallel decoding in which they sample multiple fertility predictions instead of taking the candidates with the best probability. In this way, they can generate multiple translations of different lengths. However, the noisy parallel decoding further requires an AR model to score and re-rank the translations to find the best candidate. An alternative to the noisy parallel decoding is the length beam decoding (Guo et al., 2019), which uses the best length prediction L and expands it to $[L - k, L + k]$ with a beam size of $2k + 1$.

4.1.2 . In Between Autoregressive and Non-autoregressive

Several approaches have been proposed to mix AR and non-autoregressive translations to better capture the dependencies on the target side. Semi-autoregressive translation (SAT) model is AR at the sequence level but can produce a group of tokens in parallel. Wang et al. (2018a) consecutively group K target tokens into one element by concatenating the K vectors into a higher dimension, then compute the probability of the target sequence

as:

$$P(\mathbf{e}|\mathbf{f}) = \prod_{i=1}^{[(T-1)/K]+1} P(G_i|G_1 \dots G_{i-1}, \mathbf{f}), \tag{4.2}$$

where K is the number of tokens to predict in parallel, and G_i is the concatenated vector of $e_{(i-1)\cdot K+1}, \dots, e_{i\cdot K}$. The model can therefore predict K tokens at once. SAT model lies in between AR models and fully NAT models. The model becomes a conventional AR model when $K = 1$ and turns into a fully NAT model like the one presented in (Gu et al., 2018a) when K is greater than the maximum sequence length. Wang et al. (2018a) also propose to relax the strict decoder causal mask by letting tokens in a group $e_{(i-1)\cdot K+1 \le j \le i\cdot K}$ be able to attend to all tokens in the group G_i.

Stern et al. (2018) propose blockwise parallel decoding, similar to Wang et al. (2018a), but dynamically select the number of tokens predicted in parallel. At each time step, they first generate k consecutive tokens in parallel, then select the first $\hat{k}$ tokens that match the AR model's output and keep the $\hat{k}$ tokens for the next step.

Contrary to SAT, the locally AR translation model (LAT, Kong et al., 2020) is non-autoregressive at the sequence level but generates a short sequence of tokens at each position autoregressively. It uses a small RNN-based model to predict a short sequence of K tokens at each prediction position. The predicted sequence is supposed to be fluent and well-translated since the AR module measures the local target dependency. However, LAT requires an extra merging algorithm to align and merge the output sequences to obtain the final translation. Kong et al. (2020) compute the Longest Common Subsequence (LCS) between two adjacent generated sequences, select matched tokens, and resolve conflicts for other tokens by taking the ones with a higher probability. If no LCS match is found, they simply concatenate the two sequences.

4.1.3 . Iterative Refinement Based Models

Instead of producing all tokens in parallel only once, several methods have been proposed to iteratively adjust the translation to obtain better performance by conducting multiple decoding rounds. We introduce some typical examples below.

Conditional Masked Language Model

The Conditional Masked Language Model (CMLM, Ghazvininejad et al., 2019) borrows the idea from the masked language model (MLM) training of large pre-trained language models (Devlin et al., 2019; Conneau and Lample, 2019). MLM masks out certain tokens $\mathbf{e}_{mask}$ in an input sequence by

replacing them with a special [mask] token and aims to recover the masked tokens on the output by conditioning on the other unmasked tokens $\mathbf{e}_{obs}$ and predicting the individual probabilities $P(e|\mathbf{e}_{obs})$ for each $e \in \mathbf{e}_{mask}$. CMLM extends this idea under the translation scenario by conditioning on both the source sentence and the unmasked tokens, thus computing the probability $P(e|\mathbf{f}, \mathbf{e}_{obs})$. The model architecture again follows the Transformer model and removes the decoder causal mask. During training, CMLM randomly masks certain tokens. The ratio of masked tokens is uniformly randomly sampled between 0 and 100%. This differs from MLM models, which use a constant masking rate of 15%. Similar to Gu et al. (2018a), CMLM needs to know the target sequence length to place [mask] tokens. Once again, CMLM follows Devlin et al. (2019) to prepend a special [length] token to the source sentence and use the hidden states of this [length] token from the encoder output to estimate the length of the target sentence.

The decoding procedure of CMLM does not finish in one single decoding step. Instead, Ghazvininejad et al. (2019) propose a mask-predict method to generate the final output within a constant number of cycles. At the initial iteration, CMLM predicts the target sentence length from the encoder and initializes the decoder with all [mask] tokens to generate all tokens together. For the later iterations, it selects n tokens with the lowest output probabilities, masks these tokens with [mask], then restarts another iteration to predict the masked tokens until the predefined number of iteration I is reached. The number of tokens n to be masked varies across iterations with a linear decay:

$$n = L \cdot \frac{I - i}{I}, \tag{4.3}$$

where L is the predicted target length, and i is the iteration number. The decoding methods, such as mask-predict, which repeatedly adjust the output translations, are also known as *iterative refinement* (Lee et al., 2018). CMLM is able to perform NAT in constant decoding rounds, despite the length of generated sentences, and delivers better translation quality compared to fully NAT models. To explore more candidates during inference, CMLM can sample multiple target lengths or use the top-k predictions to generate several translations. The translation with the best average log probability is finally picked.

Insertion Transformer

CMLM must know the target sequence length in advance to predict tokens, and the length cannot be changed once it is predicted. The Insertion Transformer (Stern et al., 2019) instead generates translations via insertion operations. It jointly produces pairs of a target token and the corresponding location in the sequence to insert the token. The target sequence always

contains a begin-of-sentence and an end-of-sentence token, and the actual sequence length of a target with n tokens is thus $n+2$. The number of possible insertion slots is thus $n+1$. Insertion Transformer first concatenates the decoder output hidden states to the left and right of each insertion slot as one vector to build the slot representation. The slot matrix $H \in \mathbb{R}^{(n+1)\times d}$ containing $n+1$ slot representations of dimension d is then passed through the standard output linear projection layer to obtain the $n+1$ *content-location* logits of dimension $|V|$, where $|V|$ is the target vocabulary size. Insertion Transformer models the joint content-location probability by flattening the $n+1$ logits in a single vector of size $(n+1)\times|V|$, then performs a softmax operation to obtain the joint probability distribution over all possible tokens for all slots. It can also compute the content probability of each location by separately performing the softmax operation for each location to generate tokens for multiple locations simultaneously. However, Insertion Transformer cannot generate all tokens in one single step, as it can only produce at most one token for each insertion slot. It has to iteratively generate tokens to increase insertion slots for the next iteration. The inference time complexity of the Insertion Transformer is thus logarithmic.

Levenshtein Transformer

The masking step in CMLM decoding, which masks tokens with the lowest probabilities, can be seen as a kind of deletion operation to cross out unconfident predictions. The Levenshtein Transformer (LevT, Gu et al., 2019) can be viewed as a generalization of CMLM and the Insertion Transformer, as it models possible insertion and deletion operations on target sequences. LevT is able to edit the target sequence during generation to arbitrarily change the target length and iteratively refine the output sequence in a dynamic way.

Placeholder Insertion Unlike the Insertion Transformer, which models the joint probability of insertion slot and content token, LevT separates it into two steps. The first step predicts how many tokens should be inserted for each possible insertion slot. Similar to the Insertion Transformer, LevT also concatenates the decoder state to the left and right of each slot as a single vector, then passes it through a placeholder classifier, which simply predicts the number of tokens to insert for each insertion slot. LevT then modifies the initial target sequence by adding predicted corresponding placeholder tokens to each slot, as illustrated in Figure 4.1. Thanks to this placeholder insertion step, LevT is able to insert multiple tokens in one slot within the same decoding step.

Token Prediction Once placeholders are inserted, LevT then performs similarly to CMLM to predict an output token for each placeholder, conditioned on the source and existing target tokens.

Deletion After token prediction, LevT incorporates a deletion operation in which the target sequence decoder states are passed through a deletion classifier to decide whether each token should be deleted. Figure 4.1 also illustrates the deletion operation.

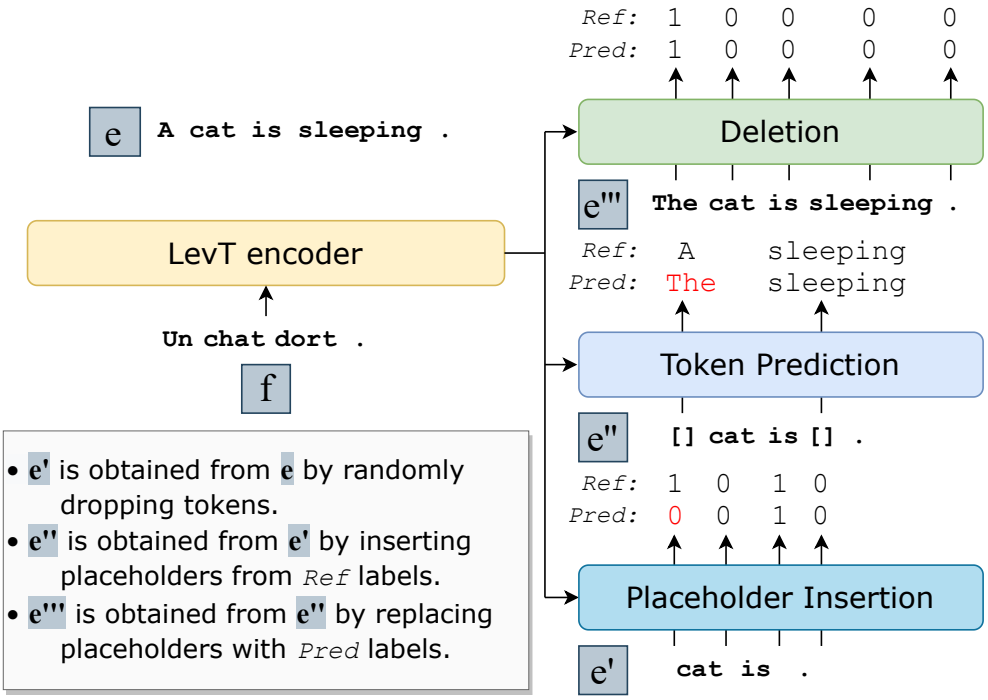


Figure 4.1: A complete training step for LevT. We omit the begin-of-sentence and end-of-sentence tokens which are always present in each target sequence.

Training With the above operations, LevT is able to make edits to the target sequence by iteratively performing the insertion and deletion operations. It does not need an extra length predictor as the length of the target sequence is dynamically adjusted during generation. However, training the LevT model requires proper training data for each operation. Specifically, the placeholder insertion module needs to learn how to insert tokens, while the deletion module needs to detect errors from a target sequence. All three operations take the same decoder architecture but only differ in the output linear classification layer. However, since the decoder input for each operation is different, each operation needs to recompute the input through

the decoder. The decoder parameters for each operation can be shared or separately trained.

The training process is performed sequentially, as illustrated in Figure 4.1. Similar to CMLM, Gu et al. (2019) first randomly drop tokens from the reference $\mathbf{e}$ to build the input sequence $\mathbf{e}'$ for placeholder insertion. The only difference is that CMLM replaces the dropped tokens with a mask token and keeps the target sequence length unchanged, while LevT actually deletes those tokens, yielding a subsequence $\mathbf{e}'$ of $\mathbf{e}$. The reference placeholder numbers for each position can be obtained by computing the edit distance between $\mathbf{e}'$ and $\mathbf{e}$. The reference for the token prediction operation can also be computed at the same time. The input $\mathbf{e}''$ thus must always be the oracle placeholder insertions. However, for the deletion operation, Gu et al. (2019) do not generate any synthetic sentences but directly use the prediction made by token prediction operation as input. Once again, reference deletion labels are obtained by computing the edit distance between $\mathbf{e}'''$ and $\mathbf{e}$.

Inference During inference, LevT starts with an empty target sequence and generates the translation by alternatively performing deletion and insertion (placeholder insertion + token prediction) operations until convergence or a maximum decoding round is reached. In the first iteration, the deletion is omitted as nothing can be deleted from an empty sequence. The iterative refinement converges when the input and output of one iteration are the same, as it predicts nothing to delete and nothing to insert in an iteration, or it encounters a loop where the deleted tokens are reinserted in a single iteration.

4.2 . Training Objectives

Most NAT models are trained with the standard cross-entropy (CE) loss as:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{T} \log P(e_i|\mathbf{f};\theta), \tag{4.4}$$

where the probability $P(e_i|\mathbf{f};\theta)$ is computed independently for each token given the source $\mathbf{f}$. CMLM and LevT measure the prediction CE loss only on the masked segments but not the entire sequence. These models only predict tokens for positions where the input is a mask or a placeholder. Therefore, computing loss on other unmasked positions is meaningless. The CE loss is also the objective function used to train placeholder insertion and deletion for LevT. The CE loss measures the exact match at each token position, thus giving high penalties to output sequences with small order shifts. However, it is very difficult for NAT models to correctly predict the

exact token at each position due to the multimodality issue as several word orders may be equally correct. To alleviate this issue, several approaches propose to consider latent alignments between the output target sequence and the reference using alternative training objectives, aiming to ease the strict match at each position of the CE loss.

4.2.1 . Connectionist Temporal Classification

Libovický and Helcl (2018) apply the Connectionist Temporal Classification (CTC, Graves et al., 2006) loss for NAT. CTC is first applied in automatic speech recognition systems. Assuming an input sequence $\mathbf{x}$ and a reference sequence $\mathbf{y}$, in which $\mathbf{x}$ is longer than $\mathbf{y}$ ($|\mathbf{x}| \geq |\mathbf{y}|$), CTC aims to label the input sequence $\mathbf{x}$ with tokens in $\mathbf{y}$ and empty tokens and find all valid monotonic alignments that can recover the reference sequence $\mathbf{y}$ after removing empty tokens and collapsing consecutive repeated tokens. CTC suits speech recognition tasks well, as the input sequence of phonemes is always longer than the output sequence of lexical tokens, and many phonemes may align to the same token or to nothing when there are pauses in the speech. For NAT, CTC allows NAT models to freely choose the best prediction regardless of strict position constraints. As CTC allows predicting empty tokens, the number of possible combinations of reference and empty tokens of an output length L and a reference length T is thus $\binom{L}{T}$. CTC is able to efficiently compute the CE loss summed over all possible alignments using dynamic programming:

$$\mathcal{L}_{\text{CTC}} = -\sum_{\mathbf{a}\in\beta(\mathbf{e})}\prod_{i=1}^{L} P(a_i|\mathbf{f};\theta), \tag{4.5}$$

where $\mathbf{a}$ is a possible alignment, and $\beta(\mathbf{e})$ refers to all possible alignments. However, unlike speech signals, the source sentences in translation may be shorter than the target. To match the CTC requirement, Libovický and Helcl (2018) upsample the source sentence k times of length S by slicing each encoder output state into k vectors. The obtained sequence of length $L = k \cdot S$ is used to initialize the decoder. In practice, NAT models often use $k = 2$ or $k = 3$.

CTC-based model (Saharia et al., 2020; Gu and Kong, 2021) is now one of the state-of-the-art models for fully NAT systems.

4.2.2 . Aligned Cross-entropy

Ghazvininejad et al. (2020) focus on penalizing lexical errors, which indicate wrong tokens, rather than word order errors. The latter refers to correct tokens appearing in the wrong positions. Given a reference sequence $\mathbf{e} = e_1, \ldots, e_T$ and a predicted sequence of probability distributions

$\mathbf{P} = P_1, \ldots, P_L$, Ghazvininejad et al. (2020) aim to find a monotonic alignment between these two sequences $a : \{1, \ldots, T\} \rightarrow \{1, \ldots, L\}$ which minimize the aligned cross-entropy (AXE) loss:

$$\mathcal{L}_{AXE} = -\sum_{i=1}^{T} \log P_{a(i)}(e_i) - \sum_{k \notin a} \log P_k(\epsilon), \tag{4.6}$$

where ϵ is an empty token allowed during training but removed in the output translation. The first term is an aligned cross-entropy between $\mathbf{e}$ and $\mathbf{P}$, and the second term is a penalty for unaligned predictions. The search for the optimal alignment is performed using dynamic programming.

4.2.3 . Order-agnostic Cross-entropy

CTC and AXE losses focus on the *monotonic* alignment between prediction and reference sequences. Du et al. (2021) extend the AXE loss to allow reordering in the alignment to find the overall best possible alignment. They propose the Order-agnostic cross-entropy (OAXE) loss, which almost removes word order errors and focuses on lexical matching. OAXE assumes the prediction sequence has the same length as the reference, similar to CE loss. However, instead of measuring the standard CE loss, OAXE aims to find a permutation of the reference tokens so that the reordered reference sequence can obtain a minimum CE loss:

$$\mathcal{L}_{OAXE} = \arg\min_{O^i \in \mathbf{O}} (-\log P(O^i|\mathbf{f})), \tag{4.7}$$

where O^i is a possible reordering or permutation of the original reference, $-\log P(O^i|\mathbf{f})$ is the CE loss of a reordered reference. Du et al. (2021) apply the Hungarian matching algorithm (Kuhn, 1955) to efficiently search for the optimal permutation.

4.3 . Learning Paradigms

Directly learning to generate the complete translation is a difficult task for NAT. Some works have studied training NAT models with a better curriculum so that models gradually learn to produce more and more complicated sequences.

4.3.1 . Glancing Transformer

Qian et al. (2021) propose the Glancing Transformer (GLAT). The idea is similar to Mihaylova and Martins (2019), who perform two-pass decoding to mix model predictions and ground-truth tokens. GLAT aims to reveal

certain reference words in the decoder input to help the model focus on generating segments instead of the entire sentence. It initializes the decoder input by a uniform copy of the encoder output states for the first decoding round. In this step, no target token is revealed to the decoder. GLAT then computes the distance $d(\hat{\mathbf{e}}, \mathbf{e})$ between the predicted sequence $\hat{\mathbf{e}}$ and the reference $\mathbf{e}$ and randomly selects $S = \lambda d(\hat{\mathbf{e}}, \mathbf{e})$ reference tokens, with λ a hyperparameter. The selected tokens are then revealed in the decoder input by replacing the corresponding encoder output states with the target token embeddings. The mixed decoder input sequence is then passed through a second decoding pass to compute losses only for unrevealed segments and update the model parameters.

The number of revealed tokens is thus dynamically adjusted through the training process, guiding the model to learn from an easier task to a harder one. At an early stage, the model is weak and will generate predictions that have a high distance to the reference. Therefore, more ground-truth tokens are revealed, so the model only focuses on predicting small segments with a larger context. As training proceeds, the model becomes more powerful and predicts translations that are closer to the reference. Therefore, fewer tokens are revealed, and the model is trained to predict longer segments.

4.3.2 . Multi-granularity Training

Ding et al. (2021a) propose progressive multi-granularity training, which is based on the same general idea as GLAT, starting with easier training examples and then more difficult ones. They break down bilingual sentences into parallel words and phrases. The training curriculum is simply first training NAT models on words, then phrases, and finally sentences.

4.3.3 . Pre-training

Large pre-trained language models benefit AR NMT models (Conneau and Lample, 2019; Edunov et al., 2019; Clinchant et al., 2019; Weng et al., 2020; Yang et al., 2020a; Zhu et al., 2020; Yang et al., 2020b). Several studies have also explored how NAT benefits from pre-trained language models.

Guo et al. (2020) use two BERT models, one for the encoder initialization and the other for the decoder. They add two feed-forward networks as an adapter layer (Bapna and Firat, 2019) in each BERT encoder layer and a cross-attention layer in each decoder layer to perform CMLM and show improvements with respect to CMLM models trained from scratch on both high-resource and low-resource language pairs. Su et al. (2021) use a single BERT model and add a conditional random field output layer to better model the target dependency. Their model only contains an encoder without a decoder.

Li et al. (2022) pre-train a multilingual model by jointly training on MLM task for the encoder and CMLM task for the decoder. The intuition is to pre-train a multilingual encoder and a *bidirectional* decoder to benefit from bilingual and monolingual data of various languages. A pre-trained bidirectional decoder may achieve better results on AR translation (Conneau and Lample, 2019). As illustrated in Figure 4.2, they first create mixed-language (MXL) source sentences by randomly replacing source tokens, which have aligned tokens on the target side, with the aligned words from a third language. The aligned words are obtained from a multilingual translation dictionary. At the same time, target tokens aligned to the replaced source tokens are masked with a `[mask]` token. After this, they also apply random dual-masking to both source and target sequences, similar to MLM. The resulting data consists of a mixed-language source sequence with masks on its original language tokens and a target sequence with masks from both steps. Li et al. (2022) also use monolingual data, setting the initial target sentence identical to the source. The source sequence passes through the encoder and is trained on the MLM task like BERT. The target sequence passes through the decoder like CMLM. Both tasks are jointly trained with a combined loss during the pre-training stage. The pre-trained model is suitable for fine-tuning on both standard AR translation and NAT with the CMLM approach and obtains performance gains for both tasks under both low-resource and high-resource scenarios.

parallel	[fr] Je rentre à la maison	[en] I return home
MXL & mask	[fr] 我 rentre à la maison	[en] [mask] return home
dual-mask	[fr] 我 rentre à la [mask]	[en] [mask] [mask] home

Figure 4.2: Masking procedures of Li et al. (2022). In the first step, the French word "Je" is aligned to "I" in English and to "我" in Chinese. Therefore, "Je" is replaced with "我", and "I" is masked. In the second step, tokens from both sentences are randomly masked.

4.4 . Knowledge Distillation

Knowledge Distillation (Hinton et al., 2015) is first proposed to distill knowledge from a strong teacher model into a relatively weak student model. The student model is trained to match the output probability distribution of the teacher model by minimizing the CE loss and the distance between the output probability distributions of the teacher and student model. Sequence-level knowledge distillation (KD, Kim and Rush, 2016) works differently. The teacher model translates all training source sentences $\mathbf{f}$ into the target language via beam search. The obtained higher-quality translation $\hat{\mathbf{e}}$ is called

distilled data. Student models are then trained on the new dataset $\hat{D} = \{(\mathbf{f}, \hat{\mathbf{e}})_i, i = 1, \ldots, N\}$ by considering the distilled data $\hat{\mathbf{e}}$ as reference. In this thesis, we abuse the term KD to denote sequence-level knowledge distillation since we are not concerned with the original knowledge distillation. KD is almost similar to the so-called "forward translation" or "self-training" (Zhang and Zong, 2016) as they all use translated data as synthetic references to train new models.

KD is first applied to NAT models by Gu et al. (2018a), who train the NAT models only with KD data and found that KD significantly improves the NAT performance. Since then, KD has been almost the default technique applied to most NAT systems. Several works have targeted analyzing why KD can improve NAT models. Zhou et al. (2020) show that KD reduces the complexity of training data, which is also observed by Crego and Senellart (2016). KD serves as a translation selection process by generating only one of the possible translations with increased consistency. Therefore, KD helps to mitigate the multimodality problem of NAT by providing more consistent translations. For example, if [*Merci* .] is always translated into [*Thank you* .] during training, then NAT models will be less likely to generate tokens that belong to the sentence [*Thanks* .]. Zhou et al. (2020) also find that distilled data is more monotonic with respect to the source sentence. In other words, the target tokens are more likely to appear in the same relative order as the source tokens, and distilled data contains fewer reorderings than the real reference. Xu et al. (2021b) further demonstrate that reducing the amount of reordering helps NAT models to better learn the alignment between source and target tokens. They also find that if source tokens are pre-reordered to match the target order (Ran et al., 2021), KD makes no improvements compared to real parallel data. Lastly, Xu et al. (2021b) also find that KD reduces the lexical diversity on the target side, which also eases the learning of source-target alignment.

However, KD is time-consuming and strongly affected by the capacity of teacher AR models. Errors made by the AR models will be propagated to NAT models during training, especially for low-frequency words, which the AR models often fail to properly generate, as pointed out by Ding et al. (2021b). They study how to expose the raw data lexical knowledge to NAT models during training by incorporating extra alignment information between source tokens and real target tokens in order to help NAT models with better lexical choices. Huang et al. (2022) aim to correct the training-inference mismatch in CMLM by replacing certain tokens in the initial target sequence $\mathbf{e}_{obs}$ with predictions made by the model itself in another decoding round, starting with an all-masked input to better simulate the inference scenario. This strategy is similar to the scheduled sampling method (Mihaylova and Martins, 2019), as introduced in Section 2.3.1. It exposes the model to

repeated tokens produced by the model in $\mathbf{e}_{obs}$ during training so that the model can learn to refine its prediction, therefore improving the performance when trained on both real and distilled data.

4.5 . Non-autoregressive Translation with Augmented Sources

4.5.1 . Lexical Constraints for Non-autoregressive Translation

Most existing works studying the integration of augmented sources into NAT focus on incorporating lexical constraints for NAT. Surprisingly, all these works are based on the LevT model (*cf.*, Section 4.1.3), which supports edit operations during the generation process.

Susanto et al. (2020) first experiment by simply using the lexical constraints as the initial sequence for the LevT inference. Instead of the standard LevT, which always starts with an empty target sequence, the initial target input to the decoding is a sequence containing all constraints. This approach does not change the training of LevT and is only applied for inference. Only initializing the decoding with constraints implies soft constraints, as the model may delete these tokens during decoding. It is also possible to perform hard constraints by disallowing deleting constraint tokens and preventing insertions in the middle of consecutive constraint tokens.

Xu and Carpuat (2021a) propose EDITOR, which replaces the deletion operation in the LevT model with a repositioning operation. It simultaneously deletes a token and inserts it at another position. EDITOR also takes the lexical constraints as initial input. However, it allows the initial constraints to be provided in arbitrary order as the repositioning operation is able to adjust constraints to better fit a translation. Xu and Carpuat (2021b) further incorporate a rule-based morphological inflection module, allowing the initial constraints to be in lemma format without explicit inflection.

Zeng et al. (2022) pay particular attention to low-frequency constraints as they are likely to be deleted with soft constraints decoding. They simulate training with rarer words when generating initial subsequences for the LevT. They first sample several tokens from the reference as pseudo constraints based on TF-IDF scores (term frequency-inverse document frequency). These tokens are then prevented from being deleted when generating the initial subsequence. Zeng et al. (2022) further provide the alignment information of target constraints (pseudo constraints for training and real constraints in inference) with the aligned source tokens by adding an extra alignment embedding to the source tokens. The alignment embedding values indicate for each source token the relative order index of the aligned target constraints.

The above approaches focus on lexical constraints for translation, while Wan et al. (2020) explore lexical constraints for automatic post-editing. They propose a multi-source LevT with a second encoder taking an external MT output as input, and they initialize the decoder with target constraints, as proposed by Susanto et al. (2020). They also explore a single-encoder approach where the source is processed as Dinu et al. (2019) to incorporate target constraints and initialize the LevT decoding by MT output.

4.5.2 . Non-autoregressive Translation for Word-level Quality Estimation

Ding et al. (2021c) adapt the LevT to perform the word-level quality estimation. The task is to predict: (a) whether each predicted target word is correct or not; (b) whether there are missing words between each pair of predicted words. These tasks perfectly suit LevT: predicting target word correctness is somewhat equivalent to the deletion operation and predicting missing words is similar to the placeholder insertion module. Therefore, Ding et al. (2021c) fine-tune a pre-trained LevT model specifically on these tasks with some task adaptations and achieve good performance when the fine-tuning data is limited.

4.6 . Conclusion

In this chapter, we reviewed NAT from various aspects like model architectures, training objectives, learning paradigms, the use of KD, and NAT with augmented sources. However, there are still some approaches like NAT with latent variable and variational auto-encoder (Ma et al., 2019; Shu et al., 2020; Lee et al., 2020; Bao et al., 2022) that are not covered as we are not concerned with these methods in this thesis. NAT is still a new research area in NMT, as many unresolved problems exist. The key problem is the conditional independence assumption that makes NAT models difficult to model the target dependencies. Besides, despite several analyses about the effectiveness of KD for NAT, we think this issue still requires further studies to thoroughly understand why real references perform worse than KD. We believe that the eventual purpose of these studies is to get rid of KD and solely use real references to train NAT models.

In this chapter, we paid particular attention to the LevT-based approaches, as these models can perform edits to an initial target sequence during the translation process. We will also explore LevT-based methods to synchronize bitexts by editing one of the two languages in Chapter 8.

5 - Disentangling Mixed-Language Sentences with Dual Decoding

5.1 . Introduction

As introduced in Chapter 1, one possible way to perform bilingual writing is to write a sentence in mixed-language (MXL), then translate the MXL sentence into monolingual sentences in both component languages. Showing texts in both a foreign language (L2) and the users' mother tongue (L1) may help the users to verify the composed texts. Besides, the existing text segments in an MXL sentence produced by the users with high confidence can provide valuable contextual information to control the translation into L2 and produce better translations. Therefore, in this chapter, we study the task of translating MXL sentences into monolingual sentences in both languages.

MXL sentences are similar to Code-Switching, which denotes the alternation of two languages within a single discourse, utterance, and sometimes words (Poplack, 1980; Sitaram et al., 2019). We mostly consider sentence-level language mixing in this thesis. Code-Switching is a common and essential communicative phenomenon that occurs in multilingual communities during informal spoken and written interactions. It generally consists of short inserts of a secondary language that are embedded within larger fragments in the primary language. Code-Switching is still an understudied domain in natural language processing (Aguilar and Solorio, 2020), and most work to date has focused on token-level language identification (LID, Samih et al., 2016) and language modeling for automatic speech recognition (Winata et al., 2019). More tasks have been considered lately, such as named entity recognition (Aguilar et al., 2018), part-of-speech tagging (Ball and Garrette, 2018), or sentiment analysis (Patwa et al., 2020). Even though similar to Code-Switching, the MXL sentences we deal with in this chapter are slightly different, as Code-Switching often refers to informal texts while MXL sentences are relatively more formal. Moreover, we intend to study interactive systems under a simulated situation. In our case, MXL sentences are artificial and not realistic. Therefore, we distinguish the term MXL sentences from Code-Switching and use MXL sentences in this chapter.

Little attention has been paid to the task of machine translation (MT) for MXL input. Johnson et al. (2017) mention using a multilingual neural machine translation (NMT) system to translate an MXL sentence to a third target language by showing only one example. Dinu et al. (2019) perform some sort of MXL translation in their replacement setting, as introduced in

Section 3.4.2. However, they mainly focus on translating target terminologies, thus only translating the MXL sentence into one of the two languages. Menacer et al. (2019) release a parallel Arabic-English MXL corpus dedicated to MT applications. This MXL data is extracted from the United Nations data: while translations into English are readily available, the purely Arabic side of the corpus is obtained using Google Translate to fill the missing Arabic bits. Gupta et al. (2021) manage to generate synthetic MXL data from parallel bilingual corpora. They first fine-tune a multilingual BERT model on an MXL dataset to perform token-level LID. To create synthetic MXL sentences, they use the fine-tuned LID model to predict possible positions to inject target words, then replace the source words with the aligned target ones. Very recently, a shared task on MT in Code-Switching settings has been held (Chen et al., 2022).

As discussed in Section 3.5.1, the multi-source way (Och and Ney, 2001; Schwartz, 2008; Zoph and Knight, 2016) to handle this task of translating one source language into two target languages generates the first translation in a target language l_{T_1}, which, once revised, can be used in conjunction with the original source to generate the translation in the other language l_{T_2}. The expected benefit of this approach is to facilitate word disambiguation. However, this approach risks cascading errors to the second translation unless the translation in l_{T_1} is thoroughly revised before the second translation round. The advent of NMT technologies (Bahdanau et al., 2015; Vaswani et al., 2017) has made it possible to design multilingual models capable of translating from multiple source languages into multiple target languages (Firat et al., 2016a; Johnson et al., 2017).

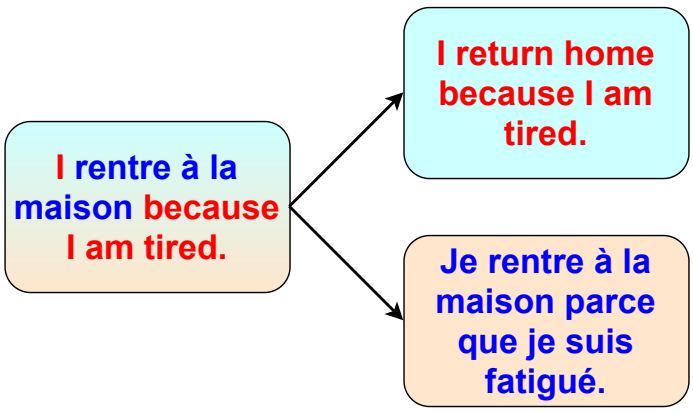


Figure 5.1: An example of simultaneously generating monolingual translations from an MXL sentence.

In this chapter, we conduct dual decoding to simultaneously translate an MXL input, denoted as a new language l_S, into both component languages, l_{T_1} and l_{T_2}. Figure 5.1 gives an illustration. While the same goal of generating

monolingual sentences from MXL inputs can be achieved with a multilingual system translating an MXL sentence independently into l_{T_1} and l_{T_2}, we expect several payoffs from a dual decoding:

- Obtaining two translations in l_{T_1} and l_{T_2} via a single decoding process;
- Better translation qualities in both languages;
- More consistent translations in l_{T_1} and l_{T_2} than if they are performed independently.

We also perform several analyses to better understand the task of translating MXL texts into monolingual sentences. Dinu et al. (2019) suggest that target tokens injected into the source sentence can be considered as lexical constraints. In our case, when simultaneously translating MXL texts into two languages, it can be considered that lexical constraints in both languages are present at the same time. Therefore, we expect the following "copy" constraint to be satisfied: *every word in the MXL source text should appear in at least one of the two outputs*. In the bilingual writing scenario, the copy constraint also indicates that texts composed by the users in either language should be preserved in the output translations.

Our main contributions are the following:

- We conduct a comparative study of architectures for MT of MXL input and propose a dual decoder model.
- We propose a synthetic MXL data generation method and demonstrate the artificial data's effectiveness in training NMT models.
- In our systems, the translation of MXL texts is almost as good as the translation of monolingual texts – a performance that regular systems are unable to match.
- MXL translation systems achieve a near-deterministic ability to recopy in the output target words found in the input, suggesting that they are endowed with some LID abilities.
- We are also able to obtain competitive results on the SemEval 2014 Task 5: L2 Writing Assistant with our model, which we see as one potential application area of MXL translation.

The contributions in this chapter are published in (Xu and Yvon, 2021a,b).

5.2 . An Architecture for Dual Decoding

In our setting, we consider the simultaneous translation of sentence $\mathbf{f}$ in source language l_S into two target sentences $\mathbf{e}^1$ and $\mathbf{e}^2$ in languages l_{T_1} and l_{T_2}. In this situation, various modeling choices can be entertained (Le et al.,

2020):

$$P(\mathbf{e}^1, \mathbf{e}^2|\mathbf{f}) = \prod_{t=1}^{T} P(e_t^1, e_t^2|\mathbf{f}, \mathbf{e}_{<t}^1, \mathbf{e}_{<t}^2) \tag{5.1}$$

$$P(\mathbf{e}^1, \mathbf{e}^2|\mathbf{f}) = \prod_{t=1}^{T} P(e_t^1|\mathbf{f}, \mathbf{e}_{<t}^1, \mathbf{e}_{<t}^2) P(e_t^2|\mathbf{f}, \mathbf{e}_{<t}^1, \mathbf{e}_{<t}^2) \tag{5.2}$$

$$P(\mathbf{e}^1, \mathbf{e}^2|\mathbf{f}) = \prod_{t=1}^{T} P(e_t^1|\mathbf{f}, \mathbf{e}_{<t}^1) P(e_t^2|\mathbf{f}, \mathbf{e}_{<t}^2), \tag{5.3}$$

where $T = \max(|\mathbf{e}^1|, |\mathbf{e}^2|)$, and we use placeholders whenever necessary. The factorization in Equation (5.1) implies a joint event space for the two languages and a computational cost we deem unreasonable. Instead, we resort to the second (*dual*) formulation that we contrast with the third one (*independent* generation) in our experiments. Taking $\mathbf{e}^1$ as an example, in dual decoding, the probability of e_t^1 is conditioned not only on the source $\mathbf{f}$ and previous outputs $\mathbf{e}_{<t}^1$ but also on the previous outputs of the other target sequence $\mathbf{e}_{<t}^2$. The probability of e_t^2 is computed accordingly. Note that we are also able to simulate other asynchronous dependency patterns, where each symbol e_t^2 is generated conditioned on $\mathbf{e}_{<t+k}^1$ and $\mathbf{e}_{<t}^2$, thus reproducing the *chained* model of Le et al. (2020). The asynchronous decoding is explored in Chapter 6.

5.2.1 . Dual Decoder Model

We implement Equation (5.2) with a synchronous coupling of two decoders sharing the same encoder. Our dual decoder model is inspired by Wang et al. (2019). Figure 5.2 illustrates this design. Compared to a standard Transformer, we use the same encoder, but we add a decoder cross attention layer in each decoder block to capture the interaction between the two decoders. We denote the output hidden states of the previous layer for each decoder as $\mathbf{H}_l^1$ and $\mathbf{H}_l^2$. The decoder cross-attention is computed as:

$$\begin{aligned} \mathbf{H}_{l+1}^1 &= \mathrm{MultiHead}(\mathbf{H}_l^1, \mathbf{H}_l^2, \mathbf{H}_l^2) \\ \mathbf{H}_{l+1}^2 &= \mathrm{MultiHead}(\mathbf{H}_l^2, \mathbf{H}_l^1, \mathbf{H}_l^1), \end{aligned} \tag{5.4}$$

where $\mathrm{MultiHead}$ is the multi-head attention described in Equation (2.5). For simplicity, we omit the other sub-layers (self-attention, encoder-decoder cross attention, feed-forward, and layer normalization). The two decoders are thus fully synchronous as each requires the hidden states of the other in each block to compute its own hidden states. Similar to the original decoder self-attention layer in the Transformer model (*cf.*, Section 2.2.1), we also apply the attention mask in the decoder cross-attention layer to prevent

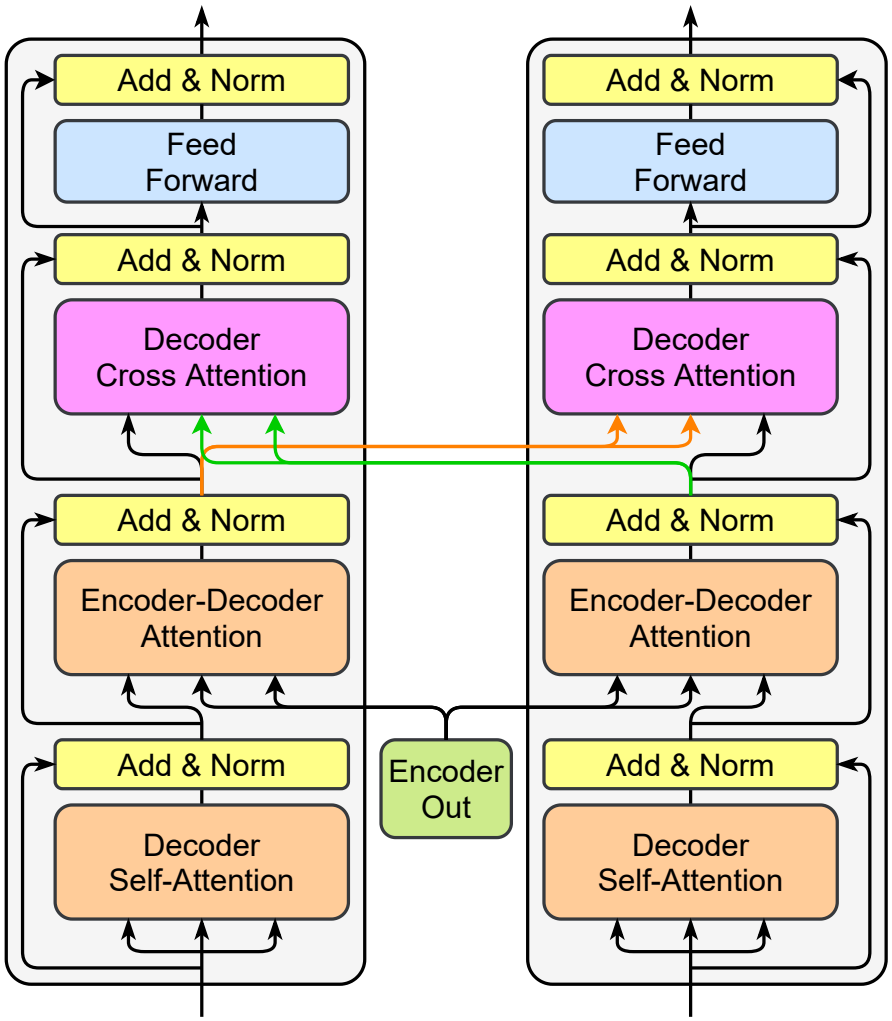


Figure 5.2: A graphical view of the dual decoder model.

the decoders from seeing future information from the other decoder. The decoder cross-attention can be inserted before or after the encoder-decoder attention block. Our preliminary experiments with these variants showed that they were performing similarly. We thus only report results obtained with the decoder cross-attention as the last attention component of a block, as illustrated in Figure 5.2.

