BACHELOR'S THESIS / ENGINEERING 2023

Neural Based Machine Translation from Catalan Sign Language glosses to written Catalan

Luka Chabaud

upf. Universitat Pompeu Fabra Barcelona

Neural Based Machine Translation from Catalan Sign Language glosses to written Catalan

TREBALL FI DE GRAU DE

Luka Chabaud

Director: Euan McGill

Grau en Enginyeria en Informàtica

Curs 2023-2024

upf. Universitat Pompeu Fabra Barcelona

Escola d'Enginyeria

A la meva família,
els meus amics
i les comunitats sordes de tot el món.

Acknowledgments

Firstly, I want to thank my supervisor, Euan McGill, who did a great job guiding me throughout the work. This wouldn't have been possible without him. I also want to thank my parents and my brother for the constant support they gave me, especially through the last weeks, and my close friends, who for some reason think that I'm way better at this than I do. Lastly, I want to thank my grandmother, who has shown so much interest in what I was doing, even though she had such a hard time understanding what I was doing. It was thanks to all this wonderful people that I was able to get to where I am today.

This work is the culmination of many years of hard work at the Pompeu Fabra University, where I was lucky to find studies I was passionate about. Looking back, it feels like it was yesterday when I attended twice the first class because I didn’t understand how groups and schedules worked. These four years went flying, but they have been an enjoyable experience which I will remember for many years.

I am now again at one of the important milestones of my life, about to start my life as an engineer. May this work serve as proof of this new beginning.

Abstract

This thesis had two objectives: the first one was to experiment with an alternative method to statistical machine translation, which had already been researched for Catalan Sign Language (LSC), using state-of-the-art Neural Networks. More precisely, this work focuses on the yet unexplored translation task from LSC glosses to written Catalan, finding the best possible neural architecture by searching for the best hyperparameter values. The second objective was to apply existing data augmentation techniques for Sign Languages, also developing specific LSC rules, based on its grammar, in order to create a parallel corpus of synthetic data. We then used this augmented data to enhance the training of the models, achieving significant improvements over a baseline without the need of new labeled data.

Resum

Aquesta tesi tenia dos objectius: el primer era experimentar amb un mètode alternatiu a la traducció automàtica estadística, que ja s'havia investigat per a la llengua de signes catalana (LSC), utilitzant Xarxes Neuronals d'última generació. Més precisament, aquest treball es centra en la tasca de traducció encara inexplorada de les gloses LSC al català escrit, trobant la millor arquitectura neuronal possible mitjançant la recerca dels millors valors d'hiperparàmetres. El segon objectiu era aplicar les tècniques d'augment de dades existents per a les llengües de signes, desenvolupant també regles específiques de la LSC, basades en la seva gramàtica, per tal de crear un corpus paral·lel de dades sintètiques. A continuació, vam utilitzar aquestes dades augmentades per millorar la formació dels models, aconseguint millores significatives respecte a una línia de base sense necessitat de noves dades etiquetades.

Resumen

Esta tesis tenía dos objetivos: el primero era experimentar un método alternativo a la traducción automática estadística, que ya había sido investigada para la Lengua de Signos Catalana (LSC), utilizando Redes Neuronales de última generación. Más precisamente, este trabajo se centra en la tarea de traducción aún inexplorada de las glosas de LSC al catalán escrito, encontrando la mejor arquitectura neuronal posible mediante la búsqueda de los mejores valores de hiperparámetros. El segundo objetivo era aplicar técnicas de aumento de datos existentes para lenguas de signos, desarrollando también reglas específicas a la LSC, basadas en su gramática, para crear un corpus paralelo de datos sintéticos. Luego utilizamos estos datos aumentados para mejorar el entrenamiento de los modelos, logrando mejoras significativas con respecto a una línea de base sin la necesidad de nuevos datos etiquetados.

Index

List of figures

List of tables

Chapter 1

Introduction

There are more than 25000 people that use Catalan Sign Language in Catalonia, including among them around 12000 deaf people (Barberà & Quer, 2012). Because of the language barrier among signers of different languages and non-signers, they are often forced to use intrusive and unnatural ways to communicate (Núñez-Marcos et al., 2023). This is where Sign Language Processing comes in, attempting to offer an accurate and non-intrusive communication channel to bypass the language barrier and improve the inclusion of the Deaf and Hard of Hearing community.

1.1 Background

Sign Language Processing (SLP) is composed of multiple sub-fields of research, ranging from Natural Language Processing tasks to Computer Vision tasks. To understand the different fields, we first need to know about the different existing ways to represent Sign Languages. We will provide a quick description of different notations that appear in Yin et al. (2021), from farthest to closest to text language representation as we know it. A comparison of all the different representations can be seen in **Figure 1.1**.

The first representation are **Videos**. Videos are the more straightforward way to capture the meaning of Sign Language. However, it is also the most expensive to work with, as it often captures more information than it needs to.

Next representation are **Poses**. Poses are a simplification of the movements into a human-looking wireframe. This reduces the complexity of the representation and anonymizes it. Poses are usually extracted from videos, but can also be obtained from motion capture systems.

After that come all the **Written Notation Systems**. These systems represent the Sign Languages with the help of symbols. However, none of these notations have seen extended use. You can see an example of some existing notations in Figure 1.1

Finally, we have **Glosses**, the notation that will be used in this research. Glosses are a text transcription of the signs, usually corresponding to the lemmas of the words they represent, making them a suitable target for Natural Language Processing techniques. However, it is important to note that this notation fails to capture all the information conveyed by sign languages. It has difficulties representing non-manual information like the body posture, and it doesn't capture well the spatial relations.