5.2.2 . Synchronous Beam Search

Our decoding algorithm uses a dual beam search. Assuming each decoder uses its own beam of size k, the cross-attention between decoders can be designed and implemented in multiple ways: for instance, in each decoder layer, one can have each hypothesis in decoder 1 attend to all hypotheses in

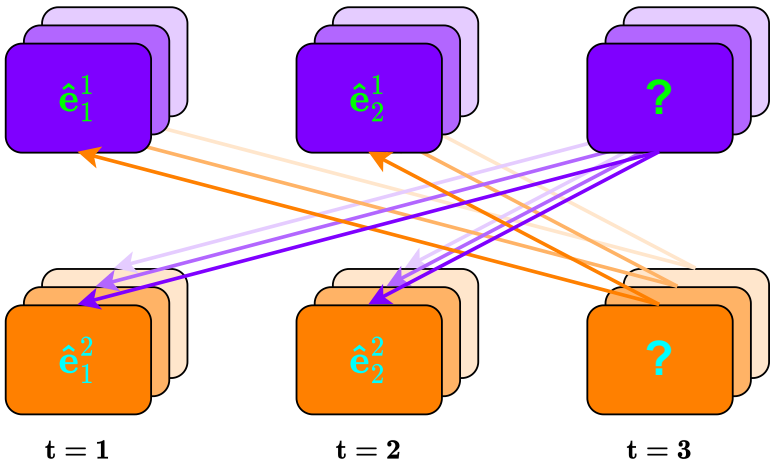


Figure 5.3: A graphical view of the dual beam search process.

decoder 2 and vice versa, creating k intermediate representations for each hypothesis. In this way, each hypothesis at each decoder layer yields k candidates, which will, however, create an exponential blow-up of the search space in the final decoder layers. Following Zhou et al. (2019), we only compute the attention between the 1st-best candidates of each decoder, the 2nd-best candidates of each decoder, etc. This heuristic ensures that the number of candidates in each decoder beam remains fixed, as illustrated in Figure 5.3. However, there is an added complexity since the ranking of hypotheses in each decoder beam evolves over time: the best hypothesis in decoder 1 at time step t may no longer be the best at time step $t + 1$. Preserving the consistency of the decoder states, therefore, implies recomputing the entire prefix representation for each hypothesis and each decoder at each time step, thus creating a significant computing overhead. In our experiments, the dual decoder is about twice as slow as the two independent ones.

We also explored other implementations where all candidate prefixes in one beam always attend to the best candidate in the other beam or the averaged hidden states of all candidates from the other beam. These variants deliver very similar results, as also found in (Zhou et al., 2019). For simplicity reasons, we use the first scheme in all experiments.

5.2.3 . Training

Training the dual decoder model requires triplets of instances comprising one source and two targets. Given a set of such examples $D = \{(\mathbf{f}, \mathbf{e}^1, \mathbf{e}^2)_i, i = 1 \ldots N\}$, the training maximizes the combined log-likelihood for the two tar-

get sequences:

$$\mathcal{L}(\theta) = \sum_{D} \left[\sum_{t=1}^{|\mathbf{e}^1|} \log P(e_t^1 | \mathbf{e}_{<t}^1, \mathbf{e}_{<t}^2, \mathbf{f}; \theta) + \sum_{t=1}^{|\mathbf{e}^2|} \log P(e_t^2 | \mathbf{e}_{<t}^2, \mathbf{e}_{<t}^1, \mathbf{f}; \theta) \right], \tag{5.5}$$

where θ represents the set of parameters.

5.3 . Mixed-language Data Generation

There exists large-scale bilingual parallel data in the community. However, parallel corpora aligned with natural code-switched or MXL data are very scarce (Menacer et al., 2019). Therefore, similar to Song et al. (2019), we generate artificial MXL sentences from regular translation data.

The Matrix Language Frame (MLF) theory (Myers-Scotton, 1997) defines the concept of *matrix* and *embedded* languages where the *matrix language* is the primary language that the sentence structure should conform to and notably provides the syntactic morphemes, while the influence of the *embedded language* is less evident and is mainly manifested in the insertion of content morphemes. We abuse the terms "matrix" and "embedded" language here, as we do not attempt to generate linguistically realistic code-switched sentences matching the constraints of the MLF theory. We use these terms in a much looser sense where the "matrix" language in an MXL sentence is the one that receives arbitrary insertions from the "embedded" language. This means that our artificial MXL sentences will contain insertions of unconstrained fragments containing both content and function words, which the theory would generally consider ungrammatical.

Taking a bilingual parallel corpus, we first compute word alignments between parallel sentences using `fast_align`[1] (Dyer et al., 2013). We then extract so-called *minimal alignment units* following the approach of Crego et al. (2005). These units correspond to small bilingual phrase pairs $(\mathbf{f}, \mathbf{e})$ extracted from symmetrized word alignments such that all alignment links outgoing from words in $\mathbf{f}$ reach a word in $\mathbf{e}$, and vice-versa. In Appendix A, we briefly introduce this extraction method.

For each pair of parallel sentences, we first randomly select one language as the matrix language; then randomly sample the number of replacements r to appear in a derived MXL sentence with an exponential distribution as:

$$P(r = k) = \frac{1}{2^{k+1}} \quad \forall k = 0, \ldots, R, \tag{5.6}$$

[1] https://github.com/clab/fast_align

where R is a predefined maximum number of replacements. We repeat the sampling process with a probability $P(\text{redo}) = \frac{1}{2^{R+1}}$ to ensure the probabilities sum to 1. We also make sure that the number of replacements does not exceed half of either the original sentence length from both parallel sentences, adjusting the actual number of replacements as:

$$n = \min(\frac{S}{2}, \frac{T}{2}, r), \tag{5.7}$$

where S and T are the lengths of the parallel sentences. We finally choose uniformly at random r alignment units and replace these fragments in the matrix language with their counterpart in the embedded language.[2] Figure 5.4 displays some examples of generated MXL sentences.

Matrix	In Oregon , planners are experimenting with giving drivers different choices .
$r = 1$	**Dans** Oregon , planners are experimenting with giving drivers different choices .
$r = 2$	**Dans** Oregon , **les planificateurs** are experimenting with giving drivers different choices .
$r = 3$	**Dans** Oregon , **les planificateurs** are experimenting **en offrant aux** drivers different choices .
Embedded	Dans l'Orégon , les planificateurs tentent l'expérience en offrant aux automobilistes différents choix .

Figure 5.4: Examples of generated MXL sentences when taking English as the matrix language and varying the number r of replacements of embedded French segments (in boldface). The alignment units are shown in color.

5.4 . Experiments

5.4.1 . Datasets

We conduct experiments with two language pairs: English-French (En-Fr) and English-Spanish (En-Es). We use WMT14 En-Fr[3] and WMT13 En-Es[4] data for MXL data generation and training NMT systems. We discard sentences that do not possess the correct language using the `fasttext` LID model[5] (Bojanowski et al., 2017). We use Moses tools (Koehn et al.,

[2] Our implementation of MXL data generation is open-sourced at https://github.com/jitao-xu/code-switch-gen

[3] https://www.statmt.org/wmt14/translation-task.html

[4] https://www.statmt.org/wmt13/translation-task.html

[5] https://dl.fbaipublicfiles.com/fasttext/supervised-models/lid.176.bin

2007) to normalize punctuation, remove non-printing characters and discard sentence pairs with a source/target ratio higher than 1.5, with a maximum sentence length of 250. We tokenize all WMT data using the Moses tokenizer.[6] For each language pair, we use a shared source-target vocabulary built with a joint Byte Pair Encoding (BPE) of 32K merge operations, using the implementation published by Sennrich et al. (2016c).[7] The actual data we use after preprocessing contains 33.9M parallel sentences for En-Fr and 14.5M for En-Es. The corresponding test sets are `newstest2014` (En-Fr) and `newstest2013` (En-Es). For each pair of sentences in the training data, we generate an MXL sentence using the method described in Section 5.3 and consider the generated MXL data as a new source language for translation. We also generate an MXL version of each test set, which we name as `mxl-newstest2014` and `mxl-newstest2013`, respectively, and use it for evaluation. Approximately half of the test sentences are mostly English with inserts in French or Spanish, and mostly French or Spanish with inserts in English for the other half.

5.4.2 . Machine Translation Systems

We implement the dual decoder model using `fairseq`[8] (Ott et al., 2019),[9] with a hidden size of 512 and a feed-forward size of $2{,}048$. We optimize with Adam (Kingma and Ba, 2015), set up with an initial learning rate of 0.0007 and an inverse square root weight decay schedule, as well as $4{,}000$ warm-up steps. All models are trained with mixed precision and a batch size of $8{,}192$ tokens for 300k iterations on 4 V100 GPUs. For standard Transformer models, we share the decoder input and output matrices (Press and Wolf, 2017; Inan et al., 2017). For dual decoder models, each decoder contains an input and an output matrix. Therefore, we share all four input and output decoder matrices.

We call the dual decoder models `dual`. To study the effectiveness of dual decoding, we compare `dual` models to several approaches:

- The `base` setting, where we train two separate Transformer models, one translating MXL into English, and the other translating MXL into Spanish or French.
- The `multi` setting, where we train one model able to generate either pure matrix or embedded language in the output. To this end, similar to a multilingual NMT model (Johnson et al., 2017), we add a tag

[6] https://github.com/moses-smt/mosesdecoder

[7] https://github.com/rsennrich/subword-nmt.

[8] https://github.com/pytorch/fairseq

[9] Our implementation is open-sourced at https://github.com/jitao-xu/dual-decoding.

System	Architecture	MXL Data
base-mono	2 Transformers	✗
bilingual	1 Transformer	✗
base	2 Transformers	✓
multi	1 Transformer	✓
indep	1 encoder + 2 indep decoders	✓
dual	1 encoder + 2 dual decoders	✓

Table 5.1: Summarization of the model architecture and data of systems used in this chapter. *2 Transformers* implies using two separate models, one for each direction. *1 Transformer* refers to one model that can perform translation in two directions. *2 indep decoders* indicates two independent decoders.

at the beginning of each MXL sentence to specify the desired target language. Taking En-Fr as an example, we add a <EN> tag for MXL-En and a <FR> tag for MXL-Fr. We use the combination of MXL-En and MXL-Fr data for training, which implies that each source side (MXL sentence) is duplicated in the training data once for each possible output.

- The indep setting, which extends multi by using one encoder and two separate decoders and training on the two output languages simultaneously. indep is a simplified multi-task version (Dong et al., 2015) of the dual model, implementing the *independent* decomposition of Equation (5.3) without the decoder cross-attention layer. For indep, the only interaction between the two outputs is thus *a shared loss function*: for each training instance, the loss function sums the two prediction terms for the embedded and the matrix language. The training data remains the same.

Note that since the number of replacements described in Equation (5.6) has $P(r = 0) = 0.5$, informing that half of the generated data remains unchanged, all systems described above also have the ability to translate monolingual source data in either direction. Table 5.1 summarizes the architecture and data of different models used in this chapter.

For comparison purposes, we also use the original parallel data to train two baselines systems:

- The base-mono setting, which consists of regular Transformer systems for the considered language pairs (En↔Fr and En↔Es), similar to base.
- The bilingual setting, which is one model capable of translating from and into both two languages, similar to multi. The selection of the desired target language relies on the same tagging mechanism as multi, which means that both types of models see exactly the same examples on the target side. The only difference with multi is thus

that this system does not see any MXL texts on the source side during training.

All resulting baseline Transformer models have exactly the same hyperparameters and use the same training scheme as `base` and `multi` approaches. Performance is measured with BLEU, computed with SacreBLEU (Post, 2018).

5.5 . Results and Analysis

5.5.1 . Main Results

We run tests using the artificial MXL datasets, as described in Section 5.4.1, as well as on the original test sets, in order to evaluate our models' ability to translate both MXL and monolingual sentences. Results are reported in Table 5.2, where we also report scores separately computed for the "matrix" and "embedded" parts of the test sets. We also report a "do-nothing" baseline (`copy`) which simply copies the input as output. As is obvious in the `copy` line, the "embedded" part contains mostly source language and corresponds to an actual translation task, whereas the "matrix" part mostly contains target words on the source side and is much easier to translate.

On the left part of this table, we see that the baseline systems, either with two (`base-mono`) or one single (`bilingual`) model(s), do better on monolingual test sets than their counterparts trained on MXL data (respectively `base` and `multi`). For both language pairs, the observed differences are in the range of 1-1.5 BLEU points. Conversely, when translating MXL sentences, models trained with MXL data perform significantly better than monolingual baseline models, which have never seen MXL on the source side.

Moreover, we note the marked differences between BLEU scores obtained by these models when the matrix language of the MXL source is the target language and when the embedded language is the target. In the former case, translation is nearly perfect, while in the latter case, they nonetheless use the little information available to improve the monolingual scores (about 1-1.5 BLEU points), nearly matching the performance of the baseline systems on monolingual sentences. As illustrated, for instance, for En-Fr, the `dual` model improves from 33.7 to 35.1; in the same condition, the `bilingual` system only improves by 0.4 points.

Among the models trained with MXL data, `multi` is the weakest, while the other three models achieve comparable performance. With joint training (`indep`) and dual decoding (`dual`), we can recover and even surpass the performance of the two separate systems used in the `base` condition with one single system. On the monolingual tests, these systems also match the multilingual baseline (`bilingual`) performance.

Test set	newstest2014				mxl-newstest2014			
Direction	En-Fr		Fr-En		MXL-Fr		MXL-En	
copy	-		-		50.0		46.5	
					2.9	93.8	2.9	93.4
base-mono	**37.6**		**35.2**		45.0		61.3	
	37.5	-	35.1	-	37.4	52.3	35.8	84.6
bilingual	36.1		34.0		46.3		59.4	
	35.8	-	33.8	-	36.4	55.4	34.2	82.7
base	36.5		34.1		67.4		**67.8**	
	36.3	-	34.1	-	37.5	95.3	35.5	97.4
multi	34.6		32.3		66.4		65.7	
	34.6	-	32.1	-	35.8	95.0	33.4	94.6
indep	35.9		34.0		67.3		67.7	
	35.9	-	33.7	-	37.1	95.5	35.1	97.4
dual	36.0		33.9		**67.5**		67.7	
	36.1	-	33.7	-	37.5	95.4	35.1	97.5
Test set	newstest2013				mxl-newstest2013			
Direction	En-Es		Es-En		MXL-Es		MXL-En	
copy	-		-		50.3		46.8	
					2.9	93.5	3.0	93.3
base-mono	**33.1**		**33.7**		39.8		57.7	
	33.2	-	34.0	-	32.8	46.3	34.6	79.6
bilingual	31.8		32.5		51.2		59.1	
	31.6	-	32.9	-	32.0	68.6	33.4	84.1
base	31.8		32.7		**66.9**		**66.5**	
	31.7	-	33.0	-	33.1	97.2	34.4	97.5
multi	30.8		31.5		66.4		64.7	
	30.9	-	31.8	-	32.1	97.2	33.0	95.1
indep	31.8		32.6		**66.9**		**66.5**	
	31.7	-	33.0	-	33.0	97.4	34.4	97.5
dual	31.8		32.3		66.8		66.4	
	31.4	-	32.6	-	32.8	97.3	34.1	97.4

Table 5.2: BLEU scores when translating the monolingual `newstest` data and the synthetic `mxl-newstest` data for two language pairs En-Fr and En-Es. `copy` refers to a do-nothing baseline that simply copies the input. Small numbers contain BLEU scores computed separately when the target language is the embedded language (left) and the matrix language (right) in MXL source. For the monolingual tests (left part), these correspond to scores computed on the same sentences that are also included in the MXL test sets.

5.5.2 . Preserving Copy Constraint

We also measure how well the copy constraint expressed in Section 5.1 is satisfied. It stipulates that every token in an MXL sentence should be either copied into one language and translated into the other or copied to

both, which mostly happens for punctuations, numbers, or proper names. No tokens should be dropped out. This is because these tokens act as strong constraints entered by the user, which therefore need to be preserved in the output.

We conduct analysis by computing the percentage of output tokens in four categories: (a) tokens that are copied exclusively, which appear only in one of the two translations; (b) tokens that are copied to both translations, which include regular tokens and (c) punctuations and digits; (d) Lost tokens that appear in neither translations. Our analysis in Table 5.3 shows that the `base` model is more likely to reproduce the patterns observed in the reference and notably is less likely to generate two copies for a token than the other systems. However, `indep` and, to a larger extent, `dual`, are able to reduce the rate of *lost tokens*, *i.e.*, of source tokens that are not found in any output for En-Fr. This shows that the interaction between the two decoders helps to increase the consistency between the two outputs. As we mainly focus on dual decoding of MXL sentences, we use the `dual` model for further analyses in this chapter.

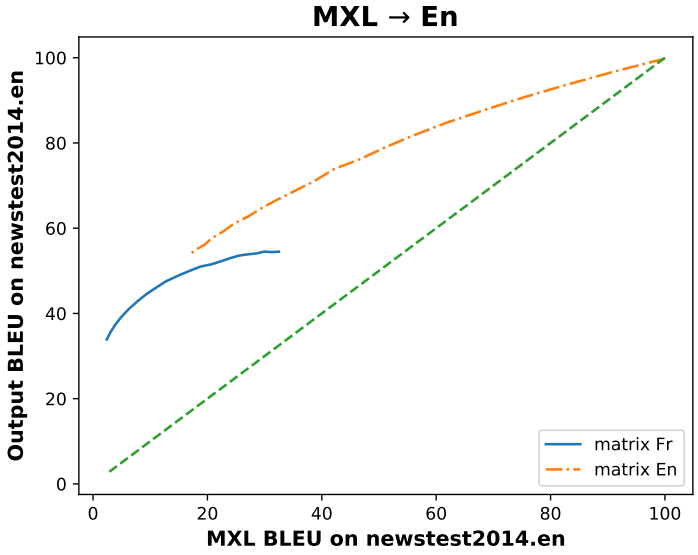
	En-Fr				En-Es			
Model	Exclusive	Both	Punc	Lost	Exclusive	Both	Punc	Lost
reference	81.56	8.10	10.34	0.00	82.67	4.96	12.37	0.00
`base`	79.14	8.85	11.29	0.72	81.16	5.72	13.02	0.10
`multi`	78.66	9.22	11.27	0.85	80.71	6.08	13.07	0.14
`indep`	78.86	9.13	11.35	0.67	80.91	5.92	13.09	0.08
`dual`	78.90	9.17	11.32	0.61	80.90	5.93	13.07	0.11

Table 5.3: Analysis of the *copy* constraint. *Exclusive* refers to the percentage of test tokens appearing in only one hypothesis. *Both* and *Punc* are for tokens and punctuations+digits appearing in both hypotheses, and *Lost* is for tokens not found in either.

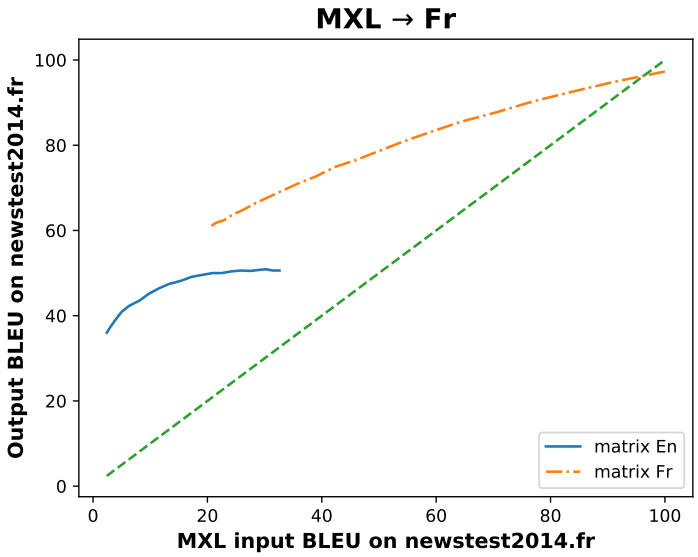
5.5.3 . Effect of Mixing Languages

In order to better study the effect of mixing languages, we modify the synthetic data generation method to keep one language as the matrix language, in which matrix segments are incrementally replaced by translations of the embedded language. We relax the constraint on the maximum number of replacements and generate new test sets with an increasing number of replacements, ranging from 1 to 20, resulting in twenty versions of the MXL test sets (in each direction).[10] In Figures 5.5 and 5.6, we plot the BLEU scores of both source MXL sentences and their translations produced

[10]We perform as many replacements for sentences that could not accommodate 20 replacements as possible.

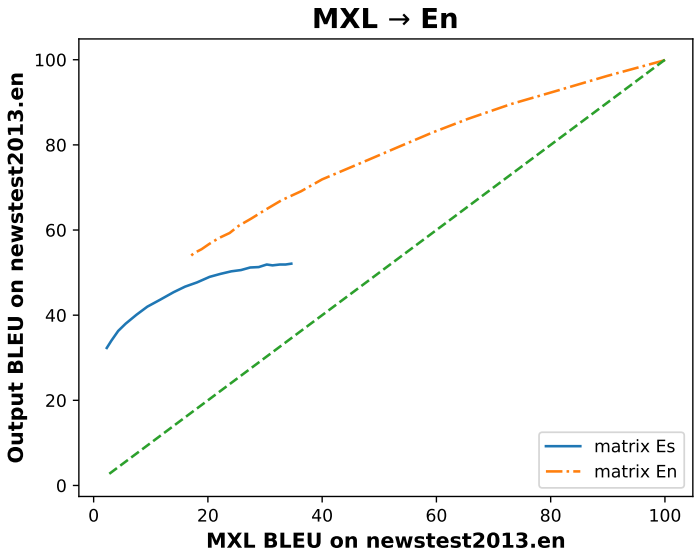


(a)

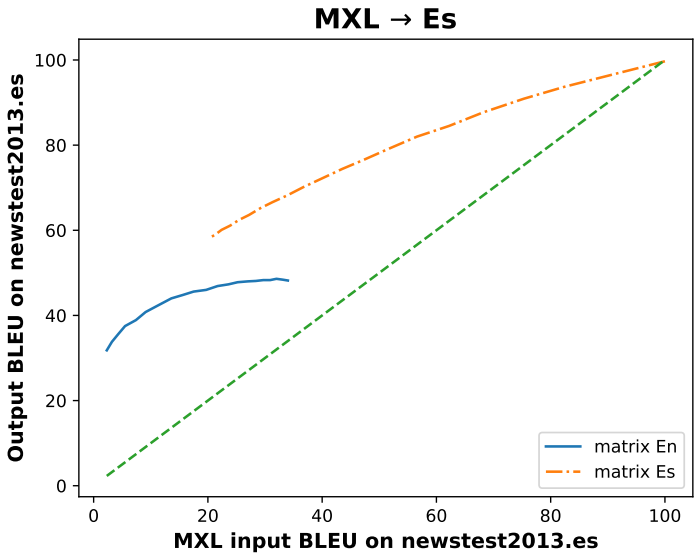


(b)

Figure 5.5: Evolution of the BLEU score of source MXL data and their target translations for En-Fr. (a) Direction MXL→En. The solid curve takes Fr as the matrix language, where we progressively inject more En segments; for the dash-dot curve, En is the matrix language, with a growing number of Fr segments. (b) Direction MXL→Fr. The target BLEU is always much higher than the source BLEU, with about a 20-point difference. The gap between the dash-dot and solid curves is due to the basic sentence structure of the matrix language (see Section 5.5.3). As dash-dot curves represent insertion in the *reference target* sentence, the corresponding BLEU score is always higher than the solid curve and actually reaches 100 (in the absence of any embedded language).



(a)



(b)

Figure 5.6: Evolution of the BLEU score of source MXL data and their target translations for En-Es. (a) Direction MXL→En. (b) Direction MXL→Es.

by the `dual` model for both language pairs by using each language as the matrix language to visualize the impact of progressively introducing more target fragments into the source.

The same behavior is observed for both language pairs and directions: on average, inserting random target fragments boosts the translation performance, with a larger payoff for the first few target segments. There exists a significant gap in the output BLEU scores when MXL source sentences with

different matrix languages reach the same (input) BLEU scores. For an MXL sentence, even though we generate a large number of replacements, the basic grammar structure of the matrix language is still maintained. Therefore, taking the target language as the matrix gives the model a pre-translated sentence structure that is much easier to reproduce.

5.5.4 . Implicit Language Identification in Translation

A second question concerns the ability of the translation system to identify target fragments in the source and to copy them to the target, even though these fragments are indistinguishable from genuine source segments. We use labels computed during the MXL data generation procedure to sort out pre-translated (target) segments from actual source segments to be translated. For instance, only tokens with a label `[eng]` denoting English are expected to be translated when translating into French. All other tokens corresponding to French words are expected to be copied. As reported in Table 5.4, our `dual` models are able to copy almost all pre-translated tokens for both language pairs and directions.

Test set	`mxl-newstest2014`		`mxl-newstest2013`	
Direction	MXL-En	MXL-Fr	MXL-En	MXL-Es
to copy	42,148	47,337	37,653	41,053
copied	41,555	46,449	37,442	40,650
copy rate (%)	98.6	98.1	99.4	99.0

Table 5.4: Analyzing the copy rate of tokens evaluated on `mxl-newstest2014` for En-Fr and `mxl-newstest2013` for En-Es. We report the number of (pre-translated) tokens that should be copied and the corresponding ratios.

Refining the analysis, we also study whether the relative order of target words changes or is preserved during the translation. Table 5.5 reports the percentage of exact copies and switched orders and the percentage of sentences where the `dual` models change the pre-translated target segments in the MXL source. We observe large differences with respect to the position of the matrix language again. When the matrix language is the target language, the model almost always preserves the observed token order since it indicates a correct sentence structure for the hypothesis. When translating into the embedded language, we observe a larger number of word order changes: in this case, inserted target segments may not appear in their correct order in the MXL sentence, an issue the model tries to fix.

An example of this order change is in Figure 5.7, where we observe a swap between the input ("différent choix") and output ("choix différent") word orders. There are also many word changes in this latter case, indicating that the model is making proper changes to make a better translation instead

Direction	En-Fr			En-Es		
	Copy	Swap	Word-change	Copy	Swap	Word-change
MXL-En	87.3	4.2	8.5	90.8	5.4	3.7
Matrix En	97.9	0.1	2.0	98.7	0.1	1.1
Matrix Fr/Es	61.1	14.3	24.6	71.8	18.2	9.9
MXL-Fr/Es	77.4	5.6	17.0	88.6	4.0	7.4
Matrix Fr/Es	84.4	0.2	15.4	97.4	0.1	2.4
Matrix En	60.5	18.5	20.9	65.1	14.2	20.7

Table 5.5: Percentage of sentences for which all target words have been precisely copied without and with order changes (*Copy* and *Swap*, respectively) and some target words in the MXL source have been changed by the model (*Word-change*), evaluated on `mxl-newstest2014` (En-Fr) and `mxl-newstest2013` (En-Es). We separately report numbers for cases where English is the matrix language (Matrix En) or embedded language (Matrix Fr/Es).

of simply reusing existing fragments. MXL translation models thus seem to perform some language identification, as they almost perfectly sort out target language tokens (which are almost always copied) from the source language tokens (which are always translated).

5.5.5 . Correcting Morphological Errors

The last issue concerns morphological errors: when inserting foreign words into a matrix source, one cannot always expect to also introduce the right inflection marks, some of which can only be determined once the target context is known. Another interesting phenomenon that we do not simulate here is when the embedded (target) lemma is adapted to bear a morphological inflection that only exists in the matrix language. For example, the English word "tokenized" should be adapted into a French sentence [*Les textes doivent être tokenisés* .] to follow the French morphological inflection. This means that two linguistic systems are mixed within the same word, thereby posing more difficulties for MT (Manandise and Gdaniec, 2011).

To illustrate the ability to correct grammar errors in input fragments, we manually add noises to an MXL sentence and display its translation in Figure 5.7. Where the input contains the lemma of the French word "tenter" (*to try*), the model inserts a modal "peuvent" to fix the context. Another illustration is the adjective "différent" which is moved into the post-nominal position, and an article ("un") is inserted. The model also corrects cross-lingual syntactic errors. The correct singular French word for "automobilist" should be "automobiliste" in the noisy input. The model corrects it by using a proper plural expression as in the reference. This indicates that the model not only copies what already exists but also tends to adjust translations whenever necessary.

En	In Oregon , planners are experimenting with giving drivers different choices .
Fr	Dans l' Orégon, les planificateurs tentent l' expérience en offrant aux automobilistes différents choix .
MXL	In l' Oregon , planners **tentent l' expérience** with giving **automobilistes différents choix** .
Hyp	*En l' Oregon , les planificateurs tentent l' expérience de donner aux automobilistes différents choix .*
Noisy MXL	In l' Oregon , planners **tenter** l' expérience with giving **automobilist différent** choix .
Hyp	*Dans l' Oregon , les planificateurs **peuvent tenter** l' expérience de donner **un choix différent aux automobilistes** .*

Figure 5.7: Translation of a noisy MXL sentence with French as both the matrix and target language.

5.6 . Computing Translations in Context

In this section, we evaluate MXL translation on a more realistic task: the SemEval 2014 Task 5: L2 Writing Assistant (van Gompel et al., 2014). This task involves translating L1 fragments within an L2 context, where the test set design is such that there is exactly one L1 fragment inserted in each utterance. This task can be handled as an MT task from MXL to monolingual, which suits our MXL translation models well. Therefore, we are able to perform direct inference on this task using our model trained in the previous sections. We evaluate two language pairs: French-English and English-Spanish, which contain 495 and 498 sentences, respectively. Below we give an example test segment provided by the organizers for each language pair (the inserted and reference segments are in boldface):

- Input (L1=English,L2=Spanish): *"Todo ello,* ***in accordance*** *con los principios que siemprehemos apoyado."*
  Reference: *"Todo ello,* ***de conformidad*** *con los principios que siempre hemos apoy-ado."*
- Input (L1=French,L2=English): *"I* ***rentre à la maison*** *because I am tired."*
  Reference: *"I* ***return home*** *because I am tired."*

The official metric for the SemEval evaluation is the word-based accuracy of the L1 fragment translation, which means that the L2 context of each sentence is not taken into account in scoring. We do not use any marks nor special tokens to locate the L1 fragment in the source text and treat it as a regular MXL sentence as in previous sections. Since our systems are full-fledged NMT systems, there is no guarantee that our systems generate translations that always contain the exact reference L2 prefix and suffix.

Therefore, we explore two options to compute these scores. The first is to post-process the output `HYP` and align it with the L2 reference context in `REF` via edit distance. This alignment allows us to retrieve and only score the relevant fragment in `HYP`. We refer to this option as `free-dec`.

The second option is to ensure that the L2 context will be present in the output translation. To this end, we use the *constrained decoding option* of `fairseq`, implementing the methods of Post and Vilar (2018); Hu et al. (2019). We explore two different ways to express the L2 context as decoding constraints. The first method treats every token in the L2 context as a separate constraint (`token-cst`). Continuing with the previous example, the L2 context "*I, because, I, am, tired.*" yields five constraints. The second way uses the prefix and suffix of the L2 context as two multi-word constraints (`presuf-cst`). In this case, "*I*" and "*because I am tired.*" yield just two constraints. In both cases, constraints are required to be present in the prescribed order in the output. Note that we only perform constrained decoding with `multi` models as they use a standard beam search during decoding.

5.6.1 . Results

Scores are computed with the SemEval evaluation tool,[11] which enables a comparison with other submissions for this task. Results are in Tables 5.6 and 5.7. For En-Es, our MXL translators (`indep` and `dual`) outperform the best system in the official evaluation (van Gompel et al., 2014) and achieve state-of-the-art performance. Note that these models are not specifically designed nor tuned in any way for the SemEval task. We only perform zero-shot inference on the SemEval test set. For Fr-En, our system achieves better performance than the fourth best participating system, with a clear gap with respect to the top results. In both cases, constraint decoding hurts performance: given that the automatic copy of target segments is already nearly perfect, introducing more constraints during the search has an apparent detrimental effect on this task.

To better study the performance gap between these language pairs, we additionally score the development and test data of this task with BLEU scores. The development sets contain 500 sentences for each language pair. We evaluate BLEU scores on full sentences generated by our systems with the complete reference sentence. Results in Table 5.8 show that we achieve performance in the same ballpark for the two language pairs, suggesting that the observed difference in the SemEval metric is likely due to a mismatch between references and system outputs. The official metric is word accuracy which may exclude acceptable translations.

[11] https://github.com/proycon/semeval2014task5

	Accuracy	Word Accuracy	Recall
UEdin-run1	0.733	0.824	1.0
UEdin-run2	0.731	0.821	1.0
UEdin-run3	0.723	0.816	1.0
CNRC-run1	0.556	0.694	1.0
`multi`			
`free-dec`	0.574	0.699	0.998
`token-cst`	0.552	0.678	0.990
`presuf-cst`	0.537	0.671	0.990
`indep`			
`free-dec`	**0.614**	**0.731**	0.998
`dual`			
`free-dec`	0.602	0.723	0.998

Table 5.6: Results of SemEval 2014 Task 5 for Fr-En.

	Accuracy	Word Accuracy	Recall
UEdin-run2	0.755	0.827	1.0
UEdin-run1	0.753	0.827	1.0
UEdin-run3	0.745	0.820	1.0
`multi`			
`free-dec`	0.733	0.807	1.0
`token-cst`	0.715	0.796	1.0
`presuf-cst`	0.715	0.796	1.0
`indep`			
`free-dec`	**0.789**	0.853	0.998
`dual`			
`free-dec`	0.787	**0.854**	1.0

Table 5.7: Results of SemEval 2014 Task 5 for En-Es.

Model	Fr-En test	En-Es test	Fr-En dev	En-Es dev
`multi`	90.3	89.2	96.9	97.4
`indep`	90.7	90.9	97.5	97.8
`dual`	90.8	90.9	97.6	97.6

Table 5.8: Results of BLEU scores on SemEval data. Performance is computed on full sentences.

5.7 . Conclusion

In this chapter, we studied the task of disentangling MXL texts into monolingual sentences of its component languages to simulate one kind of bilingual writing situation. We have conducted experiments with several NMT architectures and have proposed the dual decoder model to simultaneously generate translations in both languages within a single model. We

have also presented a data augmentation method to generate artificial MXL data. Our experiments demonstrated that artificially generated MXL data could be used to train NMT systems to translate both monolingual and MXL sentences (into one or even two languages). With dual decoding of the two languages, we were able to build systems that were as good as a baseline bilingual system on monolingual texts and much better on MXL texts. Our system does not need any explicit language identification and almost perfectly sorts out source tokens from target tokens in an MXL utterance. Another interesting feature of our system is that it is able to correct syntactic errors in the source text and make necessary modifications to produce proper translations. We have finally reported state-of-the-art results for the SemEval L2 Writing Assistant task for Es-En, while the related results for Fr-En were still somewhat lagging behind the best scores.

In the future, we would like to generate more realistic MXL texts from monolingual sentences using translation models (Gupta et al., 2020; Tarunesh et al., 2021; Song et al., 2022). Another line of research would be to continue experimenting with more realistic language data containing other linguistic phenomena. Finally, we also intend to study the somewhat more realistic condition where a mixture of languages A and B is translated into a third language C. We believe that the artificial MXL generation methods developed in our work would also be effective for this task. Note that this setting would, however, require 3-way parallel data, a situation that is considered in the next chapter, where we study other applications of dual decoding.

6 - More Applications of Dual Decoding

6.1 . Introduction

The dual decoding scheme for computing synchronized translations presented in Chapter 5 can also be used for several other purposes. In this chapter, we consider the following scenarios in which to apply dual decoding:

- Multi-target translation, where we simultaneously translate a source language into two target languages (Zhou et al., 2019);
- Bidirectional decoding, where we simultaneously *decode in two directions*, providing a new implementation of the idea of Watanabe and Sumita (2002) and Finch and Sumita (2009);
- Multi-style decoding, where we generate *coherent translation alternatives in different styles*, an idea we use to compute polite and informal variants of the same input (Sennrich et al., 2016a);
- Multilingual subtitling, where we produce subtitles in two different languages at the same time from automatic speech recognition (ASR) transcripts.

Figures 6.1 and 6.5 illustrate examples of these applications. Considering multiple applications allows us to assess the challenges and rewards of dual decoding from various angles and to better evaluate the actual agreement between the two decoders' outputs.

f	I could do that again if you want .
Zh	只要 你 愿意 我 可以 重复 一遍 。
Ja	もう 一 回 やり ましょ う か
L2R	Je peux le refaire si vous le voulez .
R2L	. voulez le vous si refaire le peux Je
polite	Ich kann das noch mal machen , wenn Sie wollen .
informal	Ich kann das noch mal machen , wenn du willst .

Figure 6.1: Instances of dual decoding: multi-target translation (§ 6.3) translating into Chinese (Zh) and Japanese (Ja); bidirectional decoding (§ 6.4) generating sequences from left to right(L2R) and right to left (R2L); multi-style decoding (§ 6.5) producing polite and informal translations.

Our main contributions are the following:

- Mitigating the data scarcity problem of multi-parallel data by fine-tuning the dual decoder model from standard translation models;
- A new parameter sharing scheme, where the two decoders share all their parameters and achieve comparable performance at a reduced

model size;
- Concrete solutions to mitigate the exposure bias problem between two decoders;
- Quantitative evaluations of the increased consistency incurred by a tight interaction between decoders.

The contributions in this chapter are published in (Xu and Yvon, 2021b; Xu et al., 2022a).

6.2 . More Techniques of Dual Decoding

6.2.1 . Fine-tuning Dual Decoder Model

In this chapter, we continue to use the dual decoder model proposed in Chapter 5. As introduced in Section 5.2.3, training the dual decoder model requires triplets of instances associating one source with two targets. However, multi-parallel corpora are relatively scarce compared to bilingual ones. Unlike Chapter 5, generating large-scale synthetic data is not trivial for applications like multi-target translation and multilingual subtitling.

We would like to take advantage of large-scale bilingual parallel data to mitigate the data scarcity issue. Therefore, we consider a two-step procedure that combines bilingual and trilingual data. In a first step, we pre-train a standard multilingual translation model (one monolingual encoder, one bilingual decoder), in which we use tags to select the target language (Johnson et al., 2017). This only requires bilingual data $\{(\mathbf{f}, \mathbf{e}^1)_i, i = 1 \dots N^1\}$ and $\{(\mathbf{f}', \mathbf{e}^2)_j, j = 1 \dots N^2\}$. We then initialize the dual decoder model with pre-trained parameters and fine-tune it with a trilingual dataset. Both decoders thus start with the same pre-trained decoder. The decoder cross-attention layers cannot benefit from pre-training and are initialized randomly. Tags are no longer necessary during the fine-tuning stage as both target translations are required.

6.2.2 . Sharing Decoders

One weakness of the dual decoder model is that it contains two separate decoders, yielding an increased number of parameters ($\times 1.6$ in our models w.r.t. standard translation models). Inspired by the idea of tying parameters in embedding matrices (Inan et al., 2017; Press and Wolf, 2017), we extend the dual decoder model by sharing all parameters in the two decoders. In this way, the two decoders become identical after sharing. The total number of parameters remains close to that of a standard translation model ($\times 1.1$) since the only increase comes from the additional decoder cross-attention layer. However, the two decoders are expected to perform decoding in different targets simultaneously. To distinguish tasks for each decoder, we prepend a

tag at the beginning of each target sentence to indicate the desired output. During inference, the tags are given as a forced prefix for each decoder to decode entire sentences. Sharing decoder is applied for multilingual subtitling in Section 6.6.

6.2.3 . Asynchronous Dual Decoding

Simultaneously generating tokens in two languages is a very strong requirement and may not bring out all the benefits of dual decoding, especially when the two target languages have different word orders. Therefore, we also propose relaxing this assumption by allowing one decoder to start generating tokens before the other. This is implemented by having the delayed decoder generate dummy tokens for a fixed number of steps before generating meaningful words, a strategy similar to the wait-k policy, which is largely used in simultaneous translation (Elbayad et al., 2020).

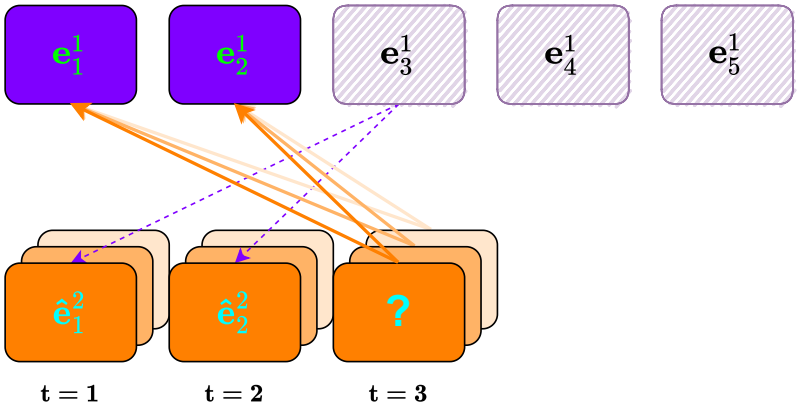


Figure 6.2: A graphical view of the asynchronous dual beam search process generating translation $\hat{\mathbf{e}}^2$ with a reference translation in $\mathbf{e}^1$ as forced prefix.

A more extreme case of delayed processing is when one decoder can access a *complete translation* in another language. In our implementation, this is simulated with partial forced decoding, where one translation is predefined while the other is being computed. We explore the asynchronous decoding in two settings: (a) within a 2-*round, sequential procedure*, where the output of the first pass for decoder 1 is fully known and fixed when computing the second-pass output for decoder 2; (b) using a *reference translation* instead of a hypothesis generated in the first pass in one of the two decoders as the forced prefix to perform controlled decoding where the output in l_{T_2} not only translates the source but is also consistent with the reference translation in l_{T_1}. The second situation is very close to multi-source translation (Och and Ney, 2001), as discussed in Section 3.5.1. Figure 6.2 illustrates the asynchronous decoding setting with a reference translation as the prefix.

Note that with the attention mask in the decoder cross-attention layer, as described in Section 5.2.1, future information from the prefixed decoder cannot be seen during the second-round decoding, even though it is available.

6.3 . Multi-target Machine Translation

6.3.1 . Data

We first evaluate our dual decoder model on the multi-target machine translation (MT) task for three directions: English to German/French (En→De/Fr), German to English/French (De→En/Fr), and English to Chinese/Japanese (En→Zh/Ja). The difference between the two settings with En, Fr, and De is that when De is in target, the word orders are very different between De and Fr. When De is the source language, the word order differences between Fr and En are not so large. The word orders are also very different between Zh and Ja, and both languages are very distinct from En. Similarly to (Wang et al., 2019; He et al., 2021a), we use the IWSLT17 dataset[1] (Cettolo et al., 2012) as our test bed. The original data is not entirely multi-parallel. We thus extract the common English sentences from En-De and En-Fr data with the corresponding translation to build a truly trilingual corpus. The En→Zh/Ja trilingual data is built similarly.

	Original De	Original Fr	Trilingual
Train	209,522	236,653	205,397
Dev	2,693	3,083	2,468
tst2014	1,305	1,306	1,168
	Original Zh	Original Ja	Trilingual
Train	235,078	226,834	213,090
Dev	3,064	3,024	2,837
tst2014	1,297	1,285	1,214

Table 6.1: The number of lines in the trilingual IWSLT data. English is used to identify trilingual sentences and is therefore not shown in this table.

The pre-training experiments, as mentioned in Section 6.2.1, additionally use WMT20[2] En-De, De-Fr, En-Zh, En-Ja, and WMT14[3] En-Fr bilingual datasets. We use the IWSLT tst2012 and tst2013 as development sets and test our model on tst2014.[4] More than 90% of the test sentences are already multi-parallel, so we follow the same procedure to build a reduced

[1] https://wit3.fbk.eu/2017-01-c

[2] https://www.statmt.org/wmt20/translation-task.html

[3] https://www.statmt.org/wmt14/translation-task.html

[4] For comparison with (He et al., 2021a), we also report the results on tst2015 in Appendix B.

trilingual test set. Table 6.1 summarizes the main statistics for trilingual training and test data used in this section.

For WMT bilingual data, we discard the ParaCrawl data and use all the rest for En-De, De-Fr, and En-Fr. We only use News Commentary, Wiki Titles, CCMT corpus, and WikiMatrix data for En-Zh. For En-Ja, we use all data except ParaCrawl and TED talks. The latter is actually the trilingual data that we do not use in the pre-training stage. We follow the preprocessing procedure as in Chapter 5 and discard sentence pairs with invalid language tags as computed by `fasttext` language identification model[5] (Bojanowski et al., 2017). For all data, we use Moses tools (Koehn et al., 2007) to normalize punctuations and remove non-printing characters. We tokenize all English, German and French data using Moses tokenizer.[6] Chinese and Japanese sentences are segmented using `jieba`[7] and `mecab`,[8] respectively. For En→De/Fr and De→En/Fr, we use a shared source-target vocabulary built with 40K Byte Pair Encoding (BPE) units (Sennrich et al., 2016c) learned on WMT data with `subword-nmt`.[9] For En→Zh/Ja, we build a 32K BPE model for English and a joint 32K BPE for Chinese and Japanese, both learned on the WMT data. Detailed statistics for the WMT data that we actually use for each language pair are in Table 6.2.

Language pair	#Sentence (M)
En-De	11.52
En-Fr	33.90
De-Fr	5.58
En-Zh	9.93
En-Ja	7.22

Table 6.2: Statistics of WMT bilingual data used in pre-training experiments for multi-target translation.

6.3.2 . Experimental Settings

We implement the dual decoder model using `fairseq`[10] (Ott et al., 2019),[11] with a hidden size of 512 and a feed-forward size of $2{,}048$. We

[5] https://dl.fbaipublicfiles.com/fasttext/supervised-models/lid.176.bin

[6] https://github.com/moses-smt/mosesdecoder

[7] https://github.com/fxsjy/jieba

[8] https://taku910.github.io/mecab/

[9] https://github.com/rsennrich/subword-nmt

[10] https://github.com/pytorch/fairseq

[11] Our implementation is open-sourced at https://github.com/jitao-xu/dual-decoding.

optimize with Adam, set up with a maximum learning rate of 0.0007 and an inverse square root decay schedule, as well as $4{,}000$ warm-up steps. For fine-tuning models, we use Adam with a fixed learning rate of $8\text{e}{-5}$. We share the decoder input and output matrices for standard Transformer models, while for dual decoder models, we share all four input and output decoder matrices. All models are trained with mixed precision and a batch size of $8{,}192$ tokens on 4 V100 GPUs. Pre-training standard translation models last for 300k iterations, while all other models are trained until no improvement is found for 4 consecutive checkpoints on the development set. Performance is computed with SacreBLEU (Post, 2018).

Similar to experiments in Chapter 5, we compare our dual decoder models `dual` to a simplified multi-task model (`indep`) without decoder cross-attention and to baseline standard Transformer models (`base`), which contain two separate models trained for each direction. All these models are trained using the IWSLT trilingual data. The pre-trained multilingual model using WMT data is denoted as `multi`.

6.3.3 . Results

We evaluate the performance of models trained only with trilingual data and the pre-trained models in a multilingual way. Table 6.3 shows that the `indep` model outperforms the `base` model in all directions, demonstrating the benefits of jointly training two independent decoders. The same gain is not observed for the `dual` model. For En→De/Fr and En→Zh/Ja, `dual` slightly outperforms the baseline models, while for De→En/Fr, the performance is even worse than the baseline. One explanation is that dual decoding suffers from a double exposure bias problem, as errors in both decoders jointly contribute to derailing the decoding process. The effect seems to be more severe for target languages with similar word orders like En and Fr. We will get back to this exposure bias issue in Section 6.4. The pre-trained `multi` models do not perform as well as `base`. This is expected as WMT data are out-of-domain for IWSLT test sets.

As discussed in Section 2.3.1, translation models are exposed to their previous predictions during inference, and errors may accumulate and derail the translation. In our dual decoder model, this exposure bias problem is amplified as the model conditions on predictions from both decoders and errors from both decoders are cascaded to future predictions. To test the possible exposure bias issue, we use the `base` model to translate source texts $\mathbf{f}$ into targets $\hat{\mathbf{e}}^1$ and $\hat{\mathbf{e}}^2$, which are then merged with the original data to build a pseudo-trilingual training set. This strategy is a form of knowledge distillation that is widely used for non-autoregressive translations, as introduced in Section 4.4. Taking En→De/Fr as an example, we use the `base` En→De model to translate half of the English source $\mathbf{f}_{1/2}$ into

Model	De-En	De-Fr	En-De	En-Fr	En-Zh	En-Ja	Avg BLEU
base	32.6	24.8	28.1	38.8	22.2	13.6	26.7
multi	31.1	24.4	25.9	37.9	25.3	10.4	25.8 (-0.9)
indep	33.8	25.4	29.1	39.8	22.6	14.8	27.6 (+0.9)
dual	31.8	22.4	28.5	38.8	22.8	15.3	26.6 (-0.1)
indep ps	33.4	26.1	28.5	39.6	22.7	14.3	27.4 (+0.7)
dual ps	33.2	25.9	28.4	39.7	22.5	14.3	27.3 (+0.6)
indep FT	37.1	28.6	30.1	42.3	26.5	17.1	30.3 (+3.6)
dual FT	37.1	28.0	30.0	42.3	26.0	17.0	30.1 (+3.4)
indep FT+ps	36.5	28.2	29.9	42.0	25.9	16.3	29.8 (+3.1)
dual FT+ps	36.5	28.4	30.1	42.1	26.1	16.5	30.0 (+3.3)

Table 6.3: BLEU scores for multi-target models. Pseudo (ps) refers to models trained from scratch with partial synthetic reference data. FT indicates models fine-tuned from the pre-trained multilingual (multi) models. FT+ps refers to models fine-tuned using partial synthetic reference data.

German $\hat{\mathbf{e}}_{1/2}$ and use the base En→Fr model to translate the other half of the English source $\mathbf{f}_{2/2}$ into French $\hat{\mathbf{e}}_{2/2}$, thus obtaining the pseudo-trilingual dataset $\{(\mathbf{f}_{1/2}, \hat{\mathbf{e}}^1_{1/2}, \mathbf{e}^2_{1/2}), (\mathbf{f}_{2/2}, \mathbf{e}^1_{2/2}, \hat{\mathbf{e}}^2_{2/2})\}$ that is as large as the original data. Pseudo datasets for De→En/Fr and En→Zh/Ja are generated similarly. For fair comparisons, we only use half of the translations as pseudo references for each target language. We see in Table 6.3 (ps) that dual models trained with these artificial references almost close the gap between the independent and dual decoder models.

Another approach we study is to fine-tune the dual decoder model on a pre-trained multilingual translation model. Initializing our trilingual models with pre-trained models, as described in Section 6.2.1, brings an additional improvement for both the indep and dual models, as shown in Table 6.3 (FT), which validates that our dual decoder model can benefit from pre-trained multilingual models even though the decoder cross attention layer is randomly initialized. This confirms that the dual decoder model can be effectively trained, even without large-scale tri-parallel data. These results also highlight the large gains of fine-tuning on a tri-parallel corpus, which improves our baseline multilingual models by nearly 5 BLEU points on average.

We additionally experiment with fine-tuning pre-trained models with the synthetic pseudo-trilingual data. However, this setting (FT+ps in Table 6.3) does not bring any gain in translation quality. For the indep models we see a small loss due to training with noisy references, while for dual, it seems that mitigating exposure bias is less impactful when starting from well-trained models.

6.3.4 . Further Analyses

The value of dual decoding is to ensure that translations $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$ are more consistent than performing independent decoding. To evaluate this, we compute the similarity scores between these two translations by computing the cosine similarity of sentence embeddings of the two translations using LASER,[12] a library to calculate and use multilingual sentence embeddings. As shown in Table 6.4, the `dual` models always generate translations that are more similar to each other on average than the `indep` models in all conditions. Note that both approaches are meant to translate the same source texts into the same languages. Therefore, the similarity scores are always quite high in all cases. Figure 6.3 shows some examples that `dual` generates more consistent translations than `indep` for the direction De→En/Fr.

Model	De→En/Fr	En→De/Fr	En→Zh/Ja	Avg SIM
`base`	89.28	91.34	81.97	87.53
`multi`	91.56	91.87	83.71	89.05 (+1.52)
`indep`	90.04	91.63	83.16	88.28 (+0.75)
`dual`	90.60	91.45	84.09	88.71 (+1.18)
`indep` ps	90.51	91.98	83.58	88.69 (+1.16)
`dual` ps	91.01	92.07	83.92	89.00 (+1.47)
`indep` FT	91.52	92.25	84.86	89.54 (+2.01)
`dual` FT	91.53	92.43	85.03	89.66 (+2.13)
`indep` FT+ps	92.17	92.68	84.76	89.87 (+2.34)
`dual` FT+ps	92.26	92.55	84.84	89.88 (+2.35)

Table 6.4: Similarity scores (SIM) for multi-target models. The similarity scores are computed as the cross-lingual similarity between sentence embeddings of the two target translations computed with LASER. Pseudo (ps) refers to models trained from scratch with partial synthetic reference data. FT indicates models fine-tuned from the pre-trained multilingual (`multi`) models. FT+ps refers to models fine-tuned using partial synthetic reference data.

As explained in Section 6.2.3, the dual decoder model is not limited to strictly synchronous generation and accommodates relaxed variants (as well as alternative dependency patterns) where one decoder can start several steps after the other. We fine-tune the "wait-k" `dual` models from the pre-trained models with $k = 3$ for the three directions and evaluate the effects. The wait-k approach requires extra training as the dual decoder model needs to be trained to consider the dummy tokens prepended in one of the two decoders. As shown in Table 6.5 (wait-3), the BLEU scores are slightly improved for both targets in some directions when either side is delayed by 3 steps. These results suggest that depending on language pairs, the

[12] https://github.com/facebookresearch/LASER

De	Ich konnte mir das niemals vorstellen.
En `indep` ps	I'**ve** never **imagined** that.
Fr `indep` ps	Je ne pouvais jamais l'imaginer.
En `dual` ps	I **could** never **imagine** that.
Fr `dual` ps	Je ne pouvais jamais l'imaginer.
De	Ich musste anders sehen, um die Vergangenheit in einer Karte nachzeichnen zu können.
En `indep` ps	I had to **look** differently to be able to **map** the past in a map.
Fr `indep` ps	J'ai dû voir différemment pour pouvoir tracer le passé dans une carte.
En `dual` ps	I had to **see** differently to be able to **track** the past in a map.
Fr `dual` ps	J'ai dû voir différemment pour pouvoir tracer le passé dans une carte.

Figure 6.3: Examples of the `dual` model generating more consistent translations than the `indep` model for De→En/Fr. Both models are trained from scratch using partial synthetic reference data (ps). Differences between `indep` and `dual` are in bold. Consistent translations in `dual` between En and Fr are marked with the same color.

Model	De-En	De-Fr	En-De	En-Fr	En-Zh	En-Ja
`dual`	31.8	22.4	28.5	38.8	22.8	15.3
`dual` 2-round	31.7	24.4	28.7	39.3	22.8	15.2
`dual` ref	33.3	30.8	28.7	39.6	23.1	15.3
`dual` FT	37.1	28.0	30.0	42.3	26.0	17.0
`dual` FT l_{T_1} wait-3	36.2	28.3	30.2	42.4	26.2	17.0
`dual` FT l_{T_2} wait-3	36.7	25.8	30.4	42.6	26.6	17.0
`dual` FT 2-round	37.1	28.1	30.0	42.5	26.2	16.8
`dual` FT ref	37.1	31.1	30.0	42.4	26.4	17.0

Table 6.5: BLEU scores for asynchronous decoding: sequential decoding on the `dual` models trained from scratch (top), wait-3 models fine-tuned on the pre-trained models, and sequential decoding on the `dual` FT models (bottom). Results using 2-round sequential decoding for one decoder are obtained in a second decoding pass using either automatic (2-round) or reference (ref) translations.

information flow between decoders can be beneficial from a small amount of asynchronicity. It can also be harmful to language pairs like De→En/Fr, as word orders in En and Fr are better synchronized than other language pairs.

Our implementation also enables us to have one decoder completing its translations before the other one begins. We thus experiment with a *2-round sequential decoding strategy* (see Section 6.2.3), in which we first compute the complete translation $\hat{\mathbf{e}}^1$ in one target language with the `dual` model,

then decode the other one $\hat{\mathbf{e}}^2$ with $\hat{\mathbf{e}}^1$ as a forced prefix in the other decoder. One can also use the reference $\mathbf{e}^1$ as the prefix when available. In this case, the second decoding step has access to both the source and the other target sequence. Compared to the wait-k approach, the 2-round decoding strategy does not require any additional training and can be applied directly during inference.

We decode both `dual` and `dual` FT models with this strategy. Results in Table 6.5, obtained with both automatic and reference translations in one language, show that this technique is able to improve the `dual` models in almost all directions trained from scratch. 2-round sequential decoding with reference in one language provides the other decoder with the ground truth, therefore alleviating the exposure bias problem that hurts the `dual` models. However, combining the results of FT models in Tables 6.3 and 6.5, we see that fine-tuned models are less sensitive to errors made during decoding. This again shows the benefits that `dual` models actually get from using pre-trained models.

6.4 . Bidirectional Decoding

Bidirectional decoding (Finch and Sumita, 2009; Zhou et al., 2019; Liu et al., 2020b) aims to integrate future information in the decoder by jointly translating in the forward (L2R) and the backward (R2L) directions. Another expectation is that the two decoders, having different views of the source, will deliver complementary translations. Dual decoding readily applies in this setting, with one decoder for each direction, with the added benefit of generating more coherent outputs than independent decoders. We evaluate this added consistency by reusing the experimental setting (data, implementation, and hyperparameters) of Section 6.3 and by training 4 bidirectional systems from En into De, Fr, Zh, and Ja. Similar to Zhou et al. (2019), the selected output translation (forward or backward) is the one that has a higher probability. We invert the translation if the R2L output is picked. We measure the consistency between translations in the two directions by computing the averaged BLEU score between the two translations.

We first train models using the same IWSLT corpora as in Section 6.3 for each language pair. To build the tri-parallel training data for this task, we add an inverted version of the target sentence to each training sample to build the R2L reference. We reuse the `base` models from Section 6.3 for the corresponding language pairs. Results are reported in Table 6.6. In this setting, the `dual` models again suffer a clear drop in BLEU scores as compared to the `indep` models. We again attribute this gap to the impact of the exposure bias. As seen in Table 6.7, the loss in BLEU scores of the `dual` systems is accompanied by a very large increase in consistency of the

Model	En-De	En-Fr	En-Zh	En-Ja	Avg BLEU
base	28.1	38.8	22.2	13.6	25.7
indep	29.1	39.4	22.5	14.8	26.5 (+0.8)
dual	25.9	36.6	20.6	4.2	21.8 (-3.9)
indep ps	29.0	39.9	22.9	15.6	26.9 (+1.2)
dual ps	28.7	38.9	23.1	15.1	26.5 (+0.8)
indep ps-dup	29.3	40.5	23.5	15.8	27.3 (+1.6)
dual ps-dup	29.6	40.1	23.4	15.3	27.1 (+1.4)

Table 6.6: BLEU scores for bidirectional decoding models trained with actual data (top) and synthetic data (bottom). Pseudo (ps) refers to models trained with partial synthetic reference data. *ps-dup* systems are trained with twice as much data as *ps* settings.

outputs ($+31.1$). This indicates that the dual models learn to directly use tokens from the other direction as if they were always correct during training. However, during inference, once the prediction made by one direction is incorrect, the other direction simply copies the wrong token, which yields severe cascading errors.

To mitigate this problem, we again introduce pseudo-parallel targets, as in Section 6.3.3, where one of the two targets is automatically generated with the base model, as also performed by Zhou et al. (2019); Wang et al. (2019); Zhang et al. (2020a); He et al. (2021a). We generate the synthetic references using individual En→De$_{L2R}$ and En→De$_{R2L}$ Transformer models, for instance. The En→De$_{R2L}$ system is trained on En-De trilingual data with the German reference simply inverted. Similar to the pseudo-data described in Section 6.3.3, we generate a pseudo dataset $\{(\mathbf{f}_{1/2}, \mathbf{e}^{fw}_{1/2}, \hat{\mathbf{e}}^{bw}_{1/2}), (\mathbf{f}_{2/2}, \hat{\mathbf{e}}^{fw}_{2/2}, \mathbf{e}^{bw}_{2/2})\}$, in which each original source sentence occurs just once. This pseudo dataset assures that the forward and backward training target sentences are not always deterministically related, which forces each decoder to put less trust in tokens from the other direction. We also consider a duplicated pseudo dataset $\{(\mathbf{f}, \mathbf{e}^{fw}, \hat{\mathbf{e}}^{bw}), (\mathbf{f}, \hat{\mathbf{e}}^{fw}, \mathbf{e}^{bw})\}$, in which each source sentence is duplicated, occurring once with the reference data in each direction. Results in Table 6.6 show that both indep and dual models can benefit from the pseudo data (ps). The dual models again close the gap with indep and yields systems that surpass the baseline by about 1 BLEU point. Systems trained with duplicated pseudo data (ps-dup) can further improve the performance by 0.6 BLEU points on average.

By computing the averaged BLEU score between the two output translations $\mathrm{BLEU}(\hat{\mathbf{e}}^{fw}, \hat{\mathbf{e}}^{bw})$ and $\mathrm{BLEU}(\hat{\mathbf{e}}^{bw}, \hat{\mathbf{e}}^{fw})$, we can also evaluate the increment of consistency incurred in dual decoding. These scores are reported in Table 6.7. Our dual models always produce more consistent translations than the indep models, with a $+13.4$ to $+17.9$ increment in consistency

Model	En-De	En-Fr	En-Zh	En-Ja	Avg Cons
`indep`	54.7	65.8	51.3	37.6	52.4
`dual`	88.9	90.5	86.0	68.4	83.5 (+31.1)
`indep` ps	65.7	73.3	62.0	48.6	62.4
`dual` ps	83.7	89.6	80.2	67.5	80.3 (+17.9)
`indep` ps-dup	70.9	76.3	67.6	53.7	67.1
`dual` ps-dup	83.5	89.6	78.2	70.7	80.5 (+13.4)

Table 6.7: Consistency scores (Cons) for bidirectional decoding models trained with actual data (top) and synthetic data (bottom). The consistency score is an averaged BLEU score between the forward and backward translations $\mathrm{BLEU}(\hat{\mathbf{e}}^{fw}, \hat{\mathbf{e}}^{bw})$ and $\mathrm{BLEU}(\hat{\mathbf{e}}^{bw}, \hat{\mathbf{e}}^{fw})$. Pseudo (ps) refers to models trained with partial synthetic reference data. *ps-dup* systems are trained with twice as much data as *ps* settings.

when averaged over language pairs, thereby demonstrating the positive impact of dual decoding.

6.5 . Multi-style Decoding

As another application of dual decoding, we study the generation of pairs of consistent translation alternatives, using variation in "politeness" as our test bed. Some languages like German, French, Chinese, and Japanese contain several levels of formalities expressed by using honorifics that English does not possess. When translating from English to these languages, it is difficult to predict the appropriate honorific, while users may want to control the politeness level of the translation (Sennrich et al., 2016a). The principal purpose of this study is to provide deterministic control over the generation to produce consistent variants with the possibility to choose which version is produced.

6.5.1 . Datasets and Experimental Settings

We borrow the experimental setting and data of Sennrich et al. (2016a).[13] The training data consists of OpenSubtitles2012 En-De data with 5.58M sentence pairs, out of which 0.48M of German references are annotated as polite and 1.06M as informal. The rest is deemed neutral. The annotation tool is based on the ParZu dependency parser[14] and an annotation script released with the data. Polite/Informal tags are based on an automatic analysis of the German side according to rules described in (Sennrich et al., 2016a). The `test` set, which we use as the development set, is a random

[13] `http://data.statmt.org/rsennrich/politeness/`

[14] `https://github.com/rsennrich/ParZu`

sample of $2{,}000$ sentences from OpenSubtitles2013. We use the `testyou` set as our main test set, which consists of $2{,}000$ random sentences also extracted from OpenSubtitles2013 where the English source contains a second-person pronoun *you(r(s(elf)))* in the English source. We built a shared vocabulary with a joint BPE of 32K merge operations.

The German reference sentences are only annotated with politeness labels, but alternative translations do not exist. Therefore, we manage to generate tri-parallel data to fit our dual decoder model. We first pre-train a tag-based MT system with politeness control as in (Sennrich et al., 2016a) and use it to predict the polite counterpart of each informal sentence and vice-versa. We label the source polite sentences with an informal tag and use the pre-trained model to generate corresponding synthetic informal translations. The polite translations are generated similarly. We also randomly extract an equivalent number of neutral sentences as the polite and informal ones, *i.e.*, 1.54M. References of neutral sentences are identical for both polite and informal targets. The resulting tri-parallel corpus thus contains 3.07M sentences. Similar to the multi-target task (Section 6.3), we fine-tune the pre-trained model with this data until convergence. During fine-tuning, we omit the control tags used in pre-training as both polite and informal translations are required. The annotation tool distributed with the data is used to assess the politeness of the output translations.

6.5.2 . Results

Table 6.8 (top) reports the performance of the pre-trained model. *Reference* refers to the annotation result of the German reference sentences. *None* is translated without adding any tags to the source text, while *polite* and *informal* are translated with all sentences tagged respectively as *polite* and *informal*. *Oracle* is obtained by prefixing each source input with the reference tag. These results show the effectiveness of side constraints for the generation of variants. For both polite and informal categories, the pre-trained model generates translations that mostly satisfy the desired requirement.

Results of the fine-tuned dual decoder models are in Table 6.8 (bottom): we see that `indep` and `dual` models are very close and tend to generate more neutral translations. Compared to the pre-trained model, they also slightly improve the BLEU scores.

As discussed in Section 6.2.3, our dual decoder model can delay one decoder until the other is finished. We redo the same 2-round decoding procedure as in Section 6.3.4. Results in Table 6.8 (bottom) indicate that given the full translation of informal variations, the `dual` model tends to generate less neutral sentences but more polite ones. The same phenomenon is also observed in the other direction. This implies that the output variations can be better controlled with 2-round sequential decoding.

Model	Tag	neutral	polite	informal	BLEU
	reference	438	525	1037	-
`pre-train`	none	1914	16	70	17.7
	polite	479	1518	3	20.9
	informal	22	0	1978	24.1
	oracle	551	406	1043	30.2
	Decoders	neutral	polite	informal	BLEU
`indep`	polite	528	1470	2	21.0
	informal	82	0	1918	24.4
`dual`	polite	541	1457	2	21.3
	informal	97	0	1903	24.3
2-round informal	polite	531	1467	2	21.2
2-round polite	informal	84	0	1916	24.4

Table 6.8: Results of politeness MT models. Tags are used for the `pre-train` model to generate the desired styles. Decoders of `indep` and `dual` compute two translations in one decoding step, while the results using sequential asynchronous decoding for one decoder are obtained with the 2-round procedure of Section 6.2.3.

6.6 . Multilingual Subtitling

As the amount of online audio-visual content continues to grow, the need for captions and subtitles in multiple languages also steadily increases, as it widens the potential audience of these contents. In this section, we use "caption" to refer to a text written in the same language as the audio and "subtitle" when translated into another language. Captions, which are often meant for viewers with hearing difficulties, and subtitles, which are produced for viewers with an imperfect command of the source language, may have slightly different traits that we ignore here.

Both activities are closely related: human subtitle translators often generate subtitles directly based on the original captions without viewing or listening to the original video/audio files. This strategy, however, runs the risk of amplifying simplifications or errors present in the captioning in the subtitle approximations. It may even happen that both texts need to be simultaneously displayed on the screen, for instance, in countries with several official languages or to help foreign language learners. This means that captions and subtitles need to be consistent not only with the video content but also with each other. It also implies that they should be synchronized (Karakanta et al., 2021). Finally, even in scenarios where only subtitles would be needed, simultaneously generating captions may still help better check the correctness of subtitles.

Early approaches to automatic subtitling (*e.g.*, Piperidis et al., 2004) also assumed a pipeline architecture (Figure 6.4 (b)), where subtitles are

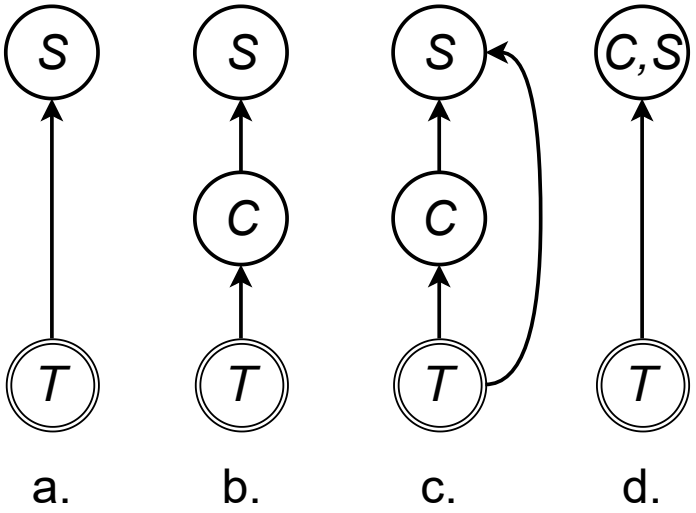


Figure 6.4: A graphical view of various captioning and subtitling strategies. T refers to transcripts. C and S denote captions and subtitles, respectively.

translated from captions derived from ASR transcripts. A recent alternative (Figure 6.4 (a)), which mitigates cascading errors, is to independently perform captioning and subtitling in an end-to-end manner (Liu et al., 2020a; Karakanta et al., 2020a). The risk, however, is to generate inconsistencies (both in alignment and content) between the two textual streams. This approach might also be limited by the lack of appropriate training resources (Sperber and Paulik, 2020). Various ways to further strengthen the interactions between these tasks by sharing parameters or loss terms are evaluated by Sperber et al. (2020). Figure 6.4 (c) illustrates these approaches.

In this section, we explore simultaneously generating both captions and subtitles from ASR transcripts using dual decoding,[15] as illustrated in Figure 6.4 (d). Note that our work is not completely end-to-end from audio signals to subtitles. We focus on text-to-text experiments. We also explore sharing the decoder parameters as described in Section 6.2.2. Generally speaking, automatically turning ASR transcripts into full-fledged captions involves multiple changes, depending on the specification of the captioning task. In our case, this transformation comprises four main aspects: the segmentation for display (via tag insertion), the removal of certain features from spoken language (*e.g.*, fillers, repetitions, or hesitations), ASR error correction, and punctuation prediction. The transcript-to-subtitle task involves the same transformations, with an additional translation step to produce text in another language. Figure 6.5 illustrates the various transformations between input transcripts and the corresponding output segments.

[15]This is a joint work with François Buet.

Transcript	**i 'm** combining specific types of signals **the** mimic how our body **response** to **in an injury** to help us regenerate
Caption	**I'm** combining specific types of signals **[eob] that** mimic how our body **responds** to **injury [eol]** to help us regenerate**. [eob]**
Subtitle	Je combine différents types de signaux [eob] qui imitent la réponse du corps [eol] aux blessures pour nous aider à guérir. [eob]

Figure 6.5: Example of a triplet (transcript, caption, subtitle) from our tri-parallel data. Differences between transcript and caption are in bold. The segmentation tags `[eob]` and `[eol]` refer to *change-of-screen* and *end-of-line*, respectively.

As our experiments suggest, a tighter integration not only improves the quality and consistency of captions and subtitles but also enables better use of all available data, with hardly any impact on model size.

6.6.1 . Datasets and Resources

For our experiments, we use MuST-Cinema[16] (Karakanta et al., 2020b), a multilingual Speech-to-Subtitles corpus compiled from TED talks, in which subtitles contain additional segmentation tags indicating change-of-screen (`[eob]`) or line (`[eol]`). Our experiments consider the translation from English into French. Our tri-parallel data also includes a pre-existing unpunctuated ASR output generated by Karakanta et al. (2020a), which achieves a word error rate (WER) score of 39.2% on the MuST-Cinema test set speech transcripts. In order to emulate a real production scenario, we segment these transcripts as if they were from an ASR system performing segmentation based on prosody. As this kind of ASR system tends to produce longer sequences compared to typical written text (Cho et al., 2012), we randomly concatenate the English captions into longer sequences, to which we align the ASR transcripts using the conventional edit distance, thus adding a sub-segmentation aspect to the translation task. This concatenation method was proposed and implemented by François Buet. More details about the data processing procedure are in Appendix C.1. The resulting training, development, and test data actually used for this experiment contains 133k, 499, and 255 lines, respectively.

Similar to Section 6.3, we continue to apply the pre-training fine-tuning scheme to take advantage of large-scale bilingual data. As we work only on the En-Fr direction, the pre-trained model is not a multilingual model and thus does not use any tags in pre-training. For pre-training, we use the

[16] https://ict.fbk.eu/must-cinema/

WMT14 En-Fr data as in Section 6.3.1. For fine-tuning, we use synthetic tri-parallel data similar to Sections 6.3 and 6.4, in which we alternatively replace one of the two target side references with hypotheses generated from a baseline system for the corresponding direction. We tokenize all data with Moses scripts and use a shared source-target vocabulary of 32K BPE (Sennrich et al., 2016c).

6.6.2 . Experimental Settings

We pre-train the En→Fr translation model for 300k iterations. All other models are trained until no improvement is found for 4 consecutive checkpoints on the development set. We mainly measure performance with BLEU. TER (Snover et al., 2006) and BERTScore (Zhang et al., 2020b) are also reported in Appendix C.3. Segmentation tags in subtitle texts are also taken into account, and BLEU scores are computed over full sentences. In addition, measuring the consistency between captions and subtitles is also important. We reuse the structural and lexical consistency score proposed by Karakanta et al. (2021).[17] *Structural consistency* measures the percentage of utterances having the same number of blocks in both languages, while *lexical scores* count the proportion of words in the two languages that are aligned in the same block (refer to Appendix C.2 for additional details).

We call the dual decoder model `dual`. Models sharing the parameters of the two decoders are denoted as `share`. Baseline translation models separately trained for each direction (T_{en}→C_{en}, T_{en}→S_{fr}) are denoted by `base`. To study the effectiveness of dual decoding, we mainly compare `dual` with a `pipeline` system. The latter uses the `base` model to produce captions which are then translated into subtitles using an independent system trained to translate from caption to subtitle (T_{en}→C_{en}→S_{fr}).

Like the `dual` model, `base` and `pipeline` systems can also benefit from pre-training. For the former, we pre-train the direct transcript-to-subtitle translation model (T_{en}→S_{fr}); for `pipeline`, the caption-to-subtitle model (C_{en}→S_{fr}) is pre-trained, while the first step (T_{en}→C_{en}) remains as in the `base` system since both source and target sides are English. Note that all fine-tuned systems start with the same model pre-trained using WMT En-Fr data, even though fine-tuned systems need to predict segmentation tags that are not seen during pre-training.

6.6.3 . Main Results

We only report in Table 6.9 the performance of the two baselines and fine-tuned (+FT) models, as our preliminary experiments showed that train-

[17] We would like to thank Alina Karakanta for sharing her code to compute the consistency scores.

Model	BLEU			Consistency ↑	
	En	Fr	Avg	Structural	Lexical
base	55.7	23.9	39.8	55.3	70.7
base + FT	55.7	24.9	40.3	54.5	71.4
pipeline	55.7	23.6	39.7	95.7	96.0
pipeline + FT	55.7	24.2	40.0	98.4	98.3
dual + FT	**56.9**	25.6	**41.3**	65.1	79.1
share + FT	56.5	**25.8**	41.2	**66.7**	**80.0**

Table 6.9: BLEU scores for captions (En) and subtitles (Fr), with structural and lexical consistency scores between the two hypotheses. The consistency scores are in percentage. The base and pipeline settings are trained from scratch with original data. +FT indicates models fine-tuned from the pre-trained translation model. share refers to tying all decoder parameters.

ing the dual decoder model with only tri-parallel data was not optimal. The BLEU score of the *do-nothing* baseline, which copies the source ASR transcripts to the output, is 28.0, suggesting that the captioning task actually involves much more transformations than simply inserting segmentation tags. We observe that fine-tuning improves subtitles (Fr) generated by base and pipeline systems by ~1 BLEU. Once fine-tuned with synthetic tri-parallel data, our dual decoder model outperforms base+FT by 0.7 BLEU and pipeline+FT by 1.4 BLEU in the subtitling direction. Fine-tuning also delivers a positive effect on the captioning direction, as dual is able to improve 1.2 BLEU over base. Sharing all parameters of both decoders yields a further increase of 0.2 BLEU for subtitling, with about one-third fewer parameters. This indicates that sharing parameters between decoders is effective.

We also measure the structural and lexical consistency between captions and subtitles generated by our systems in Table 6.9. As expected, pipeline settings always generate very consistent pairs of captions and subtitles, as subtitles are direct translations of the captions. All other methods generate both outputs directly from the ASR transcripts. dual models do not perform as well but are still able to generate captions and subtitles with a much higher structural and lexical consistency between the two outputs than the base systems. Experiments in previous sections show that dual decoder models generate translations that are more consistent in content. We further show here that our dual models generate hypotheses that are also more consistent in structure. Some examples of output captions and subtitles produced by the share model are shown in Figure 6.6.

6.6.4 . The Effect of Fine-tuning

As the pre-trained En→Fr model has never seen sentences in the source language (En) on the target side, we first only use it to initialize the subti-

Transcript	take time to write down your values your objectives and your key results do it today
En `pipeline` +FT	Take time to write down [eol] your values, your objectives, [eob] and your key results do it today. [eob]
En `share` +FT	Take time to write down your values, **[eol]** your objectives, [eob] and your key results do it today. [eob]
En ref	Take time to write down your values, [eob] your objectives and your key results. [eob] Do it today. [eob]
Fr `pipeline` +FT	Prenez le temps d'écrire vos valeurs, [eol] vos objectifs, [eob] et vos principaux résultats [eol] le font aujourd'hui. [eob]
Fr `share` +FT	Prenez le temps d'écrire vos valeurs, **[eob]** vos objectifs et **vos résultats clés. [eob] Faites-le** aujourd'hui. [eob]
Fr ref	Prenez le temps d'écrire vos valeurs, [eob] vos objectifs et vos résultats clés. [eob] Faites-le aujourd'hui. [eob]
Transcript	and as it turns out what are you willing to give up is exactly the right question to ask
En `pipeline` +FT	And as it turns out, what are you willing [eol] to give up is exactly [eob] the right question to ask? [eob]
En `share` +FT	And as it turns out, what are you willing [eol] to give up **[eob]** is exactly the right question to ask? [eob]
En ref	And as it turns out, [eob] "What are you willing to give up?" [eob] is exactly the right question to ask. [eob]
Fr `pipeline` +FT	Et il s'avère que ce que vous voulez abandonner [eol] est exactement [eob] la bonne question à poser ? [eob]
Fr `share` +FT	Et il s'avère que ce que vous voulez abandonner **[eob]** est exactement la bonne question à poser**.** [eob]
Fr ref	Et il s'avère que [eob] « Qu'êtes-vous prêts à abandonner ? » [eob] est exactement la question à poser. [eob]

Figure 6.6: Examples of dual decoding improving both captioning and subtitling. Major improvements are marked in bold.

Model	En	Fr	Avg
`dual` 1-decoder + FT	55.3	25.3	40.3
`dual` + FT	56.9	25.6	41.3
`share` + FT	56.5	25.8	41.2

Table 6.10: BLEU scores for multiple initialization settings when fine-tuning dual decoder models. 1-decoder refers to only initializing the subtitling decoder with pre-trained parameters.

tling decoder and use a random initialization for the captioning decoder. To study the effect of initialization, we conducted an ablation study by comparing three settings: initializing only the subtitling decoder (1-decoder), both

decoders, and the shared decoder. As illustrated in Table 6.10, initializing both decoders brings improvements in both directions, with a gain of 1.6 BLEU for captioning and 0.3 BLEU for subtitling. Moreover, sharing parameters between decoders further boosts the subtitling performance by 0.2 BLEU. As it seems, the captioning decoder also benefits from a decoder pre-trained in another language.

6.6.5 . Mitigating Exposure Bias

We also analyze the influence of the exposure bias issue in this application scenario. To this end, we compare fine-tuning the `dual` model with original vs. artificial tri-parallel data. For simplicity, we only report the averaged captioning and subtitling BLEU scores in Table 6.11. Results show that fine-tuning with the original data (w/real) strongly degrades the BLEU scores for the generated text, resulting in worse performance than the baseline.

Model	Decoding Setting		
	Baseline	2-round	Ref
`base`	39.8	-	-
`dual` + FT w/real	39.2	40.9	45.0
`share` + FT w/real	38.6	40.1	43.9
`dual` + FT	41.3	41.2	41.0
`share` + FT	41.2	40.9	40.5

Table 6.11: Performance of various decoding methods. Models fine-tuned with the original data are denoted with *w/real*. All BLEU scores are averaged over the two outputs. 2-*round* and *Ref* refer to 2-round decoding with respectively model predictions and references as forced prefixes in one direction.

In another set of experiments, we perform asynchronous 2-round sequential decoding, as in Section 6.3 and Section 6.5. We first decode the `dual` models to obtain hypotheses in both languages $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$. During the second decoding round, we use the output English caption $\hat{\mathbf{e}}_1$ as a forced prefix when generating the French subtitles $\hat{\hat{\mathbf{e}}}_2$. The final English caption $\hat{\hat{\mathbf{e}}}_1$ is obtained similarly. The 2-round decoding scores are shown in Table 6.11 and compared with the optimal situation where we use references instead of model predictions as forced prefixes in the second round (in column Ref).

Results in Table 6.11 suggest that dual decoder models fine-tuned with original data (w/real) are pretty sensitive to exposure bias, which can be mitigated with artificial tri-parallel data. Their performance can be improved by ~ 1.5 BLEU when using 2-round decoding, thereby almost closing the initial gap with models using synthetic data. Fine-tuning with synthetic tri-parallel data is slightly better overall and more stable across various decoding configurations, which is also found in Section 6.3.4.

6.7 . Conclusion

In this chapter, we have explored dual decoding in several application scenarios as a way to generate pairs of consistent translations. The dual decoder model can be effectively trained using actual or partly artificial data. It can also directly benefit from pre-trained models, therefore making the best use of large-scale bilingual data. Considering the four applications, we have observed that dual decoding is prone to exposure bias in the two decoders, and we have proposed practical remedies. Using these, we have achieved BLEU scores that match those of simple multi-task learners and display increased consistency between the two outputs, further confirming the findings in Chapter 5. Additionally, we showed that parameter sharing on the decoder side is viable and effective, at least for related languages. A limitation of the dual decoder model is that the decoding time is twice as slow as regular translation models, as mentioned in Section 5.2.2. However, with shared decoders, our model yields almost no additional cost in space.

In our future work, we plan to consider other strategies, such as scheduled sampling (Bengio et al., 2015; Mihaylova and Martins, 2019), to mitigate the exposure bias problem. Another area we seek to improve is the relaxation of strict synchronicity in decoding. We finally wish to study more applications of this technique, notably to generate controlled variation: controlling gender variation (Zmigrod et al., 2019) or more complex forms of formality levels, as in (Niu and Carpuat, 2020), are obvious candidates.

7 - Bilingual Synchronization

7.1 . Introduction

In this chapter, we study another scenario of bilingual writing. As introduced in Chapter 1, this scenario extends online commercial translation systems. Unlike the proposition in Chapter 5, we assume that texts in two languages are separated into two boxes, while users can freely choose to compose in either language at their will, as illustrated in Figure 7.1. Once texts in one language are edited, the other language is then automatically synchronized so that the two sentences are always mutual translations. To this end, the technical challenge is to keep the two input texts synchronized while taking the users' input into account. Our scenario is a joint bilingual edition and translation process, requiring multiple decodings with small changes. However, neural machine translation (NMT) is generally viewed as a one-shot activity process that generates the target sentence based on the sole source language input.

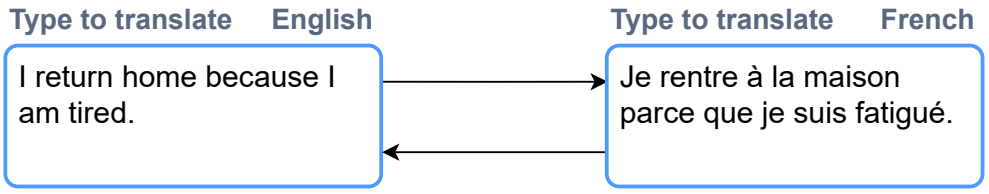


Figure 7.1: A design of an online bilingual writing system. Users can freely choose the language to write, and the system automatically updates the texts in the other box.

Here We focus on the revision part of the translation process and consider *bilingual synchronization* (Bi-sync), which we define as follows. Given an existing pair of a source ($\mathbf{f}$) and a target ($\tilde{\mathbf{e}}$) sentences, *which may or may not be mutual translations*, the task is to compute a revised version $\mathbf{e}$ of the initial translation $\tilde{\mathbf{e}}$, such that $\mathbf{e}$ is an actual translation of the source sentence $\mathbf{f}$. Our definition also implies the detection of parallel sentences, where no edition is needed. Bi-sync is necessary for the bilingual writing scenario when the source side of an existing sentence pair is edited, requiring updating the target accordingly so that both sides remain synchronized.

In principle, Bi-sync is more general than standard machine translation (MT). The latter corresponds to the synchronization with an initially empty target ($\tilde{\mathbf{e}} = [\ ]$). More interesting situations in Bi-sync occur when parts of the initial target text can be reused, meaning that the synchronization

model only needs to make a few changes to synchronize the translation. In this situation, Bi-sync mainly acts to edit an initial translation $\tilde{\mathbf{e}}$ rather than to generate a translation from scratch.

Bi-sync also encompasses several existing tasks incorporating auxiliary information in the translation process, as we introduced some of them in Chapter 3: synchronization is needed in interactive MT (Knowles and Koehn, 2016) and bilingual editing (Bronner et al., 2012), where $\tilde{\mathbf{e}}$ is the translation of a previous version of $\mathbf{f}$; in MT with lexical or terminological constraints (Hokamp and Liu, 2017; Post and Vilar, 2018), where $\tilde{\mathbf{e}}$ initially contains target-side constraints (Susanto et al., 2020; Xu and Carpuat, 2021a); in translation memory (TM) based approaches (Bulte and Tezcan, 2019), where $\tilde{\mathbf{e}}$ is a retrieved similar translation for a source sentence; in automatic post-editing (APE, do Carmo et al., 2021), where $\tilde{\mathbf{e}}$ is the output of an MT system; in a newly proposed translation error correction task (Lin et al., 2022), where $\tilde{\mathbf{e}}$ is a human translation that contains errors made by human translators when producing the translation; as well as in parallel corpus cleaning and fixing (Negri et al., 2013; Carpuat et al., 2017; Pham et al., 2018), where $\tilde{\mathbf{e}}$ is a possibly noisy translation that should be filtered out or fixed.

We develop and evaluate methods to implement sequence-to-sequence models that perform Bi-sync. In this chapter, we focus on autoregressive approaches that are similar to Bulte and Tezcan (2019), where the source sentence and the initial translation are concatenated into one input sequence, as discussed in Section 3.3.2. We also study various ways to artificially generate the triplets of examples ($\mathbf{f}$, $\tilde{\mathbf{e}}$, $\mathbf{e}$) that are needed in training. We will study non-autoregressive approaches to performing Bi-sync in Chapter 8.

We also evaluate the Bi-sync model on two specific downstream tasks: parallel corpus cleaning and fixing and NMT with TMs. The former attempts to synchronize noisy segments in a parallel corpus to improve the corpus quality. This is arguably more difficult than the Bi-sync task, as many initial translations are already correct and need to be kept unchanged. As introduced in Section 3.3, NMT with TMs tends to make use of similar translations extracted from TM to help improve the translation quality. We study how to apply fine-tuning of general Bi-sync models to both tasks to evaluate the adaptation capability of Bi-sync models to more specific tasks.

Our main contributions are the following:

- We propose a new Bi-sync task and several methods to generate synthetic editing data for different types of editing operations.
- We design a specific training procedure for autoregressive edit-based model and show that our model can be trained with artificial data.
- We conduct various analyses to better understand our model under the new Bi-sync task.
- We perform empirical validations of the Bi-sync model on several spe-

cific tasks.

Some of the contributions in this chapter are published in (Xu et al., 2020, 2022b).

7.2 . Generating Editing Data

We consider a general scenario where, given a pair of sentences $\mathbf{f}$ and $\tilde{\mathbf{e}}$, assumed to be related but not necessarily parallel, we aim to generate a target sentence $\mathbf{e}$ that is parallel to $\mathbf{f}$. We would also like $\tilde{\mathbf{e}}$ and $\mathbf{e}$ to be close, as, in our settings, $\tilde{\mathbf{e}}$ is often a valid translation of a sentence that is close to $\mathbf{f}$. Training such models requires triplets of examples $(\mathbf{f}, \tilde{\mathbf{e}}, \mathbf{e})$. While large amounts of parallel bilingual data are available for many language pairs, they are however rarely associated with related translations $\tilde{\mathbf{e}}$.[1] Therefore, we study ways to simulate synthetic $\tilde{\mathbf{e}}$ from $\mathbf{e}$, while making sure to preserve large portions of the reference sentence $\mathbf{e}$ when generating samples. As string editing operations can be decomposed into a sequence of atomic operations among insertions, substitutions, and deletions, we design our artificial training samples so that edits from $\tilde{\mathbf{e}}$ to $\mathbf{e}$ only involve one single type of operation. Figure 7.2 illustrates our data generation methodology for each type of edit, along with a generated example.

7.2.1 . Insertions

We mainly follow Xiao et al. (2022a) to generate initial translations $\tilde{\mathbf{e}}_{\text{ins}}$ for insertion by randomly dropping segments from the original target sentence $\mathbf{e}$. For each $\mathbf{e}$, we first randomly sample an integer $k \in [1, 5]$, then randomly remove k non-overlapping segments from $\mathbf{e}$. The length of each removed segment is also randomly sampled with a maximum length of 5 tokens. We also impose that the overall ratio of removed segments does not exceed 0.5 of the original target $\mathbf{e}$. Unlike Xiao et al. (2022a), the initial translations $\tilde{\mathbf{e}}_{\text{ins}}$ we generated do not contain any placeholders to locate the positions of removed segments (see Figure 7.2). This makes $\tilde{\mathbf{e}}_{\text{ins}}$ a more realistic starting point as the insertion positions are rarely known in practical settings. Our preliminary experiments to reproduce Xiao et al. (2022a) also show that identifying the insertion positions in $\tilde{\mathbf{e}}_{\text{ins}}$ makes the text infilling task much easier than when they are unknown. Models trained with explicit insertion positions possess a higher precision and recall score for predicting the expected segments.

[1] A situation where such triplets actually exist is APE. However, we do not consider APE since the MT output depends on the original MT system and is sometimes difficult to improve (Chollampatt et al., 2020). The post-editions also depend on human post-editors and are often unique, inconsistent, and difficult to generalize (Wisniewski et al., 2015).

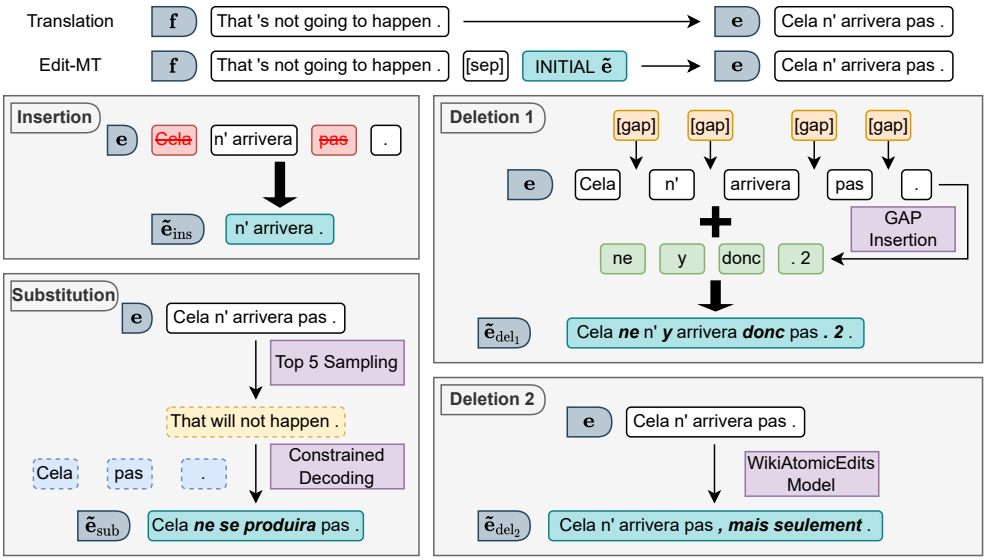


Figure 7.2: Methods for generating synthetic initial translations for each edit type. Rectangle purple boxes refer to separate models used to generate the desired operations. Differences in artificial initial translations (in blue boxes) are marked in bold. Initial translations $\tilde{\mathbf{e}}_{\text{ins}}$ for insertion are generated by randomly removing segments in the reference sentence $\mathbf{e}$. For $\tilde{\mathbf{e}}_{\text{sub}}$, $\mathbf{e}$ is first back-translated into an intermediate sentence $\mathbf{f}^*$ using top-5 sampling, then translated back to $\tilde{\mathbf{e}}_{\text{sub}}$ with LCD. The first method to generate $\tilde{\mathbf{e}}_{\text{del}}$ randomly inserts [gap] tokens into $\mathbf{e}$ and decodes with a GAP insertion model (Xiao et al., 2022a). The $\tilde{\mathbf{e}}_{\text{del}_1}$ is obtained by replacing [gap] with the predicted extra segments. The second method uses a model trained with WikiAtomicEdits data to translate $\mathbf{e}$.

7.2.2 . Substitutions

To simulate substitutions, we apply round-trip translation with lexically constrained decoding (LCD) (Post and Vilar, 2018) to generate initial translations for substitution $\tilde{\mathbf{e}}_{\text{sub}}$. Round-trip translation has already been applied for the APE task (Junczys-Dowmunt and Grundkiewicz, 2016). The generation procedure requires two standard NMT models separately trained on parallel data, one for each direction. For each training example $(\mathbf{f}, \mathbf{e})$, we first (a) translate the original target $\mathbf{e}$ into an intermediate source sentence $\mathbf{f}^*$ using top-5 sampling (Edunov et al., 2018);[2] (b) generate an abbreviated version $\tilde{\mathbf{e}}'_{\text{ins}}$ from $\mathbf{e}$, using the method described above for insertions. We then translate $\mathbf{f}^*$ using LCD, with $\tilde{\mathbf{e}}'_{\text{ins}}$ as constraints, to obtain $\tilde{\mathbf{e}}_{\text{sub}}$. In this way, we ensure that at least half of $\mathbf{e}$ remains unchanged in $\tilde{\mathbf{e}}_{\text{sub}}$, while the other parts have been substituted during round-trip translation. In order to

[2] Our preliminary experiments showed that using top-k sampling instead of beam search for decoding increased the diversity of the generated $\tilde{\mathbf{e}}_{\text{sub}}$.

increase diversity, $\tilde{\mathbf{e}}'_{\text{ins}}$ (used as lexical constraints to create $\tilde{\mathbf{e}}_{\text{sub}}$) is sampled with a different random seed than $\tilde{\mathbf{e}}_{\text{ins}}$ (used for the insertion task).

As shown in Figure 7.2 (Substitution), the target sentence $\mathbf{e}$ = [*Cela n' arrivera pas .*] is first translated into the source language $\mathbf{f}^*$ = [*That will not happen .*] using top-5 sampling. The lexical constraints we generated from $\mathbf{e}$ is $\tilde{\mathbf{e}}'_{\text{ins}}$ = [*Cela pas .*]. By translating $\mathbf{f}^*$ back to the target language with $\tilde{\mathbf{e}}'_{\text{ins}}$, we obtain $\tilde{\mathbf{e}}_{\text{sub}}$ = [*Cela* ***ne se produira*** *pas .*], where the segment [*n' arrivera*] is substituted with the equivalent [*ne se produira*].

7.2.3 . Deletions

Simulating deletions requires the initial translation $\tilde{\mathbf{e}}_{\text{del}}$ to be an extension of the original target $\mathbf{e}$. We propose two strategies to generate $\tilde{\mathbf{e}}_{\text{del}}$.

Method 1

The first method uses a GAP insertion model proposed by Xiao et al. (2022a). This approach randomly replaces word segments with a placeholder [gap] to generate an initial translation $\tilde{\mathbf{e}}_{\text{gap}}$. The task is to predict the missing segments based on a long input made as the concatenation of $\mathbf{f}$ and $\tilde{\mathbf{e}}_{\text{gap}}$. This task is different from our insertion task, as (a) insertion positions are identified as [gap] symbols in $\tilde{\mathbf{e}}_{\text{gap}}$, and (b) generation only computes the sequence of missing segments $\mathbf{e}_{\text{seg}}$, rather than a complete sentence. An interesting finding of Xiao et al. (2022a) is that the GAP model is mostly able to match the number of predicted segments to the number of placeholders in $\tilde{\mathbf{e}}_{\text{gap}}$, even though there are no specific hard constraints for this requirement.

This finding inspired our first strategy. We use the GAP insertion model to generate extra segments for a pair of parallel sentences as follows. We randomly insert $k \in [1, 5]$ [gap] tokens into the original target sentence $\mathbf{e}$, concatenate it with the source $\mathbf{f}$ and feed the input to the GAP model to predict extra segments, yielding the synthetic target sentence $\tilde{\mathbf{e}}_{\text{del}_1}$, as illustrated in Figure 7.2 (Deletion 1). Since the starting point is a pair of parallel sentences, introducing [gap] tokens forces the GAP model to predict extra segments. This method always extends parallel sentences with additional segments on the target side. However, these segments are arbitrary and may not contain any valid semantic information nor be syntactically appropriate in their contexts.

Method 2

We thus consider a second strategy based on actual edit operations collected in the WikiAtomicEdits dataset[3] (Faruqui et al., 2018). This dataset

[3] https://github.com/google-research-datasets/wiki-atomic-edits

contains edits of an original sequence x and the resulting sequence x', with exactly one insertion or deletion operation for each training example, collected from Wikipedia edit history. This notably ensures that both versions of each utterance are syntactically correct. We treat the deletion operation data of WikiAtomicEdits as "reversed" insertions and use both to train a sequence-to-sequence `wiki` model ($x_{\text{short}} \rightarrow x_{\text{long}}$), generating longer sentences from shorter ones. The `wiki` model is then used to expand the target sentence $\mathbf{e}$ into a longer version $\tilde{\mathbf{e}}_{\text{del}_2}$. Compared to $\tilde{\mathbf{e}}_{\text{del}_1}$, $\tilde{\mathbf{e}}_{\text{del}_2}$ is syntactically more correct. However, it is also, by design, very close to $\mathbf{e}$, as there is only one edit away to convert $\tilde{\mathbf{e}}_{\text{del}_2}$ to $\mathbf{e}$.

As both simulation methods for the deletion task have merits and flaws, we randomly select examples from both $\tilde{\mathbf{e}}_{\text{del}_1}$ and $\tilde{\mathbf{e}}_{\text{del}_2}$ to build the final synthetic initial translation samples for the deletion operation $\tilde{\mathbf{e}}_{\text{del}}$.

7.2.4 . Copy and Translate Operations

In order to handle parallel sentences that would not require any changes, we add a fourth *copy* operation, where the initial translation $\tilde{\mathbf{e}}_{\text{copy}}$ is identical to the target sentence ($\tilde{\mathbf{e}}_{\text{copy}} = \mathbf{e}$). For the copy operation, the model is expected to generate the exact initial target $\tilde{\mathbf{e}}_{\text{copy}}$ as output without making changes. Therefore, the data eventually used to learn edit operations is built with triplets $(\mathbf{f}, \tilde{\mathbf{e}}, \mathbf{e})$ where $\tilde{\mathbf{e}}$ is uniformly randomly sampled from $\tilde{\mathbf{e}}_{\text{ins}}$, $\tilde{\mathbf{e}}_{\text{sub}}$, $\tilde{\mathbf{e}}_{\text{del}}$, and $\tilde{\mathbf{e}}_{\text{copy}}$. The training data thus contain the same amount of data for each edit operation. Finally, we consider training examples where $\tilde{\mathbf{e}}$ is empty to also maintain the capacity to perform standard MT from scratch.

7.3 . Edit-MT

We implement Bi-sync with a Transformer-based (Vaswani et al., 2017) autoregressive model called Edit-MT. In this model, the initial translation $\tilde{\mathbf{e}}$ we generated is simply concatenated to the source $\mathbf{f}$, with a special token to separate the two sentences. This technique has been used, *e.g.*, in (Crego et al., 2016; Niehues et al., 2016; Hokamp, 2017) for APE, in (Dabre et al., 2017) for multi-source MT, or in (Bulte and Tezcan, 2019) for translation with TMs, as introduced in Chapter 3. The input side of the editing training data is thus $\mathbf{f}$`[sep]`$\tilde{\mathbf{e}}$, as shown in Figure 7.2 (top). Note that we do not reset the positional encoding for the second part $\tilde{\mathbf{e}}$, as it does not yield significant effects on the overall performance (Yang et al., 2022).

On the target side, we add a categorical prefix to indicate the type of edit operation(s) associated with a given training sample, as is commonly done for multi-domain (Kobus et al., 2017) or multilingual MT (Johnson et al., 2017) and is also applied in Sections 5.4.2 and 6.2.2. For each basic

edit (insertion, substitution, and deletion), we use a binary tag to indicate whether this operation is required. For instance, an initial translation needing insertions $\tilde{\mathbf{e}}_{\text{ins}}$ would have tags `[ins] [!sub] [!del]` prepended to the target $\mathbf{e}$, indicating that only insertion is required while substitution and deletion are not. The copy operation corresponds to the case where all three tags are set to negative as `[!ins] [!sub] [!del]`. Note that we use tags on the target side, rather than on the source side, as Johnson et al. (2017). This tagging scheme on the target side provides us with various ways to perform edit-based MT:

- We can perform direct inference without knowing the editing type associated with $\tilde{\mathbf{e}}$. In this case, Edit-MT evaluates itself the required edition types. The decoding procedure starts by first generating editing tags for the predicted edition type, then producing the corresponding edited translation.
- When the operations are known, we can use the corresponding tags as a forced prefix to generate a translation with desired edits. Edit-MT thus is indicated by a required edition type. We can generate more controllable edited translations by forcing Edit-MT with certain editions. In a bilingual writing system, we can obtain the required operations by storing edit histories and computing the text differences in one human revision.
- The inference procedure can also only generate the output editing tags and predict the relationship between the source $\mathbf{f}$ and the initial translation $\tilde{\mathbf{e}}$. For example, we can predict parallelism when the system outputs the copy tag `[!ins] [!sub] [!del]`. In this situation, the decoding is truncated to a maximum length of 3 to ignore the translation part.

For Edit-MT, the ability to perform standard MT is also preserved by training with a balanced mixture of artificial editing data and regular parallel data, as in (Bulte and Tezcan, 2019; Xu et al., 2021c). The regular translation data corresponds to an empty initial translation $\tilde{\mathbf{e}}$ = [ ]. For these examples, the target side does not contain any tags.

7.4 . Bilingual Synchronization

7.4.1 . Datasets

We first evaluate Edit-MT on the general Bi-sync task where $\tilde{\mathbf{e}}$ is assumed to be the translation of a former version of the source sentence $\tilde{\mathbf{f}}$, and only a limited number of edits is sufficient to restore the parallelism. We conduct experiments on WMT14 English-French data[4] in both directions (En-Fr &

[4] https://www.statmt.org/wmt14

Fr-En) and evaluate on two test sets. The first is an artificial derivation of the standard `newstest2014` set, and the second is a small parallel sentence compression dataset[5] of Ive and Yvon (2016).

For the artificial News task, we generate the required $\tilde{\mathbf{e}}$ for each editing operation according to the methods used in Section 7.2, resulting in four versions (Ins, Sub, Del_1, Del_2) of `newstest2014`, with each containing $3{,}003$ sentences. The sentence compression dataset contains a subset of documents also selected from `newstest2014`, where sentences from both languages are manually compressed by human annotators while remaining parallel in the two languages. We only retain utterances where the compressed and uncompressed versions actually differ on both sides, resulting in 526 test sentences. Table 7.1 summarizes the main statistics for training and test data.

Datasets	# Sentences
Training	33.9M
`newstest2014`	3,003
Compression (Ive and Yvon, 2016)	526

Table 7.1: Statistics of training and test data for Bi-sync.

We discard training examples with invalid language tags as computed by `fasttext` language identification model[6] (Bojanowski et al., 2017), yielding a training corpus of 33.9M examples. The parallel training data is identical to what we use in Chapter 5. We tokenize all data using Moses tokenizer and build a shared source-target vocabulary with 32K Byte Pair Encoding (BPE) units (Sennrich et al., 2016c) learned with `subword-nmt`.[7] Since we use both parallel and artificial editing data to train Edit-MT models, the total training data contains about 68M utterances.

7.4.2 . Experimental Settings

We implement our Edit-MT model and conduct experiments using the `fairseq`[8] toolkit (Ott et al., 2019). Edit-MT relies on the Transformer-base model of Vaswani et al. (2017). We use a hidden size of 512 and a feed-forward size of $2{,}048$. We optimize using Adam with a maximum learning rate of 0.0007, an inverse square root decay schedule, and $4{,}000$ warm-up steps. All input and output embedding matrices are tied (Press and Wolf, 2017; Inan et al., 2017), and Edit-MT is trained with mixed precision and

[5] https://github.com/fyvo/ParallelCompression

[6] https://dl.fbaipublicfiles.com/fasttext/supervised-models/lid.176.bin

[7] https://github.com/rsennrich/subword-nmt

[8] https://github.com/pytorch/fairseq

a batch size of $8{,}192$ tokens on 4 V100 GPUs for 300k iterations. We save checkpoints for every $3{,}000$ iterations and average the last 10 saved checkpoints for inference.

For comparison, we report a "do-nothing" baseline by simply copying the initial translation $\tilde{\mathbf{e}}$ as the output. Performance is computed with SacreBLEU (Post, 2018).

7.4.3 . Main Results

We first evaluate the ability of Edit-MT to perform standard translation. We use the original `newstest2014` for both directions. The source side for Edit-MT contains an empty $\tilde{\mathbf{e}}$. We copy the `base-mono` results from Section 5.5 as baselines since these models are trained and tested on the same WMT14 data as in this section. From Table 7.2, we can observe that Edit-MT only performs slightly worse than the baseline, with -1.9 BLEU for En-Fr and -0.5 BLEU for Fr-En, which is expected, as also pointed out by Bulte and Tezcan (2019). These results suggest that Edit-MT can perform standard translations with an acceptable loss of performance compared to baseline translation models. Therefore, in the rest of this section, we focus on the Bi-sync task.

Model	En-Fr	Fr-En
`base-mono`	37.6	35.2
Edit-MT	35.7	34.7

Table 7.2: BLEU scores of Edit-MT performing standard translation on `newstest2014` for both En-Fr and Fr-En.

We separately evaluate the learnability of each editing operation. For this, we generate an initial version $\tilde{\mathbf{e}}$ for each test sentence and each editing operation so that we have $3{,}003$ test samples per operation. For deletion, we test the performance of both generation methods introduced in Section 7.2.3. We also derive two tasks from the compression test set: parallel sentence compression (comp) and extension (ext). For compression, the task consists of generating the compressed target sentence $\mathbf{e}_{\text{comp}}$, given the compressed source $\mathbf{f}_{\text{comp}}$ and the original target $\mathbf{e}$. For extension, the model is expected to produce $\mathbf{e}$ with the original source $\mathbf{f}$ and the compressed target $\mathbf{e}_{\text{comp}}$. These two tasks are respectively similar to the deletion and the insertion settings. There are slight differences, in any case. The initial translation $\tilde{\mathbf{e}}$ for these settings is always syntactically correct, and the removed or inserted segments are selected for their lower informativeness. In this respect, these tasks are more about restoring an adequate rather than a fluid translation.

As mentioned in Section 7.3, the generation in Edit-MT models is conditioned on predicted or oracle editing tags that are prefixed to the output. We

Model	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	54.0	71.5	71.0	78.7	68.4	66.7
Edit-MT	75.9	77.0	86.9	**94.7**	73.1	67.9
+ tag	**76.9**	**78.5**	**88.6**	**94.7**	**74.0**	**72.7**

Table 7.3: BLEU scores of Edit-MT on Bi-sync for En-Fr. Deletions are evaluated separately for the two generation methods (Del_1 and Del_2). + *tag* refers to decoding with the oracle tag as a forced prefix. The best performance is in bold.

Model	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	51.8	70.9	71.0	78.7	63.4	61.4
Edit-MT	73.6	74.6	87.5	95.8	65.8	69.3
+ tag	**74.6**	**76.2**	**89.1**	**96.2**	**67.0**	**71.6**

Table 7.4: BLEU scores of Edit-MT on Bi-sync for Fr-En.

evaluate both situations. The oracle tag setting (+ tag) uses forced-prefix decoding with the correct tags. For the compression and extension tasks, we use the deletion and insertion tags, respectively.

Tables 7.3 and 7.4 report results for both directions, to be contrasted with a baseline score corresponding to simply copying $\tilde{\mathbf{e}}$ as the output. For this experiment, Edit-MT is able to edit the given $\tilde{\mathbf{e}}$ for all types of required edits. It obtains large gains over the copy baseline for insertion,[9] substitution, and deletion for both translation directions. When tested on the more realistic compression and extension tasks, which have different edit distributions to the artificial training data, Edit-MT can also improve $\tilde{\mathbf{e}}$ by 1.2-4.7 BLEU for En-Fr and 2.4-7.9 BLEU for Fr-En. Note that in the above settings, Edit-MT predicts the required editing type for each test sentence. By prefixing Edit-MT with the oracle editing type tags (+ tag), we can further increase the performance on almost every task in both directions. The effect of editing tags is further analyzed in Section 7.4.5.

7.4.4 . Multilingual Edit-MT

The general purpose of Bi-sync is to restore parallelism between two sentences. In the previous section, we have decomposed the synchronization directions into two separate models. The En-Fr model only edits initial French translations, and the Fr-En model only performs editing in the other direction. Ideally, we would like our model to perform Bi-sync in both directions within a single model, treating the two languages without any difference. In this section, we thus study the multilingual setting of our model.

[9] BLEU gains in insertion are artificially high. This is because the copy baseline is hindered by a high brevity penalty.

En-Fr	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	54.0	71.5	71.0	78.7	68.4	66.7
Edit-MT (En-Fr)	75.9	77.0	86.9	94.7	73.1	67.9
+ tag	**76.9**	**78.5**	**88.6**	94.7	**74.0**	72.7
`multi` Edit-MT	75.5	77.2	86.9	94.7	72.3	68.4
+ tag	76.2	78.1	88.5	**94.9**	72.9	**73.4**
Fr-En	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	51.8	70.9	71.0	78.7	63.4	61.4
Edit-MT (Fr-En)	73.6	74.6	87.5	95.8	65.8	69.3
+ tag	**74.6**	**76.2**	**89.1**	**96.2**	**67.0**	**71.6**
`multi` Edit-MT	73.5	75.1	86.7	95.8	64.6	68.0
+ tag	74.2	76.0	88.7	**96.2**	65.4	70.8

Table 7.5: BLEU scores of multilingual Edit-MT on Bi-sync for both En-Fr and Fr-En. The multilingual model denoted with `multi` is the same model used for both directions.

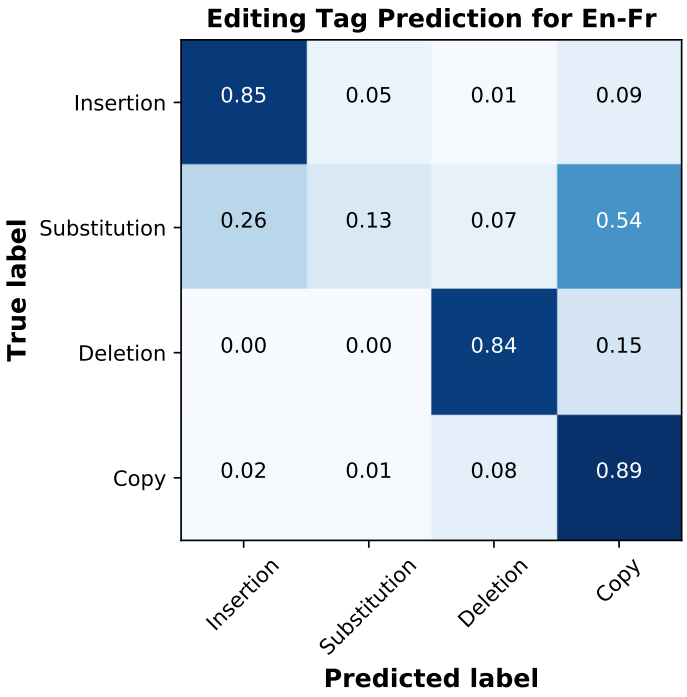
We swap the source sentence $\mathbf{f}_{\text{fr}}$ and the initial translation $\tilde{\mathbf{e}}_{\text{en}}$ for the Fr-En direction so that English sentences always appear first, regardless of being expected to be edited. The source side is therefore composed of training examples as $\mathbf{f}_{\text{en}}$ `[sep]` $\tilde{\mathbf{e}}_{\text{fr}}$ and $\tilde{\mathbf{e}}_{\text{en}}$ `[sep]` $\mathbf{f}_{\text{fr}}$. For the target side, we further prefix a language tag to indicate the desired output language, *i.e.*, the language to be edited. For instance, a French sentence needing substitutions will be prefixed with `[fr] [!ins] [sub] [!del]` to the target sentence. During inference, we always prefix the language tag to require the desired languages. The language tag is necessary for multilingual Edit-MT, as we can arbitrarily choose to edit either language to make it parallel to the other. We double the batch size when training the multilingual Edit-MT model.

As can be seen in Table 7.5, multilingual Edit-MT delivers similar results compared to its monolingual counterparts in both directions within a single model. This implies that we can effectively treat both languages equally when performing Bi-sync. In addition, multilingual models can save half of the deployment resources in real-world applications, which is always preferred over deploying two separate models.

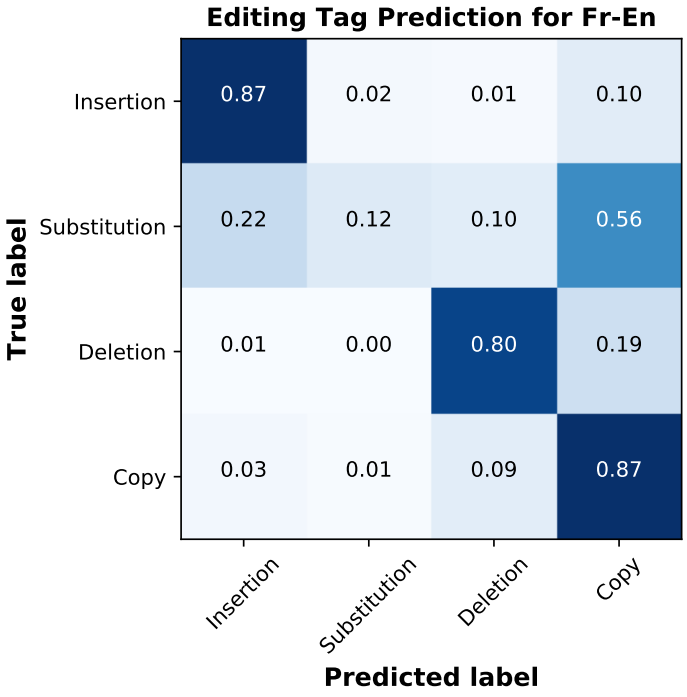
7.4.5 . The Effect of Editing Tags

We discussed in Section 7.4.3 that simply prefixing Edit-MT with the oracle editing tags during inference can improve the performance in all cases. In this section, we conduct analyses to better understand the effect of editing tags.

We first evaluate the accuracy of editing tag prediction by taking the tags predicted during the synchronization process of Edit-MT for the basic editions tasks (Ins, Sub, Del_1, Del_2). The two deletion test sets are combined as one,



(a)



(b)

Figure 7.3: Confusion matrices for predicted tags of Edit-MT for both directions.

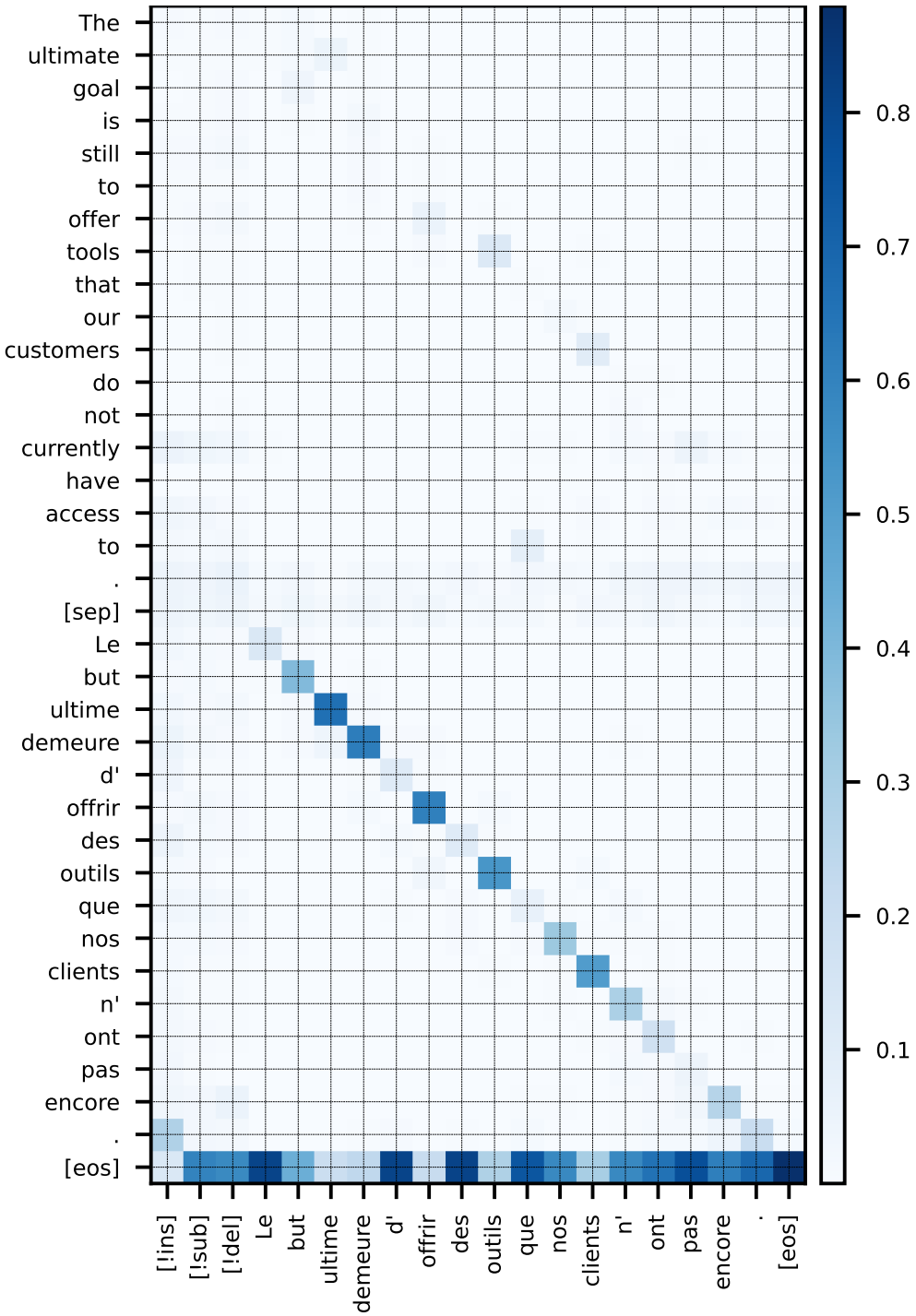


Figure 7.4: Example attention matrix of Edit-MT when translating **without** forced oracle tag. The model generates a copy tag and then copies the initial translation as output.

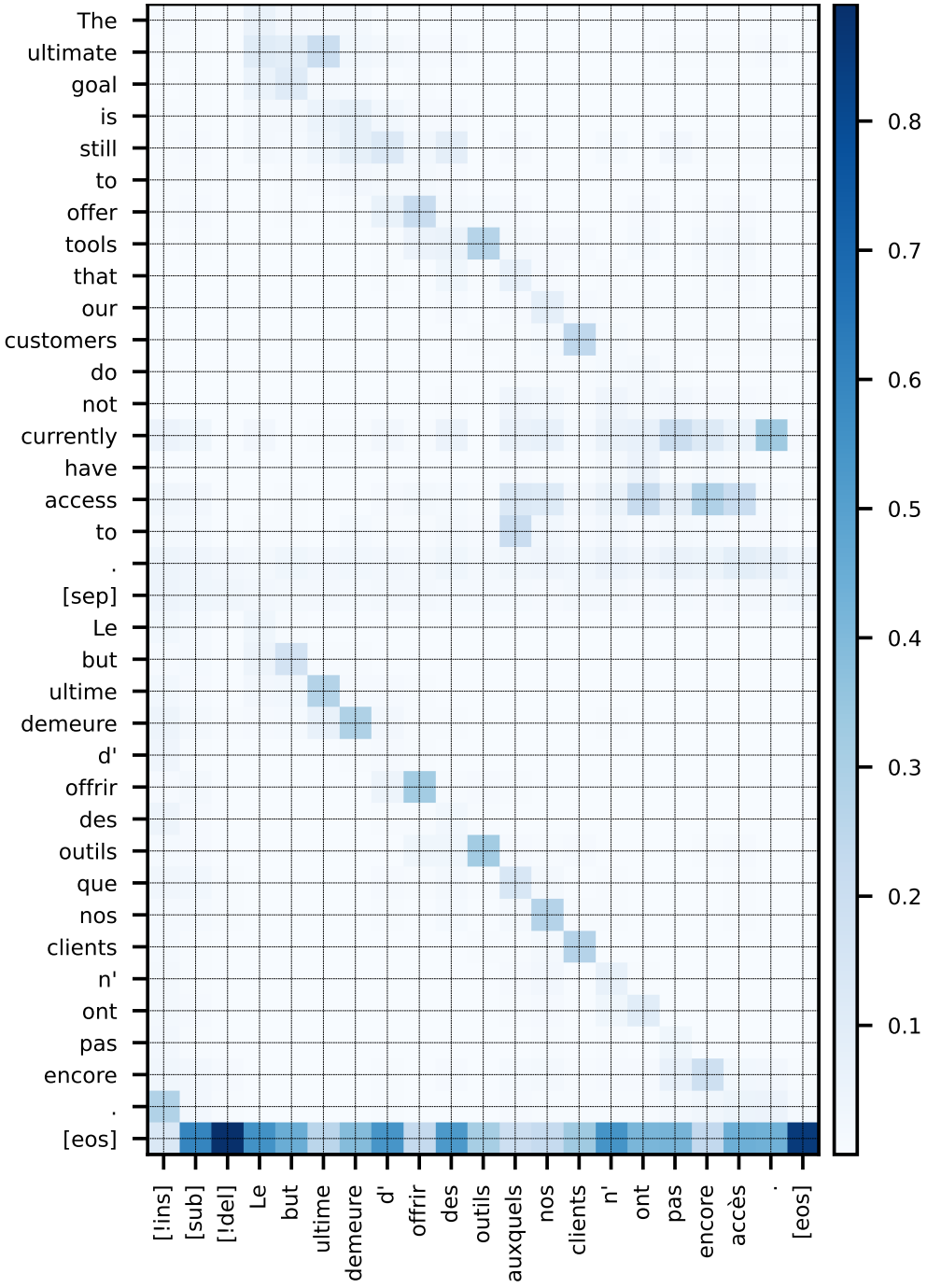


Figure 7.5: Example attention matrix of Edit-MT when translating **with** a forced substitution tag. The model properly replaces “que” with “auxquels” by paying more attention to “to” in the source and adds “accès” (access) to generate a good translation following the oracle tag.

as they share the same editing tag. To gain a better understanding of the editing tag prediction, we also consider a *copy* operation, as we train Edit-MT with four different tags. The *copy* test set is built by simply concatenating the source sentence $\mathbf{f}$ with the reference target $\mathbf{e}$ as input. Edit-MT is thus supposed to predict the copy tag for this case, then produce the reference sentence unchanged. We use tags predicted by the monolingual models for each direction and plot the confusion matrix of the predicted tags for each edition type in Figure 7.3. Results show that Edit-MT generally predicts correctly for insertion, deletion, and copy operations. However, it fails to make precise predictions for sentences requiring substitution and wrongly considers half of the sentences as already parallel.

We further analyze the effect of using forced oracle tags by showing the attention weights computed by Edit-MT during inference. We use the encoder-decoder cross-attention weights of the last decoder layer. The attention weights are averaged over all heads (*cf.*, Section 2.2.1). Figure 7.4 illustrates the attention weights when editing an initial translation requiring substitutions. We do not prefix anything, and Edit-MT incorrectly predicts the copy tag and leaves the output unchanged. The attention weights are mostly concentrated on the initial translation part $\tilde{\mathbf{e}}$ of the input. In this case, Edit-MT almost ignores the original source $\mathbf{f}$ and only looks at $\tilde{\mathbf{e}}$ to simply copy the output.

When prefixing Edit-MT with the correct substitution tag for the same input, the model is thus able to make proper changes. As shown in Figure 7.5, Edit-MT correctly substitutes "que" with "auxquels" and adds "accès", which corresponds to "access" in the source $\mathbf{f}$ but does not appear in $\tilde{\mathbf{e}}$. In this case, Edit-MT performs actual translation by looking at both the source tokens and the initial target tokens at the same time, even when a token should be directly copied.

7.5 . Translating with Translation Memories

As explained in Section 7.1, Bi-sync encompasses TM-based MT, whereby an existing similar translation retrieved from a TM is turned into an adequate translation of the source. As introduced in Section 3.3, here TM simply refers to parallel bilingual data. Edit-MT actually uses the same architecture as the retrieval-based model of Bulte and Tezcan (2019). In this section, we study the performance of our Edit-MT model in this practical scenario.

7.5.1 . Related Words in Translaton Memory Matches

Given a source sentence $\mathbf{f}$ and a TM match pair $(\tilde{\mathbf{f}}, \tilde{\mathbf{e}})$ in which $\tilde{\mathbf{f}}$ is similar to $\mathbf{f}$, apart from directly using the complete sequence of the retrieved

similar translation $\tilde{\mathbf{e}}$ as input, we also propose an alternative where we remove segments in $\tilde{\mathbf{e}}$ that are not aligned with the input source $\mathbf{f}$. Considering the example in Section 3.3, the segments [*the flight*] and [*a cold*] are not related to each other on the source language side. From that, the target segment [*le vol*] is thus irrelevant to translating the source sentence $\mathbf{f}$. In this section, we discuss an algorithm capable of identifying the set of target tokens $\mathcal{T} \in \tilde{\mathbf{e}}$ that are related to tokens in the source sentence $\mathbf{f}$. We define the related target tokens set $\mathcal{T}$ as:

$$\mathcal{T} = \left\{ \begin{array}{l} e_j \in \tilde{\mathbf{e}} : \\ \quad \exists \tilde{f}_i \in \mathcal{LCS} \mid (\tilde{f}_i, \tilde{e}_j) \in \mathcal{A} \\ \wedge \quad \forall \tilde{f}_i \notin \mathcal{LCS} \mid (\tilde{f}_i, \tilde{e}_j) \notin \mathcal{A} \end{array} \right\}, \tag{7.1}$$

where $\mathcal{A}$ is the set of word alignments between words in $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{e}}$, and $\mathcal{LCS}$ is the set of words in $\tilde{\mathbf{f}}$, which belong to the Longest Common Subsequence (LCS) between $\tilde{\mathbf{f}}$ and $\mathbf{f}$. The LCS is computed as a by-product of the edit distance (Paterson and Dančík, 1994). Word alignments are computed by `fast_align`[10] (Dyer et al., 2013).

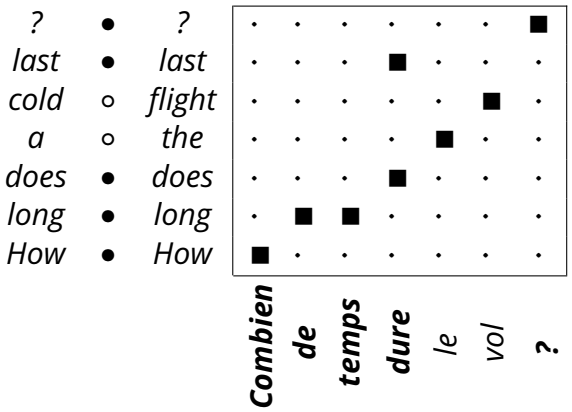


Figure 7.6: TM entries with the corresponding $\mathcal{LCS}$ of words with the source sentence (left) and word alignments (right). Black dots and squares indicate matches of LCS and word alignments, respectively. Related target tokens are in bold.

Figure 7.6 illustrates the set of related words $\mathcal{T}$, along with the $\mathcal{LCS}$ between $\mathbf{f}$ and $\tilde{\mathbf{f}}$ and the alignments between $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{e}}$. The TM source sentence $\tilde{\mathbf{f}}$ has an $\mathcal{LCS}$ set of 5 tokens {*How, long, does, last, ?*}. The set of related target words $\mathcal{T}$ is also composed of 5 tokens {*Combien, de, temps, dure, ?*}, all aligned to at least one word in $\mathcal{LCS}$ and to no other word. Note that both $\mathcal{LCS}$ and $\mathcal{T}$ consist of collections of indices (word

[10] https://github.com/clab/fast_align

positions in their corresponding sentences) while word strings are used in the examples to facilitate reading. When applying this *related* setting, the actual sequence used as a similar translation is thus $\tilde{\mathbf{e}}$ = [*Combien de temps dure ?*]. This is structurally very similar to the insertion data introduced in Section 7.2.

7.5.2 . Datasets

Our experiments use a multi-domain corpus. This corpus contains 11 different domains for the En-Fr direction, collected from OPUS[11] (Tiedemann, 2012): documents from the European Central Bank (ECB); from the European Medicines Agency (EME); Proceedings of the European Parliament (Epp); legislative texts of the European Union (JRC); News Commentaries (News); TED talk subtitles (TED); parallel sentences extracted from Wikipedia (Wiki); localization files (GNOME, KDE, and Ubuntu) and manuals (PHP). All these data were deduplicated prior to training. To evaluate the ability of our models to actually make use of TMs instead of memorizing training examples, we also test on two unseen domains: OpenOffice (Office) from OPUS and the PANACEA environment corpus[12] (ENV).

For each source sentence, we retrieve from the same domain the top 3 TM matches according to the fuzzy match score described in Equation (3.1) based on the Fuzzy-Match implementation.[13] As we are mainly interested here in the editing behavior, we only keep TMs having a sufficiently similar translation but without exact matches on the source side, *i.e.*, requiring a fuzzy match score $\mathrm{FM} \in [0.6, 1)$. We then split the data by keeping $1{,}000$ sentences with at least one match as the test set for each domain. The remaining data is used for training. Note that the ratio of sentences with at least one similar translation greatly varies across domains. Detailed statistics about these corpora are shown in Table 7.6. We use all retrieved (up to 3) TM matches for training and only the best match for testing. The initial set of 4.4M parallel sentences (*para*) is thus augmented with about 2.6M examples for which a good TM match is available (*similar*) or just the related segments (*related*). Data preprocessing is the same as in Section 7.4.1, and we build a shared source-target vocabulary with 32K BPE units (Sennrich et al., 2016c).

7.5.3 . Experimental Settings

We consider two baseline settings for TM-based MT: the `FM` setting of Bulte and Tezcan (2019) and the $\texttt{FM}^{\#}$ setting described in Section 7.5.1.

[11] https://opus.nlpl.eu/

[12] http://catalog.elda.org/en-us/repository/browse/ELRA-W0057/

[13] https://github.com/SYSTRAN/fuzzy-match

Domain	Raw	FM ratio	FM train
ECB	195,956	51.73%	234,943
EME	373,235	65.68%	624,109
Epp	2,009,489	10.12%	465,228
GNO	55,391	39.31%	42,697
JRC	503,437	50.87%	587,859
KDE	180,254	36.00%	136,456
News	151,423	2.12%	4,048
PHP	16,020	34.93%	10,350
TED	159,248	11.90%	39,895
Ubu	9,314	20.32%	1,738
Wiki	803,704	19.87%	409,755
Total	4,457,471	24.27%	2,557,078
Office	49,845	43.76%	-
ENV	13,632	6.81%	-

Table 7.6: Data used for experiments in Section 7.5. *FM ratio* is the ratio of sentences with at least one matched similar translation. *FM train* is the actual number of examples with a similar translation used for training after setting aside $1{,}000$ test sentences. Each train sentence is matched with up to 3 similar translations. GNO and Ubu refer to GNOME and Ubuntu, respectively.

Both settings concatenate the source sentence $\mathbf{f}$ and the similar translation or similar segments $\tilde{\mathbf{e}}$ as the source input, while the model architecture is an unchanged Transformer model. The `FM` model is trained using *para* + *similar* data, and the `FM`$^{\#}$ uses *para* + *related*. These two baselines are trained with the same configuration as in Section 7.4.2. We also report the "do-nothing" scores obtained by simply copying the retrieved similar translations as the output, as also done in Section 7.4.3.

The Edit-MT models trained in Section 7.4 differ from `FM` and `FM`$^{\#}$ in both the task and the domains used for training. Hence, we consider reusing the generic Edit-MT model and fine-tuning it on this task to evaluate the capability of our model to generalize to this MT with TMs task. We use *para* + *similar* + *related* data and fine-tune the Edit-MT En-Fr model for only 1 epoch with a fixed learning rate of $8e^{-5}$. As we do not have any information about the expected editing operations required to change a similar translation into the reference, we set all editing tags as positive for *similar* data and prefix the output with `[ins] [sub] [del]`. We conjecture that mostly insertions are required for the *related* data as the irrelevant parts of $\tilde{\mathbf{e}}$ have already been removed. We thus simply activate the insertion tag. Note that our fine-tuned model can perform translations with both similar sentences and related segments within the same model. We evaluate both BLEU and TER scores using SacreBLEU.

7.5.4 . Results

BLEU ↑	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	59.8	64.5	34.4	70.3	67.6	55.3	12.0	38.6	30.8	51.6	47.4	52.6
FM	59.6	54.8	55.2	55.8	69.3	54.6	26.9	39.8	61.6	56.9	65.9	56.7
FM$^{\#}$	61.2	56.8	55.8	57.5	70.3	55.8	27.9	40.2	61.3	58.9	54.9	57.7
Edit-MT	44.6	32.8	51.9	37.2	54.9	28.1	25.2	33.8	47.8	36.5	34.5	41.8
+ FT	53.3	48.9	53.7	50.0	64.3	47.3	26.6	36.3	59.2	49.6	59.3	51.8

Table 7.7: BLEU scores by performing standard translation **without** using TMs on test sets from multiple domains. *All* is computed by concatenating test sets from all domains, with 11k sentences in total. *Copy* refers to copying the retrieved similar translations to the output. + *FT* refers to the model fine-tuned on the multi-domain TM data.

We first perform standard translation on the multi-domain test sets without using the retrieved similar translations. As shown in Table 7.7, both FM and FM$^{\#}$ yield strong performance as they are specifically trained on the in-domain data. Our generic Edit-MT model performs much worse in general when performing zero-shot translations on these test sets. However, the performance greatly varies across domains. For instance, the generic Edit-MT performs similarly to FM on the News domain, as it is closer to the WMT data. The overall poorer performance of Edit-MT indicates the important domain mismatch between the generic WMT data and the in-domain data. However, by performing fine-tuning, Edit-MT can vastly improve its performance. Note that there is still a gap of about 5 BLEU points between the fine-tuned Edit-MT model and FM model, indicating that fine-tuned Edit-MT approaches specifically trained models but are still weaker. In the rest of this section, we focus on performing translations with TMs.

Results in Tables 7.8 and 7.9 reproduce the overall good performance of FM and show that FM$^{\#}$ also significantly improves the copy baseline, even though the gains are smaller than FM. The generic Edit-MT performs much worse and does not even match the copy results in BLEU. TER results, however, show that even this model actually identifies useful edits, as we see improvements with respect to the copy baseline. When prefixed with the editing tag[14] (+ tag), we observe minor improvements ($+0.8$ BLEU on average) that are further increased in the *related* scenario (+ R + tag). Fine-tuning gives us a much more significant performance boost ($+13.4$ BLEU, -0.142 TER points on average). The differences between generic and fine-tuned models highlight the effect of the task and domain mismatches on our

[14]We conjecture with a substitution tag for zero-shot inference as we have no information about the required edit type. Fine-tuned model uses the same tag as fine-tuning data.

initial results. The *related* setting also benefits from fine-tuning, albeit by a smaller margin (+9 BLEU).

BLEU ↑	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	59.8	64.5	34.4	70.3	67.6	55.3	12.0	38.6	30.8	51.6	47.4	52.6
FM	**72.1**	**72.3**	**58.3**	**80.6**	**83.2**	**66.9**	28.0	**47.2**	**62.9**	**69.3**	68.8	**67.3**
FM$^{\#}$	69.3	68.1	58.2	74.2	80.1	65.2	**28.6**	44.3	62.6	68.1	**69.0**	65.0
Edit-MT	59.3	62.5	34.7	69.8	68.0	50.6	12.1	38.0	31.2	52.3	45.6	51.8
+tag	60.3	63.0	35.7	70.3	68.4	51.9	12.9	38.8	32.6	52.1	45.6	52.6
+R+tag	56.0	53.9	45.9	64.9	68.5	50.0	17.7	39.7	44.9	59.9	52.8	53.3
+FT+tag	**70.6**	**71.5**	**57.8**	**78.2**	**82.0**	**66.2**	**28.0**	**45.1**	**61.1**	**67.7**	**66.8**	**66.0**
+FT+R+tag	66.4	63.6	57.3	71.3	77.6	60.5	**28.0**	42.1	60.9	66.7	65.0	62.3

Table 7.8: BLEU scores on test sets from multiple domains when translating **with** TMs. + *R* implies using the related segments instead of a full initial sentence for inference. + *FT* refers to the model fine-tuned on the multi-domain TM data. The best performance in each block is in bold.

TER ↓	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	.435	.377	.659	.263	.294	.457	.999	.703	.653	.372	.502	.488
FM	**.286**	**.301**	.374	**.160**	**.138**	.312	.642	**.572**	.328	**.227**	**.306**	**.314**
FM$^{\#}$	.299	.332	**.365**	.202	.152	**.309**	**.634**	.605	**.327**	.234	.309	.327
Edit-MT	.418	.390	.641	.260	.285	.455	.904	.646	.643	.359	.520	.472
+tag	.412	.382	.638	.256	.283	.452	.899	.643	.635	.360	.520	.468
+R+tag	.397	.422	.508	.268	.237	.428	.825	.636	.520	.282	.457	.430
+FT+tag	**.300**	**.308**	.376	**.182**	**.146**	**.314**	.643	**.595**	.351	**.237**	**.329**	**.326**
+FT+R+tag	.322	.357	**.375**	.219	.168	.342	**.640**	.623	**.350**	.238	.338	.346

Table 7.9: TER scores on test sets from multiple domains when translating **with** TMs.

Our best results overall, using fine-tuning, are superior to FM$^{\#}$ and close to that of FM. Note that our model is much worse than the two baseline models when performing standard translations. This has practical implications since FM$^{\#}$ and FM are specifically trained on the in-domain data to transform a retrieved translation, whereas the generic Edit-MT is initially trained with artificial edits and then only slightly fine-tuned on the in-domain data. The results show that fine-tuned Edit-MT can better use retrieved TMs than baseline models under the same architecture, as Edit-MT is pre-trained on large-scale data that may contain more varieties of editions than the in-domain data, which is relatively small.

To appreciate this difference, we directly test our models on the two unseen domains (Office and ENV). These domains are neither used to train FM and FM$^{\#}$ nor to fine-tune Edit-MT. Results in Table 7.10 unambiguously show that in this setting, the fine-tuned Edit-MT even outperforms the strong FM model. This suggests that our Edit-MT model has adapted not only to the domain but also the task, as it can effectively perform zero-shot TM-based

translation on unseen domains. The improvements indicate that Edit-MT has the potential to perform translations with TMs as a pre-trained model. The FM model requires large amounts of training examples with retrieved similar translations, which is difficult to find for some domains like News. However, training data for Edit-MT are synthetic and can be generated for all training data in arbitrary amounts. With pre-trained Edit-MT models, even a small amount of similar translations can benefit from a fine-tuning procedure, yielding an increase in performance.

Model	Office	ENV
copy	54.7	59.6
FM	**66.8**	**75.4**
$\text{FM}^{\#}$	64.0	70.6
Edit-MT + tag	56.2	60.3
+ FT + tag	**68.6**	**78.6**

Table 7.10: BLEU scores on unseen domains when translating with TMs.

7.5.5 . Analyses

B↑ $\Delta(\tilde{\mathbf{e}}, \mathbf{e})$	0	1	2	3	4	5	6	7	8-10	>10
N	540	2096	1107	882	827	782	689	607	1193	2277
copy	100.0	82.3	74.2	67.2	62.1	51.7	50.5	48.2	40.7	33.8
FM	91.6	**93.3**	**86.5**	**82.3**	**79.2**	**70.5**	**69.0**	**68.0**	**60.9**	**49.5**
Edit-MT+tag	95.3	80.8	72.9	68.0	62.7	52.4	50.7	49.6	41.5	34.3
+FT+tag	91.6	91.1	85.8	80.9	77.7	68.8	68.4	66.6	59.0	48.0
AD↓ $\Delta(\tilde{\mathbf{e}}, \mathbf{e})$	0	1	2	3	4	5	6	7	8-10	>10
copy	0.000	1.000	2.000	3.000	4.000	5.000	6.000	7.000	8.854	23.197
FM	1.600	**0.561**	**1.321**	**1.827**	**2.242**	**2.978**	**3.598**	**4.104**	**5.285**	**15.468**
Edit-MT+tag	0.850	1.210	2.266	3.008	3.959	4.937	5.888	6.741	8.578	21.409
+FT+tag	1.493	0.770	1.397	1.954	2.435	3.265	3.671	4.254	5.513	15.792

Table 7.11: BLEU (B) scores and average edit distance (AD) broken down by the distance Δ between $\tilde{\mathbf{e}}$ and $\mathbf{e}$. Each column represents a range of distances. N denotes the number of sentences in each group.

To better understand Edit-MT on TM-based translation, we further analyze its performance grouped by the difference Δ between $\tilde{\mathbf{e}}$ and $\mathbf{e}$ (computed as edit distance): Δ measures the editing effort needed to turn $\tilde{\mathbf{e}}$ into $\mathbf{e}$. For the results in Table 7.11, all 11k test sentences are merged into one test set, further broken down by values of Δ. Since translating with a complete TM sentence performs better than the *related* setting, we only take the complete TM setting for the following analyses. Apart from computing the BLEU scores, we also compute the average edit distance (AD) between systems' outputs and the references. The generic Edit-MT improves the

BLEU↑	=	I	S	D	I+S	I+D	S+D	I+S+D	All
N	540	316	3260	316	2865	58	2634	1011	11000
copy	100.0	72.0	67.9	75.4	32.5	69.8	34.0	47.3	52.6
FM	91.6	**80.6**	**86.6**	82.9	**50.0**	67.4	**58.4**	**63.0**	**67.3**
Edit-MT+tag	95.3	75.7	67.0	77.2	34.2	68.5	37.4	48.0	52.6
+FT+tag	91.6	79.7	84.6	**85.8**	48.3	**69.9**	57.6	60.8	66.0
AD↓	=	I	S	D	I+S	I+D	S+D	I+S+D	All
copy	0.000	3.791	2.336	2.744	10.696	8.621	10.555	17.125	7.813
FM	1.600	**3.038**	**1.275**	2.016	**8.002**	7.172	**5.436**	**11.389**	**5.072**
Edit-MT+tag	0.850	3.725	2.528	2.535	10.539	8.690	9.152	16.756	7.493
+FT+tag	1.493	3.225	1.501	**1.557**	8.287	**6.621**	5.572	11.763	5.264

Table 7.12: BLEU scores and average edit distance (AD) broken down by the editing operations required between $\tilde{\mathbf{e}}$ and $\mathbf{e}$. Each column represents a combination of edits. I, S, D refer to insertion, substitution, and deletion, respectively.

similar translation in terms of both BLEU and AD for $\Delta \geq 3$. However, it is difficult for Edit-MT to detect minor changes ($\Delta < 3$) without fine-tuning. Once fine-tuned, Edit-MT performs similarly to FM for small changes, which further confirms that Edit-MT adapts to the TM-based translation task.

We also break down the aggregate test set by editing operations required to transform $\tilde{\mathbf{e}}$ to $\mathbf{e}$. Table 7.12 shows that the generic Edit-MT struggles to perform well when substitutions are needed. Fine-tuning vastly improves the ability to substitute and delete from $\tilde{\mathbf{e}}$, separately or even in combination. Fine-tuned Edit-MT even outperforms FM when only deletions are required.

7.6 . Parallel Corpus Cleaning

The objective of our models is to restore bilingual parallelism between a pair of sentences. This ability is also helpful for parallel corpus cleaning tasks, which we define as follows. Given a source sentence $\mathbf{f}$ and a possibly incorrect translation $\tilde{\mathbf{e}}$, we would like to detect cases of non-parallelism and perform appropriate edits to obtain a parallel sentence pair. We study how Edit-MT deals with this new problem on two publicly available datasets: first on the SemEval 2012&3 Task 8: Cross-lingual Textual Entailment (CLTE) for Content Synchronization (Negri et al., 2011, 2012, 2013), then with the OpenSubtitles corpus (Lison and Tiedemann, 2016).

7.6.1 . Cross-Lingual Text Entailment

The CLTE task aims to identify multi-directional entailment relationships between two sentences $\mathbf{x}_1$ and $\mathbf{x}_2$, written in different languages. Each $(\mathbf{x}_1, \mathbf{x}_2)$ pair in the dataset is annotated with one of the following relation-

ships:

- *Bidirectional* ($\mathbf{x}_1 \Leftrightarrow \mathbf{x}_2$): the two fragments entail each other (semantic equivalence).
- *Forward* ($\mathbf{x}_1 \Rightarrow \mathbf{x}_2 \;\&\; \mathbf{x}_1 \not\Leftarrow \mathbf{x}_2$): unidirectional entailment from $\mathbf{x}_1$ to $\mathbf{x}_2$.
- *Backward* ($\mathbf{x}_1 \not\Rightarrow \mathbf{x}_2 \;\&\; \mathbf{x}_1 \Leftarrow \mathbf{x}_2$): unidirectional entailment from $\mathbf{x}_2$ to $\mathbf{x}_1$.
- *No Entailment* ($\mathbf{x}_1 \not\Leftrightarrow \mathbf{x}_2$): no entailment between $\mathbf{x}_1$ and $\mathbf{x}_2$.

The dataset contains a training set of 500 sentence pairs and two test sets of the same size (test-2012 and test-2013). Our evaluation is for the Fr-En direction, where $\mathbf{x}_1$ is in French and $\mathbf{x}_2$ is in English.

The tagging mechanism of Edit-MT introduced in Section 7.3 can readily be used for this classification task, subject to some adjustments as represented in Table 7.13. We then treat $\mathbf{x}_1$ as $\mathbf{f}$ and $\mathbf{x}_2$ as $\tilde{\mathbf{e}}$ to match the input format of Edit-MT and perform zero-shot inference reusing the same Edit-MT models trained in Chapter 7. The source side is the concatenation of $\mathbf{x}_1$ and $\mathbf{x}_2$, while the target sequence is truncated by only taking the first three edit tags as the predicted label for the corresponding input pair, treating our Edit-MT as a mere classification model as mentioned in Section 7.3. We also slightly fine-tune Edit-MT with the 500 examples of CLTE training data for 5 epochs with a fixed learning rate of $8\mathrm{e}{-5}$.

CLTE Fr-En	Edit-MT En-Fr	Edit-MT Fr-En
Bidirectional	Copy	Copy
Forward	Deletion	Insertion
Backward	Insertion	Deletion
No Entailment	Substitution	Substitution

Table 7.13: Label conversion scheme between CLTE task and Edit-MT editing tags.

Results are in Table 7.14, together with the best scores reported in (Negri et al., 2013) for both test sets and the scores reported in (Carpuat et al., 2017), which are the best performance reported we could find. As can be seen, out-of-the-box Edit-MT fails to clearly detect the entailment relationships. This is not surprising, as there is a significant difference between our editing training data and the CLTE test sets. For instance, the initial translation for insertion is always grammatically incorrect, while all sentences in CLTE are syntactically correct. Nevertheless, after slight fine-tuning with the CLTE data, Edit-MT for both directions can quickly adapt to the task, achieving state-of-the-art performance.[15] This hints at the fact that Edit-

[15]We have not performed hyperparameter searching for fine-tuning, even though carefully fine-tuned models may achieve even better performance. We also acknowl-

MT actually learns to identify various cases of non-parallelism and generate appropriate edits.

Methods	2012	2013
Best SemEval13	0.570	0.458
Carpuat et al. (2017)	0.604	0.436
Edit-MT En-Fr	0.350	0.284
+ FT	**0.716**	0.466
Edit-MT Fr-En	0.376	0.288
+ FT	0.710	**0.530**

Table 7.14: Accuracy scores on the SemEval CLTE tasks. FT denotes Edit-MT models fine-tuned for the CLTE classification tasks.

7.6.2 . Fixing OpenSubtitles Corpus

We further evaluate the ability of Edit-MT to detect parallel sentences and to fix noisy data. We experiment with the OpenSubtitles[16] data (Lison and Tiedemann, 2016) for the En-Fr direction. The French side is translated from English in this corpus but contains noisy segments. A standard approach is to filter out these noisy sentences from the training material when building systems. Our goal in this experiment is to see whether Edit-MT can automatically identify and edit, rather than discard, noisy sentence pairs so that training can use the entire set of available parallel data. We evaluate the performance on the $10{,}159$ segments of the En-Fr Microsoft Spoken Language Translation (MSLT) task (Federmann and Lewis, 2016), which offers an MT scenario motivated by real-world applications.

The OpenSubtitles data is first deduplicated and then processed similarly to Section 7.4.1. We first apply the fine-tuned classification model used for the CLTE task to predict the relation for all sentence pairs in OpenSubtitles data. Approximately 60% of the data is classified as parallel, indicating that no edit operation was predicted for those segments. Models trained on these 60% clean data are denoted below as `filtered`. For the other 40% of presumably noisy data, we reuse the generic Edit-MT En-Fr model trained in Section 7.4 to edit the translations, using the predicted edit tag as a prefix on the target side (*cf.*, Section 7.3). Models trained using the edited data are denoted as `fixed`. We train NMT models with all data (`full`) or just the 40% `noisy` data as baselines. For comparison, we also train a model containing the same number of sentences (15.8M) as the `noisy` subset, randomly selected from the `filtered` subset.

edge the fact that the baselines for this task may be relatively weak, as large pre-trained language models did not exist at that time.

[16] https://opus.nlpl.eu/OpenSubtitles-v2018.php

Results are in Table 7.15. Aggressively filtering the noisy data improves over using the entire training corpus ($+2$ BLEU) more than revising it ($+1$ BLEU). The second set of results yields similar conclusions with smaller datasets: here, the effect of automatically fixing a set of initially noisy data improves the BLEU score by 7.2 points and closes half the gap with a completely clean corpus of the same size. Note that these results were obtained without adaptation, simply reusing the pre-trained generic Edit-MT model. This suggests that editing-based strategies may provide an effective alternative to filtering in situations where the training data is small and noisy.

Cleaning Method	BLEU	Corpus size
`full`	44.7	41.6M
`filtered`	**46.7**	25.8M
`filtered` + `fixed`	45.7	41.6M
`noisy`	32.2	15.8M
`fixed`	39.4	15.8M
`filtered` (15.8M)	**46.7**	15.8M

Table 7.15: BLEU scores on MSLT task of models trained with different subsets of OpenSubtitles.

7.7 . Conclusion

In this chapter, we introduced another possible solution for bilingual writing, which simulates a revision scenario where updates in one language need to be propagated to another language. We proposed a new Bi-sync task that corresponds to this situation. We also proposed various ways to create artificial initial translations for different editing types needed to train Bi-sync models. We have explored autoregressive architectures for this task. Experiments showed that our Edit-MT models trained with a mixture of real parallel data and artificial triplets were able to perform Bi-sync tasks even in real-world scenarios while also maintaining the ability to perform standard translation. We have also shown that a multilingual edit-based model could perform Bi-sync in both directions within a single model. This led to more realistic application scenarios like bilingual writing, where we can treat both languages as completely equivalent.

We also explored Bi-sync on two specific downstream tasks. Our experiments demonstrated that Edit-MT could be applied to fix TMs by detecting parallel sentences with a slight amount of fine-tuning and correcting imperfect translations without any adaptations. We have also shown that Edit-MT could be quickly adapted to multi-domain MT with TMs tasks, where it could outperform dedicated models when tested on unseen domains.

In our future work, we would like to study ways to reduce the computational cost of fully re-decoding the input sequence for Edit-MT, especially in contexts where small changes that need to be reproduced in the target are repeatedly applied to the source sequence. We also want to experiment with actual bilingual writing interactive scenarios where such techniques apply.

8 - Non-autoregressive Bilingual Synchronization

8.1 . Introduction

In Chapter 7, we studied bilingual synchronization (Bi-sync) for autoregressive (AR) architectures. Bi-sync models are expected to edit an initial target sequence to restore parallelism between the source and target languages. However, AR neural machine translation (NMT) systems do not really edit an initial target sequence. Due to the AR generation, these systems have to generate a complete target sentence from scratch in order to make changes to an initial target sequence present on the source side. Even though only a few edits are required, AR models still need to regenerate the entire target sequence through a standard decoding process. In an interactive bilingual writing scenario, always recomputing the entire target sequence is time-consuming, especially for the AR generation.

On the contrary, some non-autoregressive machine translation (NAT) models have been proposed to perform iterative refinement decoding (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019), where the model generates translations by iteratively editing the outputs of previous iterations, starting with a possibly empty initial hypothesis, as we introduced in Section 4.1.3. Among these NAT approaches, edit-based models like the Levenshtein Transformer (LevT, Gu et al., 2019) seem to be a natural fit for the Bi-sync task. It has recently been applied to incorporate lexical constraints in machine translation (MT) (Susanto et al., 2020; Xu and Carpuat, 2021a), as discussed in Section 4.5.

Therefore, in this chapter, we study Bi-sync with non-autoregressive approaches. We show that the vanilla LevT cannot perform well on this task and explain why this is the case. We propose Edit-LevT, a new NAT model based on the LevT with an improved training procedure to incorporate empty and non-empty initial target sentences. Edit-LevT is able to perform Bi-sync much better than LevT. We also apply Edit-LevT to the MT with translation memories (TM) task, as also considered in Section 7.5 for AR Edit-MT.

We pay special attention to the NAT with TMs task and study it more in-depth, as it is a realistic application of Bi-sync but has never been studied to the best of our knowledge. We propose TM-LevT for this task, which is adapted from Edit-LevT with a different source input format. We also improve the training procedure in two ways: (a) by also including the initial candidate translation on the source side, as done in the AR decoding with TMs (Bulte and Tezcan, 2019); (b) by simultaneously training with empty

and non-empty initial target sentences, which is also applied to Edit-LevT. In our experiments, TM-LevT achieves comparable performance to the AR approach on various domains when translating with and without TMs. It also yields a faster decoding speed with a reduced decoding load. We also observe that incorporating an initial translation on both source and target sides makes the widely used sequence-level knowledge distillation (KD, Kim and Rush, 2016) useless, even when performing translation from scratch. This contrasts with standard NAT models, which resort to KD to alleviate the multimodality issue (Gu et al., 2018a), as discussed in Section 4.4. As far as we know, we are the first to perform NAT with TMs and match AR model performance in this setting.

Our main contributions in this chapter are the following:

- We propose a new NAT model based on the LevT that is able to perform Bi-sync and achieves similar results to AR Edit-MT models from Chapter 7.
- We perform empirical validations of the non-autoregressive Bi-sync model on the MT with TMs task to validate the generalization ability of our non-autoregressive Bi-sync model.
- We improve our proposed NAT model to perform the NAT with TMs task and achieve comparable results to a strong AR approach.

Some of the contributions in this chapter are published in (Xu et al., 2022b,c).

8.2 . Edit-LevT

Even though the edit-based nature of LevT seems to suit Bi-sync, it is still designed to perform standard MT. The original LevT model initializes the decoder with a randomly noised version of the target reference during training and starts with an empty sentence in inference (*cf.*, Section 4.1.3). This means that the input to the LevT decoder $\mathbf{e}'$ is always a subsequence of $\mathbf{e}$, and the deletion operation in LevT is only trained to detect prediction errors made by the model itself. In Bi-sync, we would instead initialize the target side with the given initial translation $\tilde{\mathbf{e}}$ in both training and inference so that the model can directly edit the given target sentence. However, the original training scheme does not suit Bi-sync, as the initial translation $\tilde{\mathbf{e}}$ may contain tokens that are not present in the reference $\mathbf{e}$ but correspond to a previous source $\tilde{\mathbf{f}}$ and should be removed. Figure 8.1 shows an example of an initial translation $\tilde{\mathbf{e}}$ containing the word "asleep" that should be replaced. The distribution of these tokens may greatly differ from token prediction errors made by LevT, as they are not really "errors" but just irrelevant to the actual reference $\mathbf{e}$. To make this point, we have experimented with original LevT models by simply replacing the noised decoder input $\mathbf{e}'$ with the initial

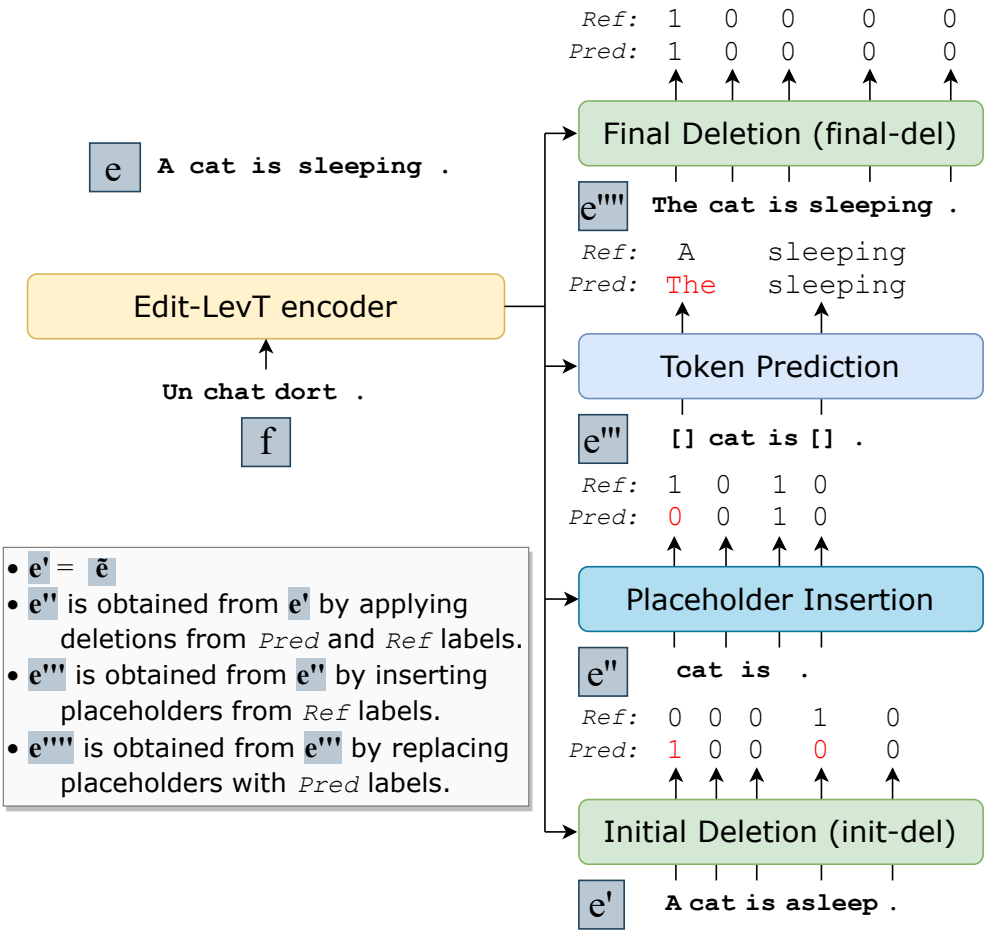


Figure 8.1: A complete training step for Edit-LevT.

translations $\tilde{\mathbf{e}}$ generated in Section 7.2 for training. However, this model cannot make proper edits and almost considers $\tilde{\mathbf{e}}$ as the final translation. This is further discussed in Section 8.3.2.

Therefore, we propose Edit-LevT, a new NAT model aiming to fix this issue and adopt the LevT model for Bi-sync. To do so, we modify the training regime and add an extra deletion operation (init-del) before the insertion operation to the original LevT. As illustrated in Figure 8.1, init-del is trained to detect irrelevant tokens from the initial sequence $\tilde{\mathbf{e}}$, while the final deletion (final-del) aims to delete prediction errors made by the model, therefore making Edit-LevT able to detect both irrelevant tokens and its own errors. We apply the initial deletions to $\mathbf{e}'$ during training by taking the union of reference and predicted deletions in the init-del operation, resulting in a subsequence $\mathbf{e}''$ which is then passed to the insertion operation. We use the same deletion classifier for both the init-del and the final-del operations, which means that Edit-LevT does not contain any extra parameters com-

pared to the original LevT. During inference, Edit-LevT performs as LevT, iteratively applying deletions and insertions to an initial candidate translation until convergence or a maximum decoding round is reached.

To adapt Edit-LevT to Bi-sync, for each triplet of samples $(\mathbf{f}, \tilde{\mathbf{e}}, \mathbf{e})$, we use the initial translation $\tilde{\mathbf{e}}$ to initialize the decoding ($\mathbf{e}' = \tilde{\mathbf{e}}$). Different from Edit-MT, the source side of Edit-LevT only contains the source sentence $\mathbf{f}$. Finally, in order to preserve the ability to perform standard MT, our training data is prepared as follows: with a tunable probability p, we decide either to train with an initial translation $\tilde{\mathbf{e}}$ or to train from scratch. In the former case, the decoder is initialized with $\tilde{\mathbf{e}}$, while in the latter case, we use a noised subsequence $\mathbf{e}'$ generated as in (Gu et al., 2019). Edit-LevT is then jointly trained on both tasks. The probability p controls the proportion of each sample type. Taking $p = 0$ is equivalent to training an Edit-LevT model with only parallel data. We use $p = 0.5$ in our experiments, making it equivalent to mixing parallel and editing data as in the Edit-MT model (*cf.*, Section 7.3). The probability p can be carefully designed with a schedule or a curriculum during training to optimize the behavior of Edit-LevT, which we leave for future work. Note that for Edit-LevT, we do not use any tags on the target side, as Edit-LevT already includes an internal mechanism to predict the editing operation(s). During inference, we initialize the target sequence with initial translations $\tilde{\mathbf{e}}$, using a setting akin to that of (Susanto et al., 2020) for LevT with lexically constrained decoding, which does not involve any changes in the iterative refinement decoding algorithm.

8.3 . Bilingual Synchronization

8.3.1 . Datasets and Experimental Settings

We reuse the same training and test data for En-Fr and Fr-En in Section 7.4.1 in this experiment. Edit-LevT is implemented using the `fairseq`[1] toolkit (Ott et al., 2019). We use a hidden size of 512 and a feed-forward size of $2{,}048$. We follow Gu et al. (2019) to optimize using Adam with a maximum learning rate of 0.0005, an inverse square root decay schedule, and $10{,}000$ warm-up steps. We also share all embedding matrices and use a shared decoder for the placeholder insertion, token prediction, and the two deletion operations. Each operation thus only differs in the linear classifier. However, the init-del and final-del operations also share the same deletion classifier. Edit-LevT models are trained with mixed precision and a batch size of $16{,}384$ tokens on 4 V100 GPUs for 300k iterations. We save checkpoints for every $3{,}000$ iterations and average the last 10 saved checkpoints for inference. For inference, we set a maximum decoding round of 10. Sequence-level

[1] https://github.com/pytorch/fairseq

KD is widely used in NAT model training (Gu et al., 2018a). However, we use real target references instead of distilled targets from an AR model for training, as our editing data $\tilde{\mathbf{e}}$ are generated based on real target sentences $\mathbf{e}$, and also to be fully comparable with Edit-MT in Chapter 7 trained on the same data.

We also train a vanilla LevT model which does not contain the init-del module. Similar to Edit-LevT, the vanilla LevT model also takes $\tilde{\mathbf{e}}$ as initialization for the decoder with a probability of $p = 0.5$ during training. Therefore, the only difference between LevT and Edit-LevT is the init-del operation. Table 8.13 summarizes the architecture and data format of different models used in this chapter. We report the "do-nothing" copy baseline and compare Edit-LevT to the vanilla LevT and the AR Edit-MT model under the same test settings in Section 7.4.3. Performance is computed with SacreBLEU (Post, 2018).

8.3.2 . Main Results

We first experiment with LevT and Edit-LevT performing standard translation for both directions. In this setting, the decoder of both NAT models is initialized with empty. Results are shown in Table 8.1. For NAT models, the overall performance lags behind AR approaches. This is not surprising, especially when trained without KD. However, our Edit-LevT can outperform the vanilla LevT model trained with the same data. This implies that training with init-del has positive effects for standard translation. In the following of this section, we focus on performing the Bi-sync task with Edit-LevT.

Model	En-Fr	Fr-En
`base-mono`	37.6	35.2
Edit-MT	35.7	34.7
LevT	28.0	28.1
Edit-LevT	29.2	28.2

Table 8.1: BLEU scores of Edit-LevT performing standard translation on `newtest2014` for both En-Fr and Fr-En. The `base-mono` results are copied from Section 5.5.

Results of the Bi-sync task for both directions are shown in Tables 8.2 and 8.3. The original LevT model can slightly improve $\tilde{\mathbf{e}}$ in most test situations. However, the performance gains are much lower, and it does not make many differences compared to the copy baseline except for the insertion task. The main reason behind this phenomenon is that, in the design of LevT training, even if we initialize the target side with $\tilde{\mathbf{e}}_{\text{sub}}$ or $\tilde{\mathbf{e}}_{\text{del}}$, which are not a subsequence of the reference $\mathbf{e}$ with certain tokens to be deleted, the training procedure always starts with the insertion operation to evaluate if it should

Model	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	54.0	71.5	71.0	78.7	68.4	66.7
Edit-MT	75.9	77.0	86.9	94.7	73.1	67.9
LevT	65.3	73.9	72.5	78.7	67.8	67.7
Edit-LevT	**72.6**	**76.3**	**81.9**	**92.2**	**71.9**	**68.4**

Table 8.2: BLEU scores of Edit-LevT on Bi-sync for En-Fr. Deletions are evaluated separately for the two generation methods (Del_1 and Del_2). The best performance of NAT methods is in bold.

Model	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	51.8	70.9	71.0	78.7	63.4	61.4
Edit-MT	73.6	74.6	87.5	95.8	65.8	69.3
LevT	66.5	72.4	72.3	78.4	62.5	64.3
Edit-LevT	**70.7**	**74.1**	**82.8**	**92.7**	**63.8**	**65.9**

Table 8.3: BLEU scores of Edit-LevT on Bi-sync for Fr-En.

insert placeholders in $\tilde{\mathbf{e}}_{\text{sub}}$ or $\tilde{\mathbf{e}}_{\text{del}}$. Therefore, tokens to be substituted or deleted in $\tilde{\mathbf{e}}_{\text{sub}}$ or $\tilde{\mathbf{e}}_{\text{del}}$ are never seen by the deletion operation, which makes the LevT model almost perform the standard NAT task rather than make use of initial translations $\tilde{\mathbf{e}}$.

On the contrary, our Edit-LevT model obtains more significant gains compared to LevT in all situations. Since the only difference between Edit-LevT and LevT is the init-del operation, the results demonstrate that adding init-del effectively helps Edit-LevT learn to detect and delete tokens from an initial translation. Nevertheless, there is still a clear performance gap between Edit-LevT and Edit-MT (without oracle tags). This is not surprising as NAT models generally suffer from a performance drop compared to autoregressive approaches due to the independent assumption of target tokens, especially in our case where Edit-LevT is trained without using the distilled data. However, the performance gap is narrowed compared to the gap when performing standard translation in Table 8.1. Edit-LevT can even outperform Edit-MT without oracle tags on the extension task for En-Fr. Note that we do not compare Edit-LevT to Edit-MT with oracle tags, as the latter incorporates extra oracle information that Edit-LevT does not possess.

8.3.3 . Multilingual Edit-LevT

Similar to Section 7.4.4, we also explore Edit-LevT under the multilingual setting. Since the source side of Edit-LevT only contains the source sentence $\mathbf{f}$, we do not use language tags to distinguish the target language and simply concatenate data from both directions for training. We also double the batch size when training the multilingual Edit-LevT.

En-Fr	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	54.0	71.5	71.0	78.7	68.4	66.7
Edit-MT (En-Fr)	**75.9**	77.0	**86.9**	**94.7**	**73.1**	67.9
`multi` Edit-MT	75.5	**77.2**	**86.9**	**94.7**	72.3	**68.4**
Edit-LevT (En-Fr)	**72.6**	**76.3**	81.9	92.2	71.9	68.4
`multi` Edit-LevT	72.4	**76.3**	**83.0**	**92.4**	**72.5**	**68.6**
Fr-En	Ins	Sub	Del_1	Del_2	Comp	Ext
copy	51.8	70.9	71.0	78.7	63.4	61.4
Edit-MT (Fr-En)	**73.6**	74.6	**87.5**	**95.8**	**65.8**	**69.3**
`multi` Edit-MT	73.5	**75.1**	86.7	**95.8**	64.6	68.0
Edit-LevT (Fr-En)	**70.7**	**74.1**	82.8	**92.7**	63.8	65.9
`multi` Edit-LevT	70.5	74.0	**83.1**	92.4	**64.5**	**66.3**

Table 8.4: BLEU scores of multilingual Edit-LevT on Bi-sync for both En-Fr and Fr-En. The multilingual model denoted with `multi` is the same model used for both directions. The best performance in each block is in bold.

As shown in Table 8.4, the multilingual Edit-LevT performs similarly to its monolingual counterparts on the synthetic test sets (Ins, Sub, Del_1, Del_2) in both directions. On the more realistic compression (comp) and extension (ext) test sets, multilingual Edit-LevT even outperforms monolingual models in both directions for both tasks. This implies that for Edit-LevT, training with data from both directions can even help to boost the performance to some extent. Note again that, similar to the previous section, we do not compare Edit-MT models with oracle tags.

8.3.4 . Simplifying Editing Data for Edit-LevT

The editing training data generated in Section 7.2 is composed of $\tilde{\mathbf{e}}_{\text{ins}}$, $\tilde{\mathbf{e}}_{\text{sub}}$, $\tilde{\mathbf{e}}_{\text{del}}$, and $\tilde{\mathbf{e}}_{\text{copy}}$. However, the original training scheme of the LevT model uses a subsequence of the target reference as the decoder input $\mathbf{e}'$. This subsequence is, in fact, similar to our insertion data $\tilde{\mathbf{e}}_{\text{ins}}$, as both are generated by randomly dropping tokens from the reference $\mathbf{e}$. However, the subsequence $\mathbf{e}'$ is generated without specific constraints, and the percentage of dropped tokens is uniformly distributed between 0 and 1. Since we include dropping constraints in Section 7.2.1, the dropped token ratio for $\tilde{\mathbf{e}}_{\text{ins}}$ is thus concentrated in a certain range. Therefore, the insertion data may be simplified since we apply a probability $p = 0.5$ to use the $\mathbf{e}'$ when training our Edit-LevT model.

On the other hand, the ability to keep already parallel sentences unchanged in Edit-LevT is also preserved by the iterative refinement decoding, which will reach a convergence when it detects the initial input as a proper translation. Therefore, we would like to study if we can simplify the editing data required to train Edit-LevT models. To do so, we prepare a new editing

En-Fr	Ins	Sub	Del_1	Del_2	Comp	Ext
Edit-LevT	72.6	76.3	81.9	92.2	71.9	68.4
Edit-LevT sub+del	**72.8**	**76.6**	**84.8**	**93.3**	**73.8**	**68.8**
Fr-En	Ins	Sub	Del_1	Del_2	Comp	Ext
Edit-LevT	**70.7**	74.1	82.8	92.7	63.8	**65.9**
Edit-LevT sub+del	70.5	**74.3**	**85.5**	**93.9**	**64.2**	65.8

Table 8.5: BLEU scores of Edit-LevT trained with different editing data on Bi-sync for both En-Fr and Fr-En. Models trained with only substitution $\tilde{\mathbf{e}}_{\text{sub}}$ and deletion $\tilde{\mathbf{e}}_{\text{del}}$ editing data are denoted with *sub+del*.

training data in which the initial translation $\tilde{\mathbf{e}}$ is composed only of $\tilde{\mathbf{e}}_{\text{sub}}$ and $\tilde{\mathbf{e}}_{\text{del}}$ data. Simplifying the training data also allows Edit-LevT to see more substitution and deletion data during training, potentially improving the performance of these two editing types. In this section, we only experiment with monolingual Edit-LevT models.

Results reported in Table 8.5 suggest that training Edit-LevT with only $\tilde{\mathbf{e}}_{\text{sub}}$ and $\tilde{\mathbf{e}}_{\text{del}}$ editing data does not hurt the performance in general and even brings gains over almost all situations for both directions. The insertion and extension tasks for Fr-En are not improved, but the performance remains almost the same. The improvement mostly comes from the compression task and deletion tasks. As expected, seeing more data helps to improve the ability to detect segments in the initial translations that should be removed. However, the performance gain for the substitution task is much smaller.

8.4 . Translating with Translation Memories

Similar to Section 7.5, we also adapt our Edit-LevT model to the MT with TMs task in this section to evaluate the ability of Edit-LevT to generalize to this practical scenario.

8.4.1 . Datasets and Experimental Settings

We reuse the same training and test datasets in Section 7.5.2 and consider the same `FM` and $\texttt{FM}^{\#}$ baselines as in Section 7.5.3. As Edit-LevT also differs from the two baselines in both the task and the domains, we also consider fine-tuning the general Edit-LevT model. We use *para* + *similar* + *related* data to fine-tune the generic Edit-LevT model. Unlike the AR models, the similar translation or segments $\tilde{\mathbf{e}}$ are not concatenated to the source for Edit-LevT. On the contrary, they are used to initialize the decoder during training (see Table 8.13). The initial translation $\tilde{\mathbf{e}}$ for *para* data is set to empty. We follow the settings in Section 8.2 to use a probability of $p = 0.5$ to select initial translations $\tilde{\mathbf{e}}$ or randomly noised reference subsequences $\mathbf{e}'$.

We use the pre-trained Edit-LevT En-Fr `sub+del` model, as it yields better general results, and fine-tune it for 2 epochs with a learning rate of $9e^{-5}$ to keep approximately the same number of updates as Edit-MT since Edit-LevT uses a bigger batch size. The fine-tuned Edit-LevT model can perform translations with both similar sentences and related segments as the initial state within the same model. We evaluate both BLEU and TER scores using SacreBLEU.

8.4.2 . Results

We first perform standard translation on the multi-domain test sets without using the retrieved similar translations. As shown in Table 8.6, the generic Edit-LevT model performs worse than the generic AR Edit-MT model. This is not surprising, as AR models are always better than NAT models. By performing fine-tuning, Edit-LevT can greatly improve its performance. Even though it always cannot match the performance of Edit-MT, the gap is similar to that in Section 8.3.2, showing that Edit-LevT adapts to the new domains in a similar trend as Edit-MT, while its general ability lags behind Edit-MT. In the following of this section, we focus on translating with TMs.

BLEU ↑	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	59.8	64.5	34.4	70.3	67.6	55.3	12.0	38.6	30.8	51.6	47.4	52.6
FM	59.6	54.8	55.2	55.8	69.3	54.6	26.9	39.8	61.6	56.9	65.9	56.7
FM$^{\#}$	61.2	56.8	55.8	57.5	70.3	55.8	27.9	40.2	61.3	58.9	54.9	57.7
Edit-MT	44.6	32.8	51.9	37.2	54.9	28.1	25.2	33.8	47.8	36.5	34.5	41.8
+ FT	53.3	48.9	53.7	50.0	64.3	47.3	26.6	36.3	59.2	49.6	59.3	51.8
Edit-LevT	35.7	27.3	46.7	30.2	45.5	22.6	21.0	27.5	44.0	35.3	32.6	35.3
+ FT	47.1	41.7	50.0	46.5	58.9	36.0	21.9	31.6	55.8	46.4	56.5	46.5

Table 8.6: BLEU scores of Edit-LevT by performing standard translation **without** using TMs on test sets from multiple domains. *All* is computed by concatenating test sets from all domains, with 11k sentences in total. *Copy* refers to copying the retrieved similar translations to the output. *+ FT* refers to models fine-tuned on the multi-domain TM data.

Results in Tables 8.7 and 8.8 indicates that the generic Edit-LevT performs similarly to the generic Edit-MT, but fine-tuning drastically improves the performance ($+10.1$ BLEU, -0.11 TER points on average). Nevertheless, the quality of fine-tuned Edit-LevT still lags behind Edit-MT. The *related* setting also benefits from fine-tuning, albeit by a smaller margin ($+6.4$ BLEU). The performance gap between Edit-LevT and Edit-MT on average is similar to that in Table 8.1, indicating that Edit-LevT also adapts to this task, with a similar trend as Edit-MT. Note that we compare the best AR Edit-MT configuration with prefixed tags, as the tags are not oracle and the effect of tags is less significant for Edit-MT in this setting.

BLEU ↑	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	59.8	64.5	34.4	70.3	67.6	55.3	12.0	38.6	30.8	51.6	47.4	52.6
FM	**72.1**	**72.3**	**58.3**	**80.6**	**83.2**	**66.9**	28.0	**47.2**	**62.9**	**69.3**	68.8	**67.3**
FM$^{\#}$	69.3	68.1	58.2	74.2	80.1	65.2	**28.6**	44.3	62.6	68.1	**69.0**	65.0
Edit-MT+t	60.3	63.0	35.7	70.3	68.4	51.9	12.9	38.8	32.6	52.1	45.6	52.6
+ FT + t	**70.6**	**71.5**	**57.8**	**78.2**	**82.0**	**66.2**	**28.0**	**45.1**	**61.1**	**67.7**	**66.8**	**66.0**
Edit-LevT	59.5	62.0	34.9	69.8	67.9	46.5	12.6	36.1	31.5	51.1	44.6	51.4
+ R	55.3	52.5	43.4	63.7	67.3	46.4	16.3	36.0	41.6	55.2	52.5	51.2
+ FT	**64.8**	**69.1**	**51.2**	**75.6**	**78.7**	**61.5**	20.5	**39.8**	**54.9**	**62.8**	**63.6**	**61.5**
+ FT + R	59.9	59.7	50.1	68.0	73.8	55.3	**20.6**	37.7	52.7	60.5	62.6	57.6

Table 8.7: BLEU scores of Edit-LevT on test sets from multiple domains when translating **with** TMs. *+t* refers to AR models decoding with prefixed tags. + *R* implies using the related segments instead of a full initial sentence for inference. + *FT* refers to models fine-tuned on the multi-domain TM data. The best performance in each block is in bold.

TER ↓	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	.435	.377	.659	.263	.294	.457	.999	.703	.653	.372	.502	.488
FM	**.286**	**.301**	.374	**.160**	**.138**	.312	.642	**.572**	.328	**.227**	**.306**	**.314**
FM$^{\#}$	.299	.332	**.365**	.202	.152	**.309**	**.634**	.605	**.327**	.234	.309	.327
Edit-MT+t	.412	.382	.638	.256	.283	.452	.899	.643	.635	.360	.520	.468
+ FT + t	**.300**	**.308**	**.376**	**.182**	**.146**	**.314**	**.643**	**.595**	**.351**	**.237**	**.329**	**.326**
Edit-LevT	.411	.395	.637	.261	.286	.474	.873	.649	.639	.385	.526	.472
+ R	.411	.436	.516	.273	.246	.460	.791	.629	.540	.317	.453	.438
+ FT	**.341**	**.321**	**.432**	**.204**	**.174**	**.379**	.739	**.603**	**.404**	**.277**	**.350**	**.362**
+ FT + R	.382	.408	.435	.262	.197	.478	**.738**	.629	.497	.335	**.350**	.402

Table 8.8: TER scores of Edit-LevT on test sets from multiple domains when translating **with** TMs.

Model	Office	ENV
copy	54.7	59.6
FM	**66.8**	**75.4**
FM$^{\#}$	64.0	70.6
Edit-MT + tag	56.2	60.3
+ FT + tag	**68.6**	**78.6**
Edit-LevT	54.4	59.8
+ FT	**62.2**	**75.1**

Table 8.9: BLEU scores of Edit-LevT on unseen domains when translating with TMs. + *tag* refers to decoding with the oracle tag as a forced prefix.

We also evaluate Edit-LevT on the two unseen domains (Office and ENV). Results in Table 8.9 show that in this setting, the fine-tuned Edit-LevT is also able to make good use of TMs to close the performance gap with both FM and FM$^{\#}$. It even surpasses the FM$^{\#}$ model and achieves performance on par with FM in the ENV domain. This suggests that Edit-LevT has also adapted to the domain and the task, as it can effectively perform zero-shot

TM-based translation on unseen domains.

8.4.3 . Analyses

To better understand Edit-LevT on TM-based translation, we further aggregate the 11 test sentences and break them down by the difference Δ between $\tilde{\mathbf{e}}$ and $\mathbf{e}$ (computed as edit distance) in Table 8.10, as in Section 7.5.5. The generic Edit-LevT model performs worse than the copy baseline in all cases. After fine-tuning, the performance gap between Edit-LevT and Edit-MT is smaller when minor edits are needed ($\Delta < 3$). Fine-tuned Edit-LevT is better than FM and fine-tuned Edit-MT models at detecting the initial translations that do not need any edits ($\Delta = 0$). As mentioned in Section 8.1, AR models need to regenerate the entire target sequence through a standard decoding process, even though they only need to copy the initial $\tilde{\mathbf{e}}$. Therefore, we cannot guarantee that they will always copy the initial $\tilde{\mathbf{e}}$ exactly. On the contrary, Edit-LevT leaves the initial $\tilde{\mathbf{e}}$ unchanged if it detects that $\tilde{\mathbf{e}}$ is already parallel to the source.

B↑ $\Delta(\tilde{\mathbf{e}}, \mathbf{e})$	0	1	2	3	4	5	6	7	8-10	>10
N	540	2096	1107	882	827	782	689	607	1193	2277
copy	100.0	82.3	74.2	67.2	62.1	51.7	50.5	48.2	40.7	33.8
FM	91.6	**93.3**	**86.5**	**82.3**	**79.2**	**70.5**	**69.0**	**68.0**	**60.9**	**49.5**
Edit-MT+tag	95.3	80.8	72.9	68.0	62.7	52.4	50.7	49.6	41.5	34.3
+ FT + tag	91.6	91.1	85.8	80.9	77.7	68.8	68.4	66.6	59.0	48.0
Edit-LevT	**95.5**	78.9	71.4	66.3	60.5	50.5	49.0	48.2	40.4	33.4
+ FT	94.1	89.2	83.0	77.1	74.4	64.7	63.6	62.3	53.0	41.4
AD↓ $\Delta(\tilde{\mathbf{e}}, \mathbf{e})$	0	1	2	3	4	5	6	7	8-10	>10
copy	0.000	1.000	2.000	3.000	4.000	5.000	6.000	7.000	8.854	23.197
FM	1.600	**0.561**	**1.321**	**1.827**	**2.242**	**2.978**	**3.598**	**4.104**	**5.285**	**15.468**
Edit-MT+tag	0.850	1.210	2.266	3.008	3.959	4.937	5.888	6.741	8.578	21.409
+ FT + tag	1.493	0.770	1.397	1.954	2.435	3.265	3.671	4.254	5.513	15.792
Edit-LevT	**0.744**	1.379	2.402	3.147	4.123	5.059	5.993	6.807	8.638	21.130
+ FT	0.872	0.748	1.537	2.227	2.712	3.584	4.290	4.771	6.418	17.419

Table 8.10: BLEU (B) scores and average edit distance (AD) of Edit-LevT broken down by the distance Δ between $\tilde{\mathbf{e}}$ and $\mathbf{e}$. Each column represents a range of distances. N denotes the number of sentences in each group. Best performance of each column is in bold.

We also break down the aggregate test set by editing operations required to transform $\tilde{\mathbf{e}}$ to $\mathbf{e}$. Table 8.11 shows that the generic Edit-LevT performs worse than the copy baseline when substitutions are needed. A substitution in Edit-LevT requires first deleting a token and then generating another token. Once a token is deleted, information from this token is lost, and Edit-LevT has to condition on other existing tokens to predict a new token. Fine-tuning vastly improves the ability to substitute and delete from $\tilde{\mathbf{e}}$, separately or even in combination.

BLEU↑	=	I	S	D	I+S	I+D	S+D	I+S+D	All
N	540	316	3260	316	2865	58	2634	1011	11000
copy	100.0	72.0	67.9	75.4	32.5	69.8	34.0	47.3	52.6
FM	91.6	**80.6**	**86.6**	82.9	**50.0**	67.4	**58.4**	**63.0**	**67.3**
Edit-MT+tag	95.3	75.7	67.0	77.2	34.2	68.5	37.4	48.0	52.6
+ FT + tag	91.6	79.7	84.6	**85.8**	48.3	**69.9**	57.6	60.8	66.0
Edit-LevT	**95.5**	76.0	65.0	77.3	33.0	68.9	37.4	47.4	51.4
+ FT	94.1	77.5	81.1	81.4	41.8	67.7	52.0	56.7	61.5
AD↓	=	I	S	D	I+S	I+D	S+D	I+S+D	All
copy	0.000	3.791	2.336	2.744	10.696	8.621	10.555	17.125	7.813
FM	1.600	**3.038**	**1.275**	2.016	**8.002**	7.172	**5.436**	**11.389**	**5.072**
Edit-MT+tag	0.850	3.725	2.528	2.535	10.539	8.690	9.152	16.756	7.493
+ FT + tag	1.493	3.225	1.501	**1.557**	8.287	**6.621**	5.572	11.763	5.264
Edit-LevT	**0.744**	3.509	2.724	2.484	10.692	8.310	8.913	16.818	7.525
+ FT	0.872	3.244	1.639	2.013	9.111	7.414	6.413	12.882	5.811

Table 8.11: BLEU scores and averaged edit distance (AD) of Edit-LevT broken down by the editing operations required between $\tilde{\mathbf{e}}$ and $\mathbf{e}$. Each column represents a combination of edits. N denotes the number of sentences in each group. I, S, D refer to insertion, substitution, and deletion, respectively.

8.5 . More Studies of Non-autoregressive Translation with Translation Memories

In the previous section, we experiment with the Edit-LevT model to perform fine-tuning on the MT with TM task. In this section, we look more in-depth at the task of NAT with TM. To the best of our knowledge, this task has not been studied yet in the literature.

8.5.1 . Incorporating Translation Memories with TM-LevT

We adapt our Edit-LevT model to TM-LevT to better fit this task. The model architecture of TM-LevT does not change compared to Edit-LevT. The only difference is in the input source format (see Table 8.13). As LevT-based models only support deletions and insertions, reordering operations can only be obtained by first deleting a token, then reinserting it at another position. As these operations are performed independently, they may cause the erasure of valid words from a similar translation $\tilde{\mathbf{e}}$. To mitigate this risk, we make sure that $\tilde{\mathbf{e}}$ is always fully available to the decoder by concatenating $\tilde{\mathbf{e}}$ to the source $\mathbf{f}$ to make the input sequence, as done in (Bulte and Tezcan, 2019) and in Edit-MT in Chapter 7. Therefore, the initial translation $\tilde{\mathbf{e}}$ in TM-LevT is present on both the source and target side as initialization.

Similar to Section 8.4, we concatenate parallel data and TM augmented data for training. In this section, we discard the `related` data and only use complete similar translations retrieved from the TM. The initial $\tilde{\mathbf{e}}$ for

parallel data is considered empty for both the encoder and decoder sides. We again use a probability of $p = 0.5$ to use $\tilde{\mathbf{e}}$ or the noised subsequence $\mathbf{e}'$ as decoder initialization. For inference, we follow the same input format as training when performing NAT with TMs.

8.5.2 . Datasets

The dataset we use is the same as in Section 8.4. However, to extend the applicability of using TMs, we perform a different range of fuzzy matches, requiring a fuzzy match score $\mathrm{FM} \in [0.4, 1)$. For each domain, we prepare two test sets with $1{,}000$ sentences each: one containing randomly selected sentences having a close match ($\mathrm{FM} \geq 0.6$) in the TM, the other containing sentences with an acceptable match ($\mathrm{FM} \in [0.4, 0.6)$). The remaining data is used for training. Details about the retrieved TM ratios for both FM ranges are shown in Table 8.12. By releasing the fuzzy match score range to 0.4, the number of sentences with TM matches approximately doubles. We also use all retrieved (up to 3) TM matches for training and only the best match for testing. This time, the initial set of 4.4M parallel sentences is thus augmented with about 5M examples for which a TM match is available. The Byte Pair Encoding (BPE) model and vocabulary are the same as in Section 8.4. In this section, we only consider using the complete sequence of retrieved TM matches.

Domain	Raw	$\mathrm{FM} \geq 0.6$	$\mathrm{FM} \in [0.4, 0.6)$
ECB	195,956	51.73%	14.06%
EME	373,235	65.68%	12.65%
Epp	2,009,489	10.12%	25.30%
GNO	55,391	39.31%	11.06%
JRC	503,437	50.87%	16.67%
KDE	180,254	36.00%	10.81%
News	151,423	2.12%	9.65%
PHP	16,020	34.93%	12.38%
TED	159,248	11.90%	26.64%
Ubu	9,314	20.32%	8.26%
Wiki	803,704	19.87%	17.32%
Total	4,457,471	24.27%	20.00%

Table 8.12: Dataset statistics for NAT with TMs experiments, with ratios of sentences with at least one TM match for various similarity ranges.

8.5.3 . Experimental Settings

We follow the same training configuration and procedure as Edit-LevT in Section 8.3.1 to train TM-LevT. We compare TM-LevT with the `FM` approach

of Bulte and Tezcan (2019). Note that the training data used in this section is different from Section 8.4, as it contains more TM matches with an FM score greater than 0.4. We also train a LevT-FM model with exactly the same training data as FM. That is to say, LevT-FM also has TMs concatenated on the source side, while the target side is a vanilla LevT decoder. Table 8.13 gives a summarization of all compared models. These baselines use the same training data as TM-LevT, and process examples with and without TM matches on the source side. Contrarily to TM-LevT, TM matches for these models only appear concatenated to the source sentence, and translation always starts from scratch for both baselines. We also report results of a "do-nothing" baseline which simply outputs the TM matches. Performance is measured by SacreBLEU (Post, 2018) and COMET (Rei et al., 2020).

Name	Architecture	Extended Source	Target Initial
FM	Transformer	✓	-
FM#	Transformer	✓(related segments)	-
Edit-MT	Transformer	✓	-
LevT	LevT	✗	✓
Edit-LevT	LevT + init-del	✗	✓
LevT-FM	LevT	✓	✗
TM-LevT	LevT + init-del	✓	✓

Table 8.13: Summarization of the architecture and data format of different models used in this chapter. *Extended source* implies concatenating the initial translation or similar TM match $\tilde{\mathbf{e}}$ to the source sentence. *Target initial* refers to initializing the decoder with $\tilde{\mathbf{e}}$. *LevT + init-del* is our proposed model with an additional initial deletion operation.

8.5.4 . Main Results

We evaluate the performance of both standard MT and MT with TMs on the two test sets introduced in Section 8.5.2. We report aggregated results computed on the concatenation of all domains (11k sentences) in the main text (Table 8.14). The results broken down by domains when performing MT with TMs on the test sets with $\mathrm{FM} \geq 0.6$ are also shown in Tables 8.15 and 8.16. Detailed results with a breakdown by domains for other settings are shown in Appendix D. As shown in Table 8.14, the FM baseline using TMs yields higher BLEU and COMET scores than the standard MT without TMs in both test sets. LevT-FM can also make good use of TM matches, but its performance lags way behind the FM model. However, unlike Gu et al. (2019), who claim that the original LevT can perform zero-shot automatic post-editing, our results (+tgt TM in Table 8.14) show that, given a trained LevT-FM model, when we initialize the inference with $\tilde{\mathbf{e}}$ instead of an empty

	$\mathrm{FM} \geq 0.6$		$\mathrm{FM} \in [0.4, 0.6)$	
BLEU ↑	w/o TM	w/ TM	w/o TM	w/ TM
copy	-	52.6	-	34.5
FM	51.2	**67.1**	**46.1**	**55.7**
LevT-FM	46.5	60.4	40.8	49.3
+tgt TM	-	52.8	-	35.0
TM-LevT	**52.6**	65.9	45.7	53.3
COMET ↑	w/o TM	w/ TM	w/o TM	w/ TM
copy	-	0.1330	-	-0.3784
FM	**0.6143**	**0.6985**	**0.5379**	**0.5900**
LevT-FM	0.4251	0.5767	0.3429	0.4404
+tgt TM	-	0.1639	-	-0.3478
TM-LevT	0.5314	0.6454	0.4263	0.4889

Table 8.14: BLEU and COMET scores on aggregated multi-domain test sets for various TM similarity ranges. *w/o TM* is standard MT, *w/ TM* adds a retrieved match on the source side and uses the match to initialize TM-LevT inference. *Copy* simply copies the retrieved match in the output. *+tgt TM* refers to using TM match as the initial target for LevT-FM.

BLEU ↑	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	59.8	64.5	34.4	70.3	67.6	55.3	12.0	38.6	30.8	51.6	47.4	52.6
FM	**71.9**	72.0	**58.9**	**80.1**	**83.2**	67.3	**28.8**	**44.7**	**63.3**	**67.6**	68.6	**67.1**
LevT-FM	62.4	53.8	55.5	77.5	78.8	63.3	26.1	28.7	60.2	66.0	67.1	60.4
+tgt TM	60.2	63.8	34.8	69.6	67.7	54.5	12.5	38.8	31.1	52.1	47.5	52.8
TM-LevT	69.8	**72.2**	56.0	78.1	82.2	**68.2**	26.0	44.1	60.3	66.3	**68.7**	65.9

Table 8.15: BLEU scores for each domain when performing **MT with TMs** with $\mathrm{FM} \geq \mathbf{0.6}$. *All* is computed by concatenating all test sets (11k sentences in total).

sequence, LevT-FM is not able to make proper edits and almost considers $\tilde{\mathbf{e}}$ as the final translation, even though the TM match $\tilde{\mathbf{e}}$ is always present on the source side. This difference may come from the fact that editing NMT outputs and TMs are quite different, even for human translators (Sánchez-Gijón et al., 2019), as mentioned in Section 3.3.

Surprisingly, TM-LevT performs remarkably well when translating from scratch, even surpassing the FM model on BLEU scores on the $\mathrm{FM} \geq 0.6$ set, which is arguably easier as it is more similar to the training data. Compared to LevT-FM, TM-LevT significantly improves the performance of standard translation. Note that these models are trained without using KD on the target side, which implies that initializing the TM-LevT decoder with TMs may be an orthogonal method to KD to improve standard translation quality. When translating using TM matches, TM-LevT almost closes the gap with the FM approach on BLEU, especially for the subset containing higher quality

COMET ↑	copy	FM	LevT-FM	LevT-FM + tgt TM	TM-LevT
ECB	0.4006	**0.7288**	0.5647	0.4086	0.6792
EME	0.4625	**0.7211**	0.3384	0.4573	0.7003
Epp	-0.0797	**0.8223**	0.7608	-0.0075	0.7591
GNO	0.4893	**0.9143**	0.8617	0.5230	0.8699
JRC	0.6893	**0.9954**	0.9355	0.7062	0.9696
KDE	0.1150	**0.6299**	0.5683	0.1437	0.6106
News	-0.6083	**0.3318**	0.2443	-0.5679	0.2093
PHP	-0.1977	**0.0801**	-0.2183	-0.1957	0.0353
TED	-0.4184	**0.7910**	0.7203	-0.3328	0.6923
Ubu	0.3296	**0.8610**	0.8091	0.3710	0.8155
Wiki	0.2843	**0.8110**	0.7618	0.3008	0.7614
All	0.1330	**0.6985**	0.5767	0.1639	0.6454

Table 8.16: COMET scores for each domain when performing **MT with TMs** with $\mathrm{FM} \geq \mathbf{0.6}$.

matches, where it even outperforms the FM approach in some domains, as shown in Table 8.15. The gap in COMET score between TM-LevT and the FM model (see Table 8.16) is much smaller than reported by Helcl et al. (2022), indicating that TM-LevT actually outputs valid translations.

Figure 8.2 illustrates the performance gains obtained by translating with TMs of different similarity ranges. The FM score ranges also indicate the similarities between the test sentences and the training data. Higher similarities imply that the test sets are closer to the training data, therefore, easier to translate from scratch. For both FM and TM-LevT, using more similar TMs yields larger improvements, while using less similar TMs can still obtain significant performance gains. However, the improvements obtained by TM-LevT are always smaller than FM in both cases. The benefit of using less similar TM matches is important as high-similarity TMs are often difficult to find in real-world applications.

The FM approaches are known to improperly copy "unrelated tokens" from TM matches into output translations, as we report in (Xu et al., 2020). We define here unrelated tokens as those present in $\tilde{\mathbf{e}}$ but not in $\mathbf{e}$, which is different from Xu et al. (2020). More precisely, we compute the edit operations between $\tilde{\mathbf{e}}$ and $\mathbf{e}$ based on edit distance and count tokens in $\tilde{\mathbf{e}}$ that should be deleted or substituted as unrelated tokens. We then compute the edit operations between $\tilde{\mathbf{e}}$ and the hypothesis $\hat{\mathbf{e}}$ produced by our models to evaluate if these tokens are correctly deleted or mistakenly kept in the output. We omit the use of a word alignment model to detect unrelated tokens aligned with the source $\mathbf{f}$, as introduced in Section 7.5.1, as word alignment models are not perfect and yield additional errors that may lead to an imprecise measure of unrelated tokens. Table 8.17 reports the percentage of such unrelated tokens that are not removed from the hypotheses. TM-

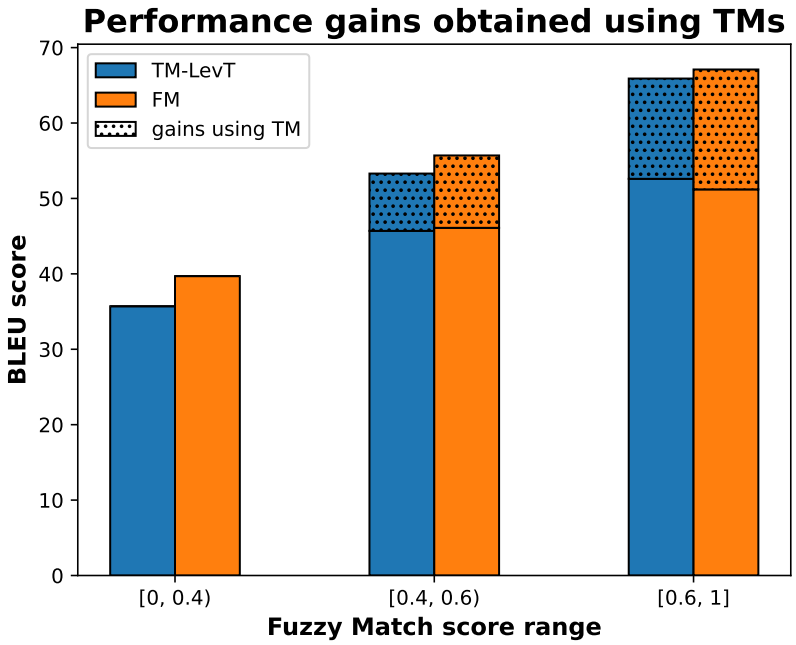


Figure 8.2: Comparison of performance gains when translating with TMs of different similarity ranges for FM and TM-LevT. $\mathrm{FM} \in [0, 0.4)$ is a held-out set of $1{,}000$ sentences per domain with no match found.

LevT is less prone than the FM model to recopy unrelated parts of the TM matches. LevT-FM does even better in this setting, but its lower BLEU scores suggest that this approach also discards related tokens.

Unrelated token rate ↓	$\mathrm{FM} \geq 0.6$	$\mathrm{FM} \in [0.4, 0.6)$
FM	28.42	17.78
LevT-FM	21.39	13.74
TM-LevT	26.67	16.56

Table 8.17: The average percentage of unrelated tokens from the retrieved TM matches appearing in the final translations evaluated on the multi-domain test sets.

8.5.5 . The Effect of Knowledge Distillation

KD is now the "by default" technique used in most NAT models, as it reduces the complexity and lexical diversity of target sentences, thereby helping NAT approaches mitigate the multimodality issue (Zhou et al., 2020; Xu et al., 2021b). Here we conduct experiments with KD to evaluate its effectiveness in NAT with TMs. We train a Transformer-base teacher model on the 4.4M parallel data and use it for distillation. The distillation process is done by translating the source sentences using the teacher model with

beam search. As we use TMs, which are also in the target language, we thus consider two situations: applying KD only to the target reference and to both TMs used in training and the reference. Note that applying KD simply replaces the target reference and/or TMs with the corresponding distilled data generated by the AR teacher model, while all other settings are unchanged.

As expected, using KD on the reference side does improve the performance of TM-LevT on standard translation, as reported in Table 8.18. This further confirms the hypothesis in Section 8.5.4 that KD is complementary to initializing the TM-LevT decoder with TMs for standard translation. However, KD seems unnecessary when editing a similar initial translation (w/TM) and yields a significant drop in scores compared to using real data for training. One possible explanation is that there is a mismatch between real TM matches and distilled training references. However, this mismatch also exists in training as we perform inference under the same condition as training. Applying KD also to TM matches crosses out this mismatch both for training and inference. Nevertheless, it still hurts the performance on both tasks with and without TMs.

	$\mathrm{FM} \geq 0.6$		$\mathrm{FM} \in [0.4, 0.6)$	
BLEU	w/o TM	w/ TM	w/o TM	w/ TM
TM-LevT	52.6	65.9	45.7	53.3
+ KD	54.3	57.1	47.6	49.3
+ KD TM	53.8	56.0	47.3	48.5

Table 8.18: BLEU scores for KD effects on the aggregated multi-domain test sets. + *KD* applies KD to the training references, + *KD TM* applies KD to both references and TM matches.

8.5.6 . Ablation Analysis

We further study the effectiveness of each component of our method. We experiment with several settings and train a new model for each contrast:

- Removing the initial translation from the target side (- tgt TM), which makes the model similar to the original LevT model, but with an additional init-del operation;
- Removing the TM from the source side (- src TM);
- Removing the final-del operation in training (- final-del);
- Using the reference deletion operation for init-del instead of the self-prediction operation (- self-pred).

As illustrated in Table 8.19, training without TM matches on the target side vastly degrades the scores in all test conditions, indicating again that standard MT can also benefit from training with TMs as initial targets. This

also implies that only performing the init-del without initial translations is meaningless. On the contrary, training without TM matches on the source side improves standard MT, as also pointed out by Bulte and Tezcan (2019) and found in Section 7.4.3. However, it has a negative impact when translating with TMs. This highlights the benefits of remembering $\tilde{\mathbf{e}}$ on the source side.

	$\mathrm{FM} \geq 0.6$		$\mathrm{FM} \in [0.4, 0.6)$	
BLEU	w/o TM	w/ TM	w/o TM	w/ TM
TM-LevT	52.6	65.9	45.7	53.3
- tgt TM	46.6	60.7	40.7	49.6
- src TM	53.2	64.3	45.9	52.2
- final-del	38.5	64.2	32.7	50.8
- self-pred	52.6	65.2	45.6	52.7

Table 8.19: BLEU scores for various configurations. - *tgt TM* (resp. - *src TM*) is the model trained without TM match on the target (resp. source) side. - *final-del* is trained without the final-del operation, - *self-pred* only applies reference deletions during training.

We also compare alternative implementations of the deletion operation. Results in Table 8.19 (- final-del) show that removing the final deletion operation mainly impacts TM-LevT in the standard MT setting, where the detection of wrong predictions matters most (Huang et al., 2022). This further demonstrates that unrelated tokens from TMs and prediction errors made by the model are quite different, and training with both operations is necessary. Last, we experiment by using only reference deletion labels to train the insertion operation instead of using the union of both the reference and model predictions (see Section 8.2 and Figure 8.1). We observe (- self-pred) a slight performance drop with respect to the complete TM-LevT, suggesting that this component is less important in the model than others.

8.5.7 . Inference Efficiency with TM-LevT

	$\mathrm{FM} \geq 0.6$		$\mathrm{FM} \in [0.4, 0.6)$	
Decoding iterations	w/o TM	w/ TM	w/o TM	w/ TM
LevT-FM	2.027	1.899	2.544	2.538
TM-LevT	1.781	1.348	2.260	1.880

Table 8.20: Averaged decoding iterations per sentence evaluated on the aggregated multi-domain test sets.

Using TMs in MT is expected to not only improve the translation quality but also reduce the decoding load, as certain segments in TMs can be directly

reused, and models only need to edit a similar translation. However, the decoding load is not really reduced in FM or LevT-FM, as they always perform a standard translation process with only an extended source input. The decoding procedure always starts from scratch. TM-LevT, on the contrary, uses an initial translation to speed up decoding. We measure the average number of iterations needed in inference in Table 8.20. As can be seen, translating with TMs reduces the decoding effort for TM-LevT by about 20% fewer iterations, while LevT-FM remains almost unchanged. We also find that TM-LevT needs fewer iterations to converge than LevT-FM in all conditions.

Decoding time	FM		TM-LevT	
($\times 10^{-3}$s)	w/o TM	w/ TM	w/o TM	w/ TM
ECB	10.78	18.66	3.76	4.77
EME	11.52	14.04	2.93	4.00
Epp	4.54	5.35	2.21	2.95
GNO	5.85	6.90	2.40	3.09
JRC	7.89	10.41	2.93	3.96
KDE	4.48	5.49	2.38	2.87
News	3.16	4.03	2.00	3.04
PHP	25.56	12.60	2.75	4.44
TED	2.73	3.65	2.04	2.90
Ubu	3.43	3.69	1.95	2.59
Wiki	4.68	5.82	2.65	4.13
Average	7.69	8.24	2.55	3.52

Table 8.21: Average decoding time ($\times 10^{-3}$s) per sentence evaluated on the multi-domain test sets for both $\mathrm{FM} \geq 0.6$ and $\mathrm{FM} \in [0.4, 0.6)$.

The main advantage of using NAT models is the inference speedup when generating multiple tokens in a single decoding step. Table 8.21 compares the average decoding time per sentence on each domain for the AR FM approach and TM-LevT. TM-LevT is much faster than FM when performing MT both with and without TMs. Even though Helcl et al. (2022) state that NAT models are not as fast as it seems since the evaluation condition matters, we use the same hardware condition and inference batch size for all cases, making our results comparable, at least in our scenario. The decoding time of FM greatly varies across domains, while it is more stable for TM-LevT. This is mainly due to the length of generated sequence. Generating longer sequences for FM takes more time, while NAT models can parallel the generation process of all tokens. We also notice that TM-LevT takes more time when translating with TMs than without TMs, even though it requires fewer decoding iterations. This is probably due to the data processing step when initializing the target sequence with a TM. Besides, encoding an ex-

tended source input also takes more time, which is consistent for both FM and TM-LevT.

8.6 . Conclusion

In this chapter, we studied Bi-sync with non-autoregressive approaches to simulate the bilingual writing scenario. We demonstrated that non-autoregressive Edit-LevT could also perform well on the general Bi-sync tasks. Multilingual Edit-LevT even outperformed monolingual ones on the more realistic compression and extension tasks. Edit-LevT could also be quickly adapted to multi-domain MT with TMs tasks, where it could outperform strong AR baselines when tested on unseen domains. In the in-depth study of NAT with TMs, we found that by copying the TM matches both on the source side and on the target side as an initial target sequence, our TM-LevT model vastly outperformed the LevT-FM model and achieved BLEU scores that approached those of a strong AR model both when decoding from scratch and when editing a TM match. TM-LevT also generated translations that contained fewer unrelated tokens and reduced the decoding load with fewer iterations. Finally, we demonstrated that training with TMs helps to improve NAT performance on standard MT without resorting to KD.

In our future work, we would like to experiment with actual users in realistic interactions and explore questions related to human-machine interaction in a bilingual writing assistant system, *e.g.*, the visualization of automatic edits or verifying that the user inputs are not erased by the system, etc. Besides, we would like to further explore more in NAT systems, which are computationally faster, and try to improve their performance to generate high-quality automatic edits within a low latency. For this, we intend to study a carefully designed training curriculum. We would also like to experiment on other language pairs for the NAT with TMs task, especially on more distant language pairs, to further validate the TM-LevT model.

9 - Conclusion

In this dissertation, we have explored new methods to assist bilingual writing in two different simulated situations. The first simulation allows a single sentence to be composed in mixed-language (MXL), which is then translated into monolingual sentences in both component languages. The second scenario assumes texts in two languages are separate and makes it possible to write in either language, while the other language is automatically updated to keep both sides synchronized. In both situations, our ultimate purpose is to perform double authoring, *i.e.*, to have two versions of the same text in the two languages of equally good quality.

For the first case, we have thoroughly studied the task of generating monolingual sentences from an MXL input. We proposed the dual decoder model, a single model capable of simultaneously generating texts in both languages. We also proposed a method to generate synthetic MXL data from parallel sentences and showed that the artificial data could effectively train our dual decoder model to translate both regular monolingual and MXL sentences. Our dual decoder model achieved results on par with baseline approaches, containing two separately trained translation models for each language. The proposed dual decoder model could almost perfectly disentangle tokens in one language from the other language and make necessary corrections when the input sentence contained errors. We have achieved state-of-the-art performance on the SemEval 2014 L2 Writing Assistant task for the Spanish-English direction. Besides, we further applied the dual decoder model to four other tasks and discovered that dual decoding is prone to exposure bias. We explored practical remedies to mitigate this issue and have achieved similar performance compared to simple multi-task approaches while constantly generating more consistent translation pairs. We also demonstrated that the dual decoder model could benefit from pre-trained regular translation models to achieve improved translation quality in various tasks, alleviating the common data scarcity problem in the four situations where generating synthetic data was not trivial.

In the second situation, we defined the framework of bilingual synchronization (Bi-sync), which aims at editing existing initial translations to restore parallelism with a source sentence. We proposed various methods to simulate initial translations that require different types of edits. The artificial editing data was effective in training both autoregressive and non-autoregressive models. Both our proposed autoregressive Edit-MT and the non-autoregressive Edit-LevT could make good use of initial translations to reach high translation quality, even in real-world scenarios. Experiments in multilingual settings demonstrated that one single model could perform Bi-

sync in both directions without decreasing the overall performance, making it possible to treat both languages equivalently without distinguishing a source language. In addition, we studied more specific tasks with edit-based models. We demonstrated that models trained for the general Bi-sync task could be quickly adapted to parallel corpus cleaning and neural machine translation (NMT) with translation memories (TMs) tasks with only slight fine-tuning. The performance even outperformed dedicated systems trained on specific tasks. We paid particular attention to the non-autoregressive translation (NAT) with TMs task and explored the TM-LevT model, which greatly surpassed the original Levenshtein Transformer model and approached a strong autoregressive method with a lower appearance rate of unrelated words and a much faster decoding speed. Training with TMs also improved the performance of TM-LevT on standard translation without using knowledge distillation (KD). Our work is one of the first studies in this new subdomain of NAT and translation with TM. We hope it can motivate more research work to further explore this area.

Future Outlook

Along with our work, there still remain several promising research directions. We discuss below some future perspectives with open questions that emerged from our study in this thesis.

Our work in this thesis was based on simulated situations where we statically performed translations without actual interaction with humans. It will be a promising direction to design user interfaces, collect real user feedback to evaluate the differences between simulated conditions and real-world scenarios, study more precisely how expert and non-expert translators will use this framework, and improve bilingual writing systems to suit this use case better. It is also interesting to contrast our systems with post-editing NMT outputs or interactive translation systems in terms of edition time, especially for texts that need incremental updates.

As mentioned in Chapter 6, the exposure bias issue is amplified in the dual decoder model. We proposed to generate partial synthetic reference data to mitigate this problem but have not stepped further. Future work may concentrate on developing other methods to better alleviate this problem so that the dual decoder model can generate high-quality and consistent translations.

NAT is a newly emerged domain in the area of NMT. However, due to the conditional independence assumption, the performance of NAT models is mostly lagging behind autoregressive baselines. Besides, NAT models highly rely on KD from autoregressive teacher models. KD implicitly defines the upper bound performance of NAT models to be around the teacher model.

Even though there were studies analyzing why KD is effective for NAT, it is still a mystery why KD can improve NAT. More work and better analyses could be conducted to determine the key reason behind KD. We showed that training with TMs could improve NAT performance without using KD. We hope this provides a new direction and wish our work can inspire more studies on NAT models without relying on KD.

Bibliography

Sadaf Abdul Rauf and François Yvon. 2020. Document level contexts for neural machine translation. Research Report 2020-003, LIMSI-CNRS.

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia. Association for Computational Linguistics.

Gustavo Aguilar and Thamar Solorio. 2020. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8033–8044, Online. Association for Computational Linguistics.

Vicent Alabau, Christian Buck, Michael Carl, Francisco Casacuberta, Mercedes García-Martínez, Ulrich Germann, Jesús González-Rubio, Robin Hill, Philipp Koehn, Luis Leiva, Bartolomé Mesa-Lao, Daniel Ortiz-Martínez, Herve Saint-Amand, Germán Sanchis Trilles, and Chara Tsoukala. 2014. CASMACAT: A computer-assisted translation workbench. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 25–28, Gothenburg, Sweden. Association for Computational Linguistics.

Robert B. Allen. 1987. Several studies on natural language and back-propagation. In *Proceedings of the IEEE First International Conference on Neural Networks*, volume 2, pages 335–341, Piscataway, NJ, USA. IEEE.

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. 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 82–91, New Orleans, Louisiana. Association for Computational Linguistics.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kelsey Ball and Dan Garrette. 2018. Part-of-speech tagging for code-switched, transliterated texts without explicit language identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3084–3089, Brussels, Belgium. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 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, Ann Arbor, Michigan. Association for Computational Linguistics.

Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. 2022. latent-GLAT: Glancing at latent variables for parallel text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8398–8409, Dublin, Ireland. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine 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

1538–1548, Hong Kong, China. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Toms Bergmanis and Mārcis Pinnis. 2021. Facilitating terminology translation with target lemma annotations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3105–3111, Online. Association for Computational Linguistics.

Michael Bloodgood and Benjamin Strauss. 2014. Translation memory retrieval methods. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 202–210, Gothenburg, Sweden. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Julien Bourdaillet, Stéphane Huet, Philippe Langlais, and Guy Lapalme. 2011. TransSearch: from a bilingual concordancer to a translation finder. *Machine Translation*, 24(3):241.

Amit Bronner, Matteo Negri, Yashar Mehdad, Angela Fahrni, and Christof Monz. 2012. Cosyne: Synchronizing multilingual wiki content. In *Proceedings of the Eighth Annual International Symposium on Wikis and Open Collaboration*, WikiSym '12, New York, NY, USA. Association for Computing Machinery.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Bram Bulte and Arda Tezcan. 2019. Neural fuzzy repair: Integrating fuzzy matches into neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1800–1809, Florence, Italy. Association for Computational Linguistics.

Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural machine translation with monolingual translation memory. 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 7307–7318, Online. Association for Computational Linguistics.

Marine Carpuat, Yogarshi Vyas, and Xing Niu. 2017. Detecting cross-lingual semantic divergence for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 69–79, Vancouver. Association for Computational Linguistics.

Isaac Caswell, Ciprian Chelba, and David Grangier. 2019. Tagged back-translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 53–63, Florence, Italy. Association for Computational Linguistics.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Annual conference of the European Association for Machine Translation*, pages 261–268, Trento, Italy. European Association for Machine Translation.

Mei-Hua Chen, Shih-Ting Huang, Hung-Ting Hsieh, Ting-Hui Kao, and Jason S. Chang. 2012. FLOW: A first-language-oriented writing assistant system. In *Proceedings of the ACL 2012 System Demonstrations*, pages 157–162, Jeju Island, Korea. Association for Computational Linguistics.

Shuguang Chen, Gustavo Aguilar, Anirudh Srinivasan, Mona Diab, and Thamar Solorio. 2022. Calcs 2021 shared task: Machine translation for code-switched data. *CoRR*, abs/2202.09625.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy. Association for Computational Linguistics.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers), pages 1756–1766, Melbourne, Australia. Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Eunah Cho, Jan Niehues, and Alex Waibel. 2012. Segmentation and punctuation prediction in speech language translation using a monolingual translation system. In *Proceedings of the 9th International Workshop on Spoken Language Translation: Papers*, pages 252–259, Hong Kong, Table of contents.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 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.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Gyu-Hyeon Choi, Jong-Hun Shin, and Young-Kil Kim. 2018. Improving a multi-source neural machine translation model with corpus extension for low-resource languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Shamil Chollampatt, Raymond Hendy Susanto, Liling Tan, and Ewa Szymanska. 2020. Can automatic post-editing improve NMT? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2736–2746, Online. Association for Computational Linguistics.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1304–1319, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 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.

Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. 2019. On the use of BERT for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 108–117, Hong Kong. Association for Computational Linguistics.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7059–7069. Curran Associates, Inc.

Gonçalo M. Correia and André F. T. Martins. 2019. A simple and effective approach to automatic post-editing with transfer learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3050–3056, Florence, Italy. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Josep M. Crego, José B. Mariño, and Adrià de Gispert. 2005. Reordered search, and tuple unfolding for ngram-based SMT. In *Proceedings of Machine Translation Summit X: Papers*, pages 283–289, Phuket, Thailand.

Josep M. Crego, José B. Marino, and Adria de Gispert. 2004. Finite-state-based and phrase-based statistical machine translation. In *Proc. Interspeech 2004*, pages 37–40.

Josep Maria Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurélien Coquard, Yongchao Deng, Satoshi Enoue, Chiyo Geiss, Joshua Johanson, Ardas Khalsa, Raoum Khiari, Byeongil Ko, Catherine Kobus, Jean Lorieux, Leidiana Martins, Dang-Chuan Nguyen, Alexandra Priori, Thomas Riccardi, Natalia Segal, Christophe Servan, Cyril Tiquet, Bo Wang, Jin Yang, Dakun Zhang, Jing Zhou, and Peter Zoldan. 2016. Systran's pure neural machine translation systems. *CoRR*, abs/1610.05540.

Josep Maria Crego, Aurélien Max, and François Yvon. 2010. Local lexical adaptation in machine translation through triangulation: SMT helping

SMT. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 232–240, Beijing, China. Coling 2010 Organizing Committee.

Josep Maria Crego and Jean Senellart. 2016. Neural machine translation from simplified translations. *CoRR*, abs/1612.06139.

Alister Cumming. 1989. Writing expertise and second-language proficiency*. *Language Learning*, 39(1):81–135.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. 2020. A survey of multilingual neural machine translation. *ACM Comput. Surv.*, 53(5).

Raj Dabre, Fabien Cromieres, and Sadao Kurohashi. 2017. Enabling multi-source neural machine translation by concatenating source sentences in multiple languages. *CoRR*, abs/1702.06135.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, and Stephan Vogel. 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.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021a. Progressive multi-granularity training for non-autoregressive translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2797–2803, Online. Association for Computational Linguistics.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. 2021b. Understanding and improving lexical choice in non-autoregressive translation. In *International Conference on Learning Representations*.

Shuoyang Ding, Marcin Junczys-Dowmunt, Matt Post, and Philipp Koehn. 2021c. Levenshtein training for word-level quality estimation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6724–6733, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Georgiana Dinu, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan. 2019. Training neural machine translation to apply terminology constraints. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3063–3068, Florence, Italy. Association for Computational Linguistics.

Félix do Carmo, Dimitar Shterionov, Joss Moorkens, Joachim Wagner, Murhaf Hossari, Eric Paquin, Dag Schmidtke, Declan Groves, and Andy Way. 2021. A review of the state-of-the-art in automatic post-editing. *Machine Translation*, 35(2):101–143.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language 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 1723–1732, Beijing, China. Association for Computational Linguistics.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2849–2859. PMLR.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 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, Atlanta, Georgia. Association for Computational Linguistics.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. 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 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Marc'Aurelio Ranzato, and Michael Auli. 2020. On the evaluation of machine translation systems trained with back-translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2836–2846, Online. Association for Computational Linguistics.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2020. Efficient Wait-k Models for Simultaneous Machine Translation. In *Interspeech 2020 - Conference of the International Speech Communication Association*, pages 1461–1465, Shangai (Virtual Conf), China.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. WikiAtomicEdits: A multilingual corpus of Wikipedia edits for modeling language and discourse. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 305–315, Brussels, Belgium. Association for Computational Linguistics.

Christian Federmann and William D. Lewis. 2016. Microsoft speech language translation (MSLT) corpus: The IWSLT 2016 release for English, French and German. In *Proceedings of the 13th International Conference on Spoken Language Translation*, Seattle, Washington D.C. International Workshop on Spoken Language Translation.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Andrew Finch and Eiichiro Sumita. 2009. Bidirectional phrase-based statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1124–1132, Singapore. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875. Association for Computational Linguistics.

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas. Association for Computational Linguistics.

George Foster, Philippe Langlais, and Guy Lapalme. 2002. User-friendly text prediction for translators. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 148–155. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Ekaterina Garmash and Christof Monz. 2016. Ensemble learning for multi-source neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1409–1418, Osaka, Japan. The COLING 2016 Organizing Committee.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. 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 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Kyle Gorman. 2016. Pynini: A Python library for weighted finite-state grammar compilation. In *Proceedings of the SIGFSM Workshop on Statistical*

NLP and Weighted Automata, pages 75–80, Berlin, Germany. Association for Computational Linguistics.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 366–371, Vancouver, Canada. Association for Computational Linguistics.

David Grangier and Michael Auli. 2018. QuickEdit: Editing text & translations by crossing words out. 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 272–282, New Orleans, Louisiana. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA. Association for Computing Machinery.

Spence Green, Sida I. Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and Christopher D. Manning. 2014. Human effort and machine learnability in computer aided translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1225–1236, Doha, Qatar. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018a. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 120–133, Online. Association for Computational Linguistics.

Jiatao Gu and Xu Tan. 2022. Non-autoregressive sequence generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 21–27, Dublin, Ireland. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, volume 32, pages 11181–11191. Curran Associates, Inc.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. 2018b. Search engine guided neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3723–3730.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. 2020. Incorporating bert into parallel sequence decoding with adapters. In *Advances in Neural Information Processing Systems*, volume 33, pages 10843–10854. Curran Associates, Inc.

Abhirut Gupta, Aditya Vavre, and Sunita Sarawagi. 2021. Training data augmentation for code-mixed translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5760–5766, Online. Association for Computational Linguistics.

Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2020. A semi-supervised approach to generate the code-mixed text using pre-trained encoder and transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2267–2280, Online. Association for Computational Linguistics.

Barry Haddow, Rachel Bawden, Antonio Valerio Miceli Barone, Jindřich Helcl, and Alexandra Birch. 2022. Survey of Low-Resource Machine Translation. *Computational Linguistics*, pages 1–60.

Eva Hasler, Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2018. Neural machine translation decoding with terminology constraints. 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 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

Hao He, Qian Wang, Zhipeng Yu, Yang Zhao, Jiajun Zhang, and Chengqing Zong. 2021a. Synchronous interactive decoding for multilingual neural

machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12981–12988.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Qiuxiang He, Guoping Huang, Qu Cui, Li Li, and Lemao Liu. 2021b. Fast and accurate neural machine translation with translation memory. 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 3170–3180, Online. Association for Computational Linguistics.

Jindřich Helcl, Barry Haddow, and Alexandra Birch. 2022. Non-autoregressive machine translation: It's not as fast as it seems. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1780–1790, Seattle, United States. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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

Chris Hokamp. 2017. Ensembling factored neural machine translation models for automatic post-editing and quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 647–654, Copenhagen, Denmark. Association for Computational Linguistics.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019. Improved lexically constrained decoding for translation and monolingual rewriting. 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 839–850, Minneapolis, Minnesota. Association for Computational Linguistics.

Chung-chi Huang, Ping-che Yang, Keh-jiann Chen, and Jason S. Chang. 2012. Transahead: A computer-assisted translation and writing tool. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 352–356, Montréal, Canada. Association for Computational Linguistics.

Guoping Huang, Lemao Liu, Xing Wang, Longyue Wang, Huayang Li, Zhaopeng Tu, Chengyan Huang, and Shuming Shi. 2021. Transmart: A practical interactive machine translation system. *CoRR*, abs/2105.13072.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2022. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Julia Ive and François Yvon. 2016. Parallel sentence compression. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1503–1513, Osaka, Japan. The COLING 2016 Organizing Committee.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Zhifeng Wu, Yonghui andhen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2018. MS-UEdin submission to the WMT2018 APE shared task: Dual-source transformer for automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 822–826, Belgium, Brussels. Association for Computational Linguistics.

Alina Karakanta, Marco Gaido, Matteo Negri, and Marco Turchi. 2021. Between flexibility and consistency: Joint generation of captions and subtitles. In *Proceedings of the 18th International Conference on Spoken Language Translation (IWSLT 2021)*, pages 215–225, Bangkok, Thailand (online). Association for Computational Linguistics.

Alina Karakanta, Matteo Negri, and Marco Turchi. 2020a. Is 42 the answer to everything in subtitling-oriented speech translation? In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 209–219, Online. Association for Computational Linguistics.

Alina Karakanta, Matteo Negri, and Marco Turchi. 2020b. MuST-cinema: a speech-to-subtitles corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3727–3734, Marseille, France. European Language Resources Association.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Rebecca Knowles and Philipp Koehn. 2016. Neural interactive translation prediction. In *Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track*, pages 107–120, Austin, TX, USA. The Association for Machine Translation in the Americas.

Rebecca Knowles, John Ortega, and Philipp Koehn. 2018. A comparison of machine translation paradigms for use in black-box fuzzy-match repair. In *Proceedings of the AMTA 2018 Workshop on Translation Quality Estimation and Automatic Post-Editing*, pages 249–255, Boston, MA. Association for Machine Translation in the Americas.

Catherine Kobus, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378, Varna, Bulgaria. INCOMA Ltd.

Philipp Koehn. 2010. Enabling monolingual translators: Post-editing vs. options. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 537–545, Los Angeles, California. Association for Computational Linguistics.

Philipp Koehn, Vishrav Chaudhary, Ahmed El-Kishky, Naman Goyal, Peng-Jen Chen, and Francisco Guzmán. 2020. Findings of the WMT 2020 shared task on parallel corpus filtering and alignment. In *Proceedings of the Fifth Conference on Machine Translation*, pages 726–742, Online. Association for Computational Linguistics.

Philipp Koehn, Francisco Guzmán, Vishrav Chaudhary, and Juan Pino. 2019. Findings of the WMT 2019 shared task on parallel corpus filtering for low-resource conditions. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 54–72, Florence, Italy. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel L. Forcada. 2018. Findings of the WMT 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.

Philipp Koehn and Jean Senellart. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of the Second Joint EM+/CNGL Workshop: Bringing MT to the User: Research on Integrating MT in the Translation Industry*, pages 21–32, Denver, Colorado, USA. Association for Machine Translation in the Americas.

Philipp Koehn, Chara Tsoukala, and Herve Saint-Amand. 2014. Refinements to interactive translation prediction based on search graphs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 574–578, Baltimore, Maryland. Association for Computational Linguistics.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. 2020. Incorporating a local translation mechanism into non-autoregressive translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1067–1073, Online. Association for Computational Linguistics.

Julia Kreutzer and Artem Sokolov. 2018. Learning to segment inputs for NMT favors character-level processing. In *Proceedings of the 15th International Conference on Spoken Language Translation*, pages 166–172, Brussels. International Conference on Spoken Language Translation.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Philippe Langlais, George Foster, and Guy Lapalme. 2000. TransType: a computer-aided translation typing system. In *ANLP-NAACL 2000 Workshop: Embedded Machine Translation Systems*.

Hang Le, Juan Pino, Changhan Wang, Jiatao Gu, Didier Schwab, and Laurent Besacier. 2020. Dual-decoder transformer for joint automatic speech recognition and multilingual speech translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3520–3533, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Dongjun Lee, Junhyeong Ahn, Heesoo Park, and Jaemin Jo. 2021. IntelliCAT: Intelligent machine translation post-editing with quality estimation and translation suggestion. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 11–19, Online. Association for Computational Linguistics.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Jason Lee, Raphael Shu, and Kyunghyun Cho. 2020. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1006–1015, Online. Association for Computational Linguistics.

Mariëlle Leijten, Luuk Van Waes, Karen Schriver, and John R. Hayes. 2014. Writing in the workplace: Constructing documents using multiple digital sources. *Journal of Writing Research*, 5(3):285–337.

Huayang Li, Lemao Liu, Guoping Huang, and Shuming Shi. 2021. GWLAN: General word-level AutocompletioN for computer-aided translation. 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 4792–4802, Online. Association for Computational Linguistics.

Pengfei Li, Liangyou Li, Meng Zhang, Minghao Wu, and Qun Liu. 2022. Universal conditional masked language pre-training for neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6379–6391, Dublin, Ireland. Association for Computational Linguistics.

Jindřich Libovický and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021, Brussels, Belgium. Association for Computational Linguistics.

Jindřich Libovický, Helmut Schmid, and Alexander Fraser. 2022. Why don't people use character-level machine translation? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2470–2485, Dublin, Ireland. Association for Computational Linguistics.

Jessy Lin, Geza Kovacs, Aditya Shastry, Joern Wuebker, and John DeNero. 2022. Automatic correction of human translations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 494–507, Seattle, United States. Association for Computational Linguistics.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Danni Liu, Jan Niehues, and Gerasimos Spanakis. 2020a. Adapting end-to-end speech recognition for readable subtitles. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 247–256, Online.

Lemao Liu, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2020b. Agreement on target-bidirectional recurrent neural networks for sequence-to-sequence learning. *Journal of Artificial Intelligence Research*, 67:581–606.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016a. Agreement on target-bidirectional neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 411–416, San Diego, California. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. 2021a. Confidence-aware scheduled sampling for neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2327–2337, Online. Association for Computational Linguistics.

Yijin Liu, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. 2021b. Scheduled sampling based on decoding steps for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3285–3296, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuanchao Liu, Xin Wang, Ming Liu, and Xiaolong Wang. 2016b. Writerighter: An academic writing assistant system. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

António V. Lopes, M. Amin Farajian, Gonçalo M. Correia, Jonay Trénous, and André F. T. Martins. 2019. Unbabel's submission to the WMT2019 APE shared task: BERT-based encoder-decoder for automatic post-editing. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 118–123, Florence, Italy. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. 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.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. 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 4282–4292, Hong Kong, China. Association for Computational Linguistics.

Esmé Manandise and Claudia Gdaniec. 2011. Morphology to the rescue redux: Resolving borrowings and code-mixing in machine translation. In *Systems and Frameworks for Computational Morphology*, pages 86–97, Berlin, Heidelberg. Springer Berlin Heidelberg.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. 2021. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. 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 7297–7306, Online. Association for Computational Linguistics.

Benjamin Marie and Aurélien Max. 2015. Touch-based pre-post-editing of machine translation output. In *Proceedings of the 2015 Conference on*

Empirical Methods in Natural Language Processing, pages 1040–1045, Lisbon, Portugal. Association for Computational Linguistics.

Benjamin Marie, Raphael Rubino, and Atsushi Fujita. 2020. Tagged back-translation revisited: Why does it really work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5990–5997, Online. Association for Computational Linguistics.

Sameen Maruf, Fahimeh Saleh, and Gholamreza Haffari. 2021. A survey on document-level neural machine translation: Methods and evaluation. *ACM Comput. Surv.*, 54(2).

Mohamed Menacer, David Langlois, Denis Jouvet, Dominique Fohr, Odile Mella, and Kamel Smaïli. 2019. Machine Translation on a parallel Code-Switched Corpus. In *Canadian AI 2019 - 32nd Conference on Canadian Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Ontario, Canada.

Elise Michon, Josep Crego, and Jean Senellart. 2020. Integrating domain terminology into neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3925–3937, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *CoRR*, abs/2112.10508.

Tsvetomila Mihaylova and André F. T. Martins. 2019. Scheduled sampling for transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356, Florence, Italy. Association for Computational Linguistics.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Mehryar Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3):321–350.

Jihyung Moon, Hyunchang Cho, and Eunjeong L. Park. 2020. Revisiting round-trip translation for quality estimation. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 91–104, Lisboa, Portugal. European Association for Machine Translation.

Carol Myers-Scotton. 1997. *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

Matteo Negri, Luisa Bentivogli, Yashar Mehdad, Danilo Giampiccolo, and Alessandro Marchetti. 2011. Divide and conquer: Crowdsourcing the creation of cross-lingual textual entailment corpora. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 670–679, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2012. Semeval-2012 task 8: Cross-lingual textual entailment for content synchronization. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 399–407, Montréal, Canada. Association for Computational Linguistics.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2013. Semeval-2013 task 8: Cross-lingual textual entailment for content synchronization. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 25–33, Atlanta, Georgia, USA. Association for Computational Linguistics.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018. ESCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Graham Neubig, Philip Arthur, and Kevin Duh. 2015. Multi-target machine translation with multi-synchronous context-free grammars. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 293–302, Denver, Colorado. Association for Computational Linguistics.

Jan Niehues. 2021. Continuous learning in neural machine translation using bilingual dictionaries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 830–840, Online. Association for Computational Linguistics.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1828–1836, Osaka, Japan. The COLING 2016 Organizing Committee.

Yuta Nishimura, Katsuhito Sudoh, Graham Neubig, and Satoshi Nakamura. 2018. Multi-source neural machine translation with missing data. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 92–99, Melbourne, Australia. Association for Computational Linguistics.

Xing Niu and Marine Carpuat. 2020. Controlling neural machine translation formality with synthetic supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8568–8575.

Franz Josef Och and Hermann Ney. 2001. Statistical multi-source translation. In *Proceedings of Machine Translation Summit VIII*, Santiago de Compostela, Spain.

John Ortega, Felipe Sánchez-Martínez, and Mikel Forcada. 2016. Fuzzy-match repair using black-box machine translation systems: what can be expected? In *Conferences of the Association for Machine Translation in the Americas: MT Researchers' Track*, pages 27–39, Austin, TX, USA. The Association for Machine Translation in the Americas.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3956–3965. PMLR.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Santanu Pal, Nico Herbig, Antonio Krüger, and Josef van Genabith. 2018. A transformer-based multi-source automatic post-editing system. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 827–835, Belgium, Brussels. Association for Computational Linguistics.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 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. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Mike Paterson and Vlado Dančík. 1994. Longest common subsequences. In *Mathematical Foundations of Computer Science 1994*, pages 127–142, Berlin, Heidelberg. Springer Berlin Heidelberg.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. SemEval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 774–790, Barcelona (online). International Committee for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Minh Quang Pham, Jitao Xu, Josep Crego, François Yvon, and Jean Senellart. 2020. Priming neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 462–473, Online. Association for Computational Linguistics.

MinhQuang Pham, Josep Crego, Jean Senellart, and François Yvon. 2018. Fixing translation divergences in parallel corpora for neural MT. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2967–2973, Brussels, Belgium. Association for Computational Linguistics.

MinhQuang Pham, Josep Maria Crego, and François Yvon. 2021. Revisiting multi-domain machine translation. *Transactions of the Association for Computational Linguistics*, 9:17–35.

Stelios Piperidis, Iason Demiros, Prokopis Prokopidis, Peter Vanroose, Anja Hoethker, Walter Daelemans, Elsa Sklavounou, Manos Konstantinou, and Yannis Karavidas. 2004. Multimodal, multilingual resources in the subtitling process. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Jordan B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46(1):77–105.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en español: toward a typology of code-switching 1. *Linguistics*, 18:581–618.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. 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 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. 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 1993–2003, Online. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13727–13735.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 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.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online. Association for Computational Linguistics.

Younes Samih, Suraj Maharjan, Mohammed Attia, Laura Kallmeyer, and Thamar Solorio. 2016. Multilingual code-switching identification via LSTM recurrent neural networks. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 50–59, Austin, Texas. Association for Computational Linguistics.

Sebastin Santy, Sandipan Dandapat, Monojit Choudhury, and Kalika Bali. 2019. INMT: Interactive neural machine translation prediction. 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): System Demonstrations*, pages 103–108, Hong Kong, China. Association for Computational Linguistics.

Danielle Saunders. 2021. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *CoRR*, abs/2104.06951.

Lane Schwartz. 2008. Multi-source translation methods. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas: Student Research Workshop*, pages 279–288, Waikiki, USA. Association for Machine Translation in the Americas.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. 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.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Jaehun Shin and Jong-Hyeok Lee. 2018. Multi-encoder transformer network for automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 840–845, Belgium, Brussels. Association for Computational Linguistics.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8846–8853.

Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007. Rule-based translation with statistical phrase-based post-editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 203–206, Prague, Czech Republic. Association for Computational Linguistics.

Sunayana Sitaram, Khyathi Raghavi Chandu, Sai Krishna Rallabandi, and Alan W. Black. 2019. A survey of code-switched speech and language processing. *CoRR*, abs/1904.00784.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 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, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Kai Song, Yue Zhang, Heng Yu, Weihua Luo, Kun Wang, and Min Zhang. 2019. Code-switching for enhancing NMT with pre-specified 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 449–459, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhenqiao Song, Hao Zhou, Lihua Qian, Jingjing Xu, Shanbo Cheng, Mingxuan Wang, and Lei Li. 2022. switch-GLAT: Multilingual parallel machine translation via code-switch decoder. In *International Conference on Learning Representations*.

Matthias Sperber and Matthias Paulik. 2020. Speech translation and the end-to-end promise: Taking stock of where we are. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7409–7421, Online. Association for Computational Linguistics.

Matthias Sperber, Hendra Setiawan, Christian Gollan, Udhyakumar Nallasamy, and Matthias Paulik. 2020. Consistent transcription and translation of speech. *Transactions of the Association for Computational Linguistics*, 8:695–709.

Felix Stahlberg. 2019. Neural machine translation: A review. *CoRR*, abs/1912.02047.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985. PMLR.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jinsong Su, Xiangwen Zhang, Qian Lin, Yue Qin, Junfeng Yao, and Yang Liu. 2019. Exploiting reverse target-side contexts for neural machine translation via asynchronous bidirectional decoding. *Artificial Intelligence*, 277:103168.

Yixuan Su, Deng Cai, Yan Wang, David Vandyke, Simon Baker, Piji Li, and Nigel Collier. 2021. Non-autoregressive text generation with pre-trained language models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 234–243, Online. Association for Computational Linguistics.

Raymond Hendy Susanto, Shamil Chollampatt, and Liling Tan. 2020. Lexically constrained neural machine translation with Levenshtein transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3536–3543, Online. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Pilar Sánchez-Gijón, Joss Moorkens, and Andy Way. 2019. Post-editing neural machine translation versus translation memory segments. *Machine Translation*, 33(1):31–59.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021. From machine translation to code-switching: Generating high-quality code-switched text. 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 3154–3169, Online. Association for Computational Linguistics.

Amirhossein Tebbifakhr, Ruchit Agrawal, Matteo Negri, and Marco Turchi. 2018. Multi-source transformer with combined losses for automatic post editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 846–852, Belgium, Brussels. Association for Computational Linguistics.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey. European Language Resources Association (ELRA).

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017*

Conference on Empirical Methods in Natural Language Processing, pages 1400–1410, Copenhagen, Denmark. Association for Computational Linguistics.

Maarten van Gompel, Iris Hendrickx, Antal van den Bosch, Els Lefever, and Véronique Hoste. 2014. SemEval 2014 task 5 - L2 writing assistant. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 36–44, Dublin, Ireland. Association for Computational Linguistics.

Tom Vanallemeersch and Vincent Vandeghinste. 2015. Assessing linguistically aware fuzzy matching in translation memories. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, pages 153–160, Antalya, Turkey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Sriram Venkatapathy and Shachar Mirkin. 2012. An SMT-driven authoring tool. In *Proceedings of COLING 2012: Demonstration Papers*, pages 459–466, Mumbai, India. The COLING 2012 Organizing Committee.

David Wan, Chris Kedzie, Faisal Ladhak, Marine Carpuat, and Kathleen McKeown. 2020. Incorporating terminology constraints in automatic post-editing. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1193–1204, Online. Association for Computational Linguistics.

Chunqi Wang, Ji Zhang, and Haiqing Chen. 2018a. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488, Brussels, Belgium. Association for Computational Linguistics.

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2013. Integrating translation memory into phrase-based machine translation during decoding. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–21, Sofia, Bulgaria. Association for Computational Linguistics.

Qian Wang, Jiajun Zhang, Lemao Liu, Guoping Huang, and Chengqing Zong. 2020. Touch editing: A flexible one-time interaction approach for 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 1–11, Suzhou, China. Association for Computational Linguistics.

Rui Wang, Xu Tan, Renqian Luo, Tao Qin, and Tie-Yan Liu. 2021. A survey on low-resource neural machine translation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4636–4643. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018b. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Yining Wang, Jiajun Zhang, Long Zhou, Yuchen Liu, and Chengqing Zong. 2019. Synchronously generating two languages with interactive decoding. 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 3350–3355, Hong Kong, China. Association for Computational Linguistics.

Taro Watanabe and Eiichiro Sumita. 2002. Bidirectional decoding for statistical machine translation. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2020. Acquiring knowledge from pre-trained model to neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9266–9273.

Gian Wiher, Clara Meister, and Ryan Cotterell. 2022. On decoding strategies for neural text generators. *CoRR*, abs/2203.15721.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019. Code-switched language models using neural based synthetic data from parallel sentences. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 271–280, Hong Kong, China. Association for Computational Linguistics.

Guillaume Wisniewski, Nicolas Pécheux, and François Yvon. 2015. Why predicting post-edition is so hard? failure analysis of LIMSI submission to the APE shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 222–227, Lisbon, Portugal. Association for Computational Linguistics.

Mark Wolfersberger. 2003. L1 to L2 writing process and strategy transfer: A look at lower proficiency writers. *TESL-EJ*, 7(2):1–12.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Joern Wuebker, Spence Green, John DeNero, Saša Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Berlin, Germany. Association for Computational Linguistics.

Mengzhou Xia, Guoping Huang, Lemao Liu, and Shuming Shi. 2019. Graph based translation memory for neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7297–7304.

Yanling Xiao, Lemao Liu, Guoping Huang, Qu Cui, Shujian Huang, Shuming Shi, and Jiajun Chen. 2022a. BiTIIMT: A bilingual text-infilling method for interactive machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1958–1969, Dublin, Ireland. Association for Computational Linguistics.

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-yan Liu. 2022b. A survey on non-autoregressive generation for neural machine translation and beyond. *CoRR*, abs/2204.09269.

Jitao Xu, François Buet, Josep Crego, Elise Bertin-Lemée, and François Yvon. 2022a. Joint generation of captions and subtitles with dual decoding. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 74–82, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Jitao Xu, Josep Crego, and Jean Senellart. 2020. Boosting neural machine translation with similar translations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1580–1590, Online. Association for Computational Linguistics.

Jitao Xu, Josep Crego, and François Yvon. 2022b. Bilingual synchronization: Restoring translational relationships with editing operations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8016–8030, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jitao Xu, Josep Crego, and François Yvon. 2022c. Non-autoregressive machine translation with translation memories. *CoRR*, abs/2210.06020.

Jitao Xu, Minh Quang Pham, Sadaf Abdul Rauf, and François Yvon. 2021a. LISN @ WMT 2021. In *Proceedings of the Sixth Conference on Machine Translation*, pages 232–242, Online. Association for Computational Linguistics.

Jitao Xu and François Yvon. 2021a. Can you traducir this? Machine translation for code-switched input. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 84–94, Online. Association for Computational Linguistics.

Jitao Xu and François Yvon. 2021b. One source, two targets: Challenges and rewards of dual decoding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8533–8546, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weijia Xu and Marine Carpuat. 2021a. EDITOR: An Edit-Based Transformer with Repositioning for Neural Machine Translation with Soft Lexical Constraints. *Transactions of the Association for Computational Linguistics*, 9:311–328.

Weijia Xu and Marine Carpuat. 2021b. Rule-based morphological inflection improves neural terminology translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5902–5914, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weijia Xu, Shuming Ma, Dongdong Zhang, and Marine Carpuat. 2021b. How does distilled data complexity impact the quality and confidence of non-autoregressive machine translation? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4392–4400, Online. Association for Computational Linguistics.

Weijia Xu, Yuwei Yin, Shuming Ma, Dongdong Zhang, and Haoyang Huang. 2021c. Improving multilingual neural machine translation with auxiliary

source languages. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3029–3041, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Masaru Yamada. 2011. The effect of translation memory databases on productivity. *Translation research projects*, 3:63–73.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Weinan Zhang, Yong Yu, and Lei Li. 2020a. Towards making the most of bert in neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9378–9385.

Zhen Yang, Bojie Hu, Ambyera Han, Shen Huang, and Qi Ju. 2020b. CSP: Code-switching pre-training for neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2624–2636, Online. Association for Computational Linguistics.

Zhen Yang, Fandong Meng, Yingxue Zhang, Ernan Li, and Jie Zhou. 2022. WeTS: A benchmark for translation suggestion. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5278–5290, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Anna Zaretskaya, Gloria Corpas Pastor, and Miriam Seghiri. 2015. Integration of machine translation in cat tools: State of the art, evaluation and user attitudes. *Skase Journal of Translation and Interpretation*, 8(1):76–89.

Chun Zeng, Jiangjie Chen, Tianyi Zhuang, Rui Xu, Hao Yang, Qin Ying, Shimin Tao, and Yanghua Xiao. 2022. Neighbors are not strangers: Improving non-autoregressive translation under low-frequency lexical constraints. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5777–5790, Seattle, United States. Association for Computational Linguistics.

Jiajun Zhang, Long Zhou, Yang Zhao, and Chengqing Zong. 2020a. Synchronous bidirectional inference for neural sequence generation. *Artificial Intelligence*, 281:103234.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018a. Guiding neural machine translation with retrieved translation pieces. 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 1325–1335, New Orleans, Louisiana. Association for Computational Linguistics.

Longtu Zhang and Mamoru Komachi. 2018. Neural machine translation of logographic language using sub-character level information. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 17–25, Brussels, Belgium. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, Florence, Italy. Association for Computational Linguistics.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018b. Asynchronous bidirectional decoding for neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5698–5705. AAAI Press.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 368–374, Online. Association for Computational Linguistics.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.

Ran Zmigrod, Sabrina J. Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1651–1661, Florence, Italy. Association for Computational Linguistics.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California. Association for Computational Linguistics.

A - More Details for Extracting the Minimal Alignment Units (Chapter 5)

The minimal alignment units are extracted from bidirectional word-to-word alignments generated by any alignment procedure, like `fast_align` (Dyer et al., 2013). Extracted units should follow the constraints (Crego et al., 2004): no word inside a unit is aligned to words outside this unit, and no smaller units can be extracted without violating the previous constraint. The extraction is decomposed into two steps:

- Firstly, in an iterative procedure, words on one side are grouped when they are linked to the same word or group on the other side. The procedure continues to group words on both sides until no new groups can be obtained.
- The second step outputs the resulting groups (units), keeping the word order of target sentence words. Though, the sequence of units may modify the source sentence word order.

The `fast_align` model proposed by Dyer et al. (2013) is a reparameterization of the IBM Model 2 (Brown et al., 1993) using lexical translation. The latter works as follows. Given a source sentence $\mathbf{f}$ of length S, it first predicts the length of target sentence T, then generates an alignment $\mathbf{a} = a_1, a_2, \ldots, a_T$, which indicates which source word or null token each target word is translated from. Finally, it generates the T target words, where each target word e_i depends only on the aligned source word f_{a_i}.

The alignment and target word prediction follow the distributions:

$$\begin{aligned} a_i \mid i, T, S &\sim \delta(\cdot \mid i, T, S) \quad 1 \leq i \leq T \\ e_i \mid a_i, f_{a_i} &\sim \theta(\cdot \mid f_{a_i}) \qquad 1 \leq i \leq T, \end{aligned} \tag{A.1}$$

where $\theta(\cdot \mid f_{a_i})$ is the lexical translation probabilities, and $\delta(\cdot \mid i, T, S)$ is computed as:

$$\begin{aligned} \delta(a_i = j \mid i, T, S) &= \left\{ \begin{array}{ll} p_0 & j = 0 \\ (1 - p_0) \times \frac{\exp(\lambda h(i,j,T,S))}{Z_\lambda(i,T,S)} & 0 < j \leq S \\ 0 & \text{otherwise} \end{array} \right\} \\ h(i, j, T, S) &= -\left| \frac{i}{T} - \frac{j}{S} \right| \\ Z_\lambda(i, T, S) &= \sum_{j'=1}^{S} \exp \lambda h(i, j', T, S), \end{aligned} \tag{A.2}$$

where p_0 is a null alignment probability, and $\lambda \geq 0$ is a precision that controls the trend diagonal alignment. When λ is larger, the model is more likely to

generate diagonal alignment. The probability that the word at position i of $\mathbf{e}$ is e_i can be computed as:

$$\begin{aligned} P(e_i, a_i \mid \mathbf{f}, T, S) &= \delta(a_i \mid i, T, S) \times \theta(e_i \mid f_{a_i}) \\ P(e_i \mid \mathbf{f}, T, S) &= \sum_{j=0}^{S} P(e_i, a_i = j \mid \mathbf{f}, T, S). \end{aligned} \tag{A.3}$$

The posterior probability over alignments is thus computed as:

$$P(a_i \mid e_i, \mathbf{f}, T, S) = \frac{P(e_i, a_i \mid \mathbf{f}, T, S)}{P(e_i \mid \mathbf{f}, T, S)}. \tag{A.4}$$

The model assumes that all tokens in the target sentence are conditionally independent of each other. Therefore, the translation probability is measured as:

$$P(\mathbf{f} \mid \mathbf{e}) = \prod_{i=1}^{T} P(e_i \mid \mathbf{f}, T, S). \tag{A.5}$$

The `fast_align` model is optimized using the Expectation-Maximization algorithm. In the E-step, the posterior probabilities over alignments are computed as Equation (A.4). The lexical translation probabilities θ are updated in the M-step. The parameter λ is also updated during the M-step.

B - More Results on IWSLT tst2015 for Multi-target Translation (Chapter 6)

Tables B.1 and B.2 reports results for the multi-target translation experiments of Section 6.3 using the IWSLT tst2015 test set, a setting also used in (He et al., 2021a).

Model	De-En	De-Fr	En-De	En-Fr	En-Zh	En-Ja	Avg BLEU
base	33.0	26.2	28.8	38.1	28.4	13.7	28.0
multi	31.6	26.0	27.8	35.7	31.5	11.6	27.4 (-0.6)
indep	34.3	26.8	29.8	38.7	29.1	14.3	28.8 (+0.8)
dual	32.0	22.1	29.4	37.2	28.9	14.2	27.3 (-0.7)
indep ps	33.5	26.7	29.1	38.7	28.8	13.8	28.4 (+0.4)
dual ps	33.0	26.5	29.3	37.9	28.7	14.3	28.3 (+0.3)
indep FT	37.0	29.5	32.1	42.0	32.0	16.5	31.5 (+3.5)
dual FT	36.8	28.4	31.8	41.0	32.6	16.5	31.2 (+3.2)
indep FT+ps	36.4	29.1	31.4	40.9	31.8	16.0	30.9 (+2.9)
dual FT+ps	36.3	29.2	31.8	40.9	32.1	16.0	31.1 (+3.1)

Table B.1: BLEU and scores for multi-target models on tst2015. Pseudo (ps) refers to models trained from scratch with partial synthetic reference data. FT indicates models fine-tuned from the pre-trained multilingual (multi) models. FT+ps refers to models fine-tuned using partial synthetic reference data.

Model	De→En/Fr	En→De/Fr	En→Zh/Ja	Avg SIM
base	90.09	91.77	81.99	87.95
multi	92.24	91.99	83.60	89.28 (+1.33)
indep	90.78	92.00	83.22	88.67 (+0.72)
dual	91.13	91.75	84.24	89.04 (+1.09)
indep ps	91.05	92.52	83.37	88.98 (+1.03)
dual ps	91.55	92.49	83.80	89.28 (+1.33)
indep FT	91.98	92.53	84.74	89.75 (+1.80)
dual FT	91.87	92.80	85.01	89.89 (+1.94)
indep FT+ps	92.47	92.97	84.64	90.03 (+2.08)
dual FT+ps	92.56	92.91	84.62	90.03 (+2.08)

Table B.2: Similarity scores (SIM) for multi-target models on tst2015. The similarity scores are computed as the cross-lingual similarity between sentence embeddings of the two target translations computed with LASER. Pseudo (ps) refers to models trained from scratch with partial synthetic reference data. FT indicates models fine-tuned from the pre-trained multilingual (multi) models. FT+ps refers to models fine-tuned using partial synthetic reference data.

C - More Details for Multilingual Subtitling (Chapter 6)

C.1 . Data Processing Details

For the English to French language pair, MuST-Cinema (Karakanta et al., 2020b) contains 275k sentences for training and 1,079 and 544 lines for development and testing, respectively. The automatic speech recognition (ASR) system used by Karakanta et al. (2020a) to produce transcripts was based on the KALDI toolkit (Povey et al., 2011) and had been trained on the clean portion of LibriSpeech (Panayotov et al., 2015) (~460h) and a subset of MuST-Cinema (~450h). In order to emulate a real production scenario, we segment these transcripts as if they were from an ASR system performing segmentation based on prosody. As this kind of system tends to produce longer sequences compared to typical written text (Cho et al., 2012), we randomly concatenate the English captions into longer sequences, to which we align the ASR transcripts using the conventional edit distance, thus adding a sub-segmentation aspect to the translation task. Edit distance computations are based on a Weighted Finite-State Transducer (WSFT), implemented with Pynini (Gorman, 2016), which represents editing operations (match, insertion, deletion, replacement) at the character level, with weights depending on the characters and the previous operation context. After composing the edit WFST with the transcript string and the caption string, the optimal operation sequence is computed using a shortest-distance algorithm (Mohri, 2002). The number of sentences to be concatenated is sampled normally, with an average of around 2. This process results in 133k, 499, and 255 lines for training, development, and testing, respectively. The approach to concatenate examples is proposed and implemented by François Buet.

For pre-training, we use all available WMT14 En-Fr data,[1] in which we discard sentence pairs with invalid language labels as computed by `fasttext` language identification model[2] (Bojanowski et al., 2017). This pre-training data contains 33.9M sentence pairs.

C.2 . Consistency Score

We borrow the following example from Karakanta et al. (2021) to illustrate the lexical and structural consistency scores:

[1] https://statmt.org/wmt14

[2] https://dl.fbaipublicfiles.com/fasttext/supervised-models/lid.176.bin

0:00:50,820, 00:00:53,820
To put the assumptions very clearly:
Enonçons clairement nos hypothèses : le capitalisme,

00:00:53,820, 00:00:57,820
capitalism, after 150 years, has become acceptable,
après 150 ans, est devenu acceptable, au même titre

00:00:58,820, 00:01:00,820
and so has democracy.
que la democratie.

As defined by Karakanta et al. (2021), for structural consistency, both captions (C) and subtitles (S) have the same number of 3 blocks. For lexical consistency, there are 6 tokens of the subtitles which are not aligned to captions in the same block: *"le capitalisme ," , "au même titre"*. The $Lex_{C \to S}$ is calculated as the percentage of aligned words normalized by the number of words in the caption. Therefore, $Lex_{C \to S} = \frac{20}{22} = 90.9\%$; the computation is identical in the other direction, yielding $Lex_{S \to C} = \frac{17}{23} = 73.9\%$, the average lexical consistency of this segment is thus $Lex_{pair} = \frac{Lex_{C \to S} + Lex_{S \to C}}{2} = 82.4\%$.

When computing the *lexical consistency* between captions and subtitles, we use the WMT14 En-Fr data as used in Chapter 5 to train an alignment model using `fast_align`[3] (Dyer et al., 2013) in both directions and use it to predict word alignments for model outputs. The model is the same as used in Section 5.3.

C.3 . Additional Metrics

Table C.1 reports TER and BERTScores[4] (Zhang et al., 2020b). TER is computed using SacreBLEU.[5] Note that for BERTScores, we remove segmentation tokens (`[eob]` and `[eol]`) from hypotheses and references, as special tokens are out-of-vocabulary for pre-trained BERT models.[6]

[3] https://github.com/clab/fast_align

[4] https://github.com/Tiiiger/bert_score

[5] https://github.com/mjpost/sacrebleu

[6] Signature of BERTScore (En): microsoft/deberta-xlarge-mnli_L40_no-idf_version=0.3.11(hug_trans=4.10.3)-rescaled_fast-tokenizer. Signature of BERTScore (Fr): bert-base-multilingual-cased_L9_no-idf_version=0.3.11(hug_trans=4.10.3)-rescaled_fast-tokenizer.

Model	TER ↓			BERTScore-F1 ↑		
	En	Fr	Avg	En	Fr	Avg
`base`	0.264	0.662	0.463	0.7346	0.3961	0.5654
`base` +FT	0.264	0.654	0.459	0.7346	0.4026	0.5686
`pipeline`	0.264	0.650	0.457	0.7346	0.3912	0.5629
`pipeline` +FT	0.264	0.652	0.458	0.7346	0.3924	0.5635
`dual` +FT	**0.256**	**0.640**	**0.448**	0.7378	**0.4074**	0.5726
`share` +FT	0.259	**0.640**	0.450	**0.7396**	0.4066	**0.5731**

Table C.1: TER and BERTScore for captions (En) and subtitles (Fr). The `base` and `pipeline` settings are trained from scratch with original data. +FT indicates models fine-tuned from the pre-trained translation model. `share` refers to tying all decoder parameters.

D - Detailed Results on Each Domain for Non-autoregressive Translation with Translation Memories (Chapter 8)

BLEU and COMET scores for each domain are in Tables D.1, D.2, D.3, D.4, D.5, and D.6. The variation in scores across domains is large, confirming that TM matches can be very beneficial for some technical domains (*e.g.*, ECB, EME, GNO, KDE, JRC), for which we often find good matches that help to greatly increase the performance. On the other hand, News, Wiki, and TED yield fewer matches, which only helps when the similarity is high ($\mathrm{FM} \geq 0.6$).

BLEU	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	59.8	64.5	34.4	70.3	67.6	55.3	12.0	38.6	30.8	51.6	47.4	52.6
FM	58.7	53.8	55.8	55.0	68.8	53.9	27.1	18.2	62.0	54.0	65.0	51.2
LevT-FM	46.6	30.7	51.8	51.0	62.3	47.0	23.6	12.5	58.7	50.0	61.9	46.5
TM-LevT	53.0	49.7	53.2	51.5	64.7	50.8	24.5	37.1	59.5	50.4	64.0	52.6

Table D.1: BLEU scores for each domain, the task is **standard MT** with $\mathrm{FM} \geq \mathbf{0.6}$. *All* is computed by concatenating all test sets (11k sentences in total). *Copy* refers to copying the TM match into the output.

COMET	copy	FM	LevT-FM	TM-LevT
ECB	0.4006	0.6333	0.4251	0.5637
EME	0.4625	0.6402	0.1322	0.5559
Epp	-0.0797	0.8137	0.7460	0.7513
GNO	0.4893	0.7190	0.6181	0.6355
JRC	0.6893	0.9057	0.8291	0.8477
KDE	0.1150	0.5116	0.3879	0.4218
News	-0.6083	0.3241	0.2037	0.1660
PHP	-0.1977	-0.0556	-0.6139	-0.0980
TED	-0.4184	0.7848	0.6912	0.6929
Ubu	0.3296	0.7031	0.5636	0.5768
Wiki	0.2843	0.7786	0.6947	0.7335
All	0.1330	0.6143	0.4251	0.5314

Table D.2: COMET scores for each domain, the task is **standard MT** with $\mathrm{FM} \geq \mathbf{0.6}$.

BLEU	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	47.3	47.6	12.7	52.6	53.0	42.7	5.8	29.7	8.2	35.1	13.0	34.5
FM	52.3	52.7	44.7	54.4	64.7	53.2	30.0	17.9	41.7	49.2	42.2	46.1
LevT-FM	40.7	31.4	42.6	51.0	59.8	46.8	27.6	11.9	38.7	45.7	40.2	40.8
TM-LevT	47.9	47.7	41.5	51.6	61.1	50.1	26.8	34.3	38.0	46.8	41.0	45.7

Table D.3: BLEU scores for each domain, the task is **standard MT** with $\mathrm{FM} \in [\mathbf{0.4}, \mathbf{0.6})$.

COMET	copy	FM	LevT-FM	TM-LevT
ECB	0.0310	0.5229	0.2908	0.4370
EME	0.1527	0.5920	0.1245	0.5231
Epp	-0.7608	0.7735	0.6996	0.6515
GNO	0.1416	0.7048	0.5956	0.6205
JRC	0.1919	0.8834	0.8069	0.8116
KDE	-0.1703	0.5522	0.4140	0.4576
News	-0.9719	0.4688	0.3567	0.2948
PHP	-0.6279	-0.1819	-0.7332	-0.2343
TED	-1.1419	0.5501	0.3979	0.3655
Ubu	-0.1837	0.6363	0.5194	0.5035
Wiki	-0.8222	0.4157	0.3011	0.2600
All	-0.3784	0.5379	0.3429	0.4263

Table D.4: COMET scores for each domain, the task is **standard MT** with $\mathrm{FM} \in [\mathbf{0.4}, \mathbf{0.6})$.

BLEU	ECB	EME	Epp	GNO	JRC	KDE	News	PHP	TED	Ubu	Wiki	All
copy	47.3	47.6	12.7	52.6	53.0	42.7	5.8	29.7	8.2	35.1	13.0	34.5
AR	62.3	62.8	44.9	69.6	75.4	62.1	29.9	39.2	42.6	58.1	43.9	55.7
LevT	52.3	47.1	42.7	65.7	71.9	57.6	27.5	23.8	39.0	55.0	40.8	49.3
+tgt TM	47.4	48.0	13.2	53.2	53.5	42.9	6.0	29.7	9.1	37.1	13.2	35.0
TM-LevT	59.7	61.9	41.4	68.1	73.0	61.4	26.4	39.1	37.5	56.1	39.7	53.3

Table D.5: BLEU scores for each domain, the task is **MT with TMs** with $\mathrm{FM} \in [\mathbf{0.4}, \mathbf{0.6})$.

COMET	copy	FM	LevT-FM	LevT-FM + tgt TM	TM-LevT
ECB	0.0310	0.5814	0.4283	0.0487	0.5102
EME	0.1527	0.6607	0.2846	0.1569	0.6281
Epp	-0.7608	0.7740	0.6998	-0.7208	0.6368
GNO	0.1416	0.8380	0.7697	0.1883	0.8142
JRC	0.1919	0.9220	0.8746	0.2167	0.8741
KDE	-0.1703	0.6217	0.5437	-0.1151	0.5814
News	-0.9719	0.4741	0.3660	-0.9508	0.2781
PHP	-0.6279	-0.1140	-0.4900	-0.6205	-0.1853
TED	-1.1419	0.5543	0.4107	-1.0949	0.3523
Ubu	-0.1837	0.7453	0.6676	-0.1234	0.6727
Wiki	-0.8222	0.4344	0.2910	-0.8100	0.2172
All	-0.3784	0.5900	0.4404	-0.3478	0.4889

Table D.6: COMET scores for each domain, the task is **MT with TMs** with $\mathrm{FM} \in [\mathbf{0.4}, \mathbf{0.6})$.

E - Résumé Français

Dans un monde de plus en plus globalisé, il est de plus en plus courant d'avoir à s'exprimer dans une langue étrangère ou dans plusieurs langues. Cependant, pour de nombreuses personnes, parler ou écrire dans une langue étrangère n'est pas une tâche facile. Les outils de traduction automatique peuvent aider à générer des textes en plusieurs langues. Grâce aux progrès récents de la traduction automatique neuronale (NMT), les technologies de traduction fournissent en effet des traductions utilisables dans un nombre croissant de contextes. Pour autant, il n'est pas encore réaliste d'attendre des systèmes NMT qu'ils produisent des traductions sans erreur. En revanche, les utilisateurs ayant une bonne maîtrise d'une langue étrangère donnée peuvent trouver des aides auprès des technologies de traduction aidée par ordinateur.

Lorsqu'ils rencontrent des difficultés, les utilisateurs écrivant dans une langue étrangère peuvent accéder à des ressources externes telles que des dictionnaires, des terminologies ou des concordanciers bilingues. Cependant, la consultation de ces ressources provoque une interruption du processus de rédaction et déclenche une autre activité cognitive. Pour rendre le processus plus fluide, il est possible d'étendre les systèmes d'aide à la rédaction afin de prendre en charge la composition de textes bilingues. Cependant, les études existantes se sont principalement concentrées sur la génération de textes dans une langue étrangère. Nous suggérons que l'affichage de textes correspondants dans la langue maternelle de l'utilisateur peut également aider les utilisateurs à vérifier les textes composés à partir d'entrées bilingues. Dans cette thèse, nous étudions des techniques pour construire des systèmes d'aide à la rédaction bilingues qui permettent la composition libre dans les deux langues et affichent des textes monolingues synchronisés dans les deux langues. Nous présentons deux types de systèmes interactifs simulés.

La première solution permet aux utilisateurs de composer des textes dans un mélange de langues, qui sont ensuite traduits dans leurs équivalents monolingues. Nous étendons le modèle Transformer pour la traduction en ajoutant un décodeur duel: notre modèle comprend un encodeur partagé et deux décodeurs pour produire simultanément des textes en deux langues. Nous ajoutons une couche d'attention croisée du décodeur dans chaque bloc du décodeur pour construire l'interaction entre les deux décodeurs. En conséquence, les deux décodeurs sont complètement synchronisés et notre modèle de décodeur duel est capable de générer simultanément les traductions dans les deux langues. Nous explorons également le modèle de décodeur duel pour plusieurs autres tâches, telles que la traduction multi-cible, la traduction bidirectionnelle, la génération de variantes de traduction et le sous-titrage multilingue.

La deuxième contribution vise à étendre les systèmes de traduction commerciaux disponibles en ligne en permettant aux utilisateurs d'alterner librement entre les deux langues, en changeant la boîte de saisie du texte à leur volonté. Dans ce scénario, le défi technique consiste à maintenir la synchronisation des deux textes d'entrée tout en tenant compte des entrées des utilisateurs, toujours dans le but de créer deux versions également bonnes du texte. Pour cela, nous introduisons une tâche générale de synchronisation bilingue et nous implémentons et expérimentons des systèmes de synchronisation autorégressifs et non-autorégressifs. Nous proposons un nouveau modèle de traduction non-autorégressif basé sur le modèle Levenshtein Transformer qui peux effectivement éditer une phrase de cible pour rendre le parallélisme des deux langues. Nous étudions également l'utilisation de modèles de synchronisation bilingue pour d'autres tâches spécifiques, telles que le nettoyage de corpus parallèles et la NMT avec mémoire de traduction, afin de mieux évaluer la capacité de généralisation des modèles proposés. Notre travail est en particulier le premier à étudier en profondeur les modèles de traduction non-autorégressifs complétés par des mémoires de traduction.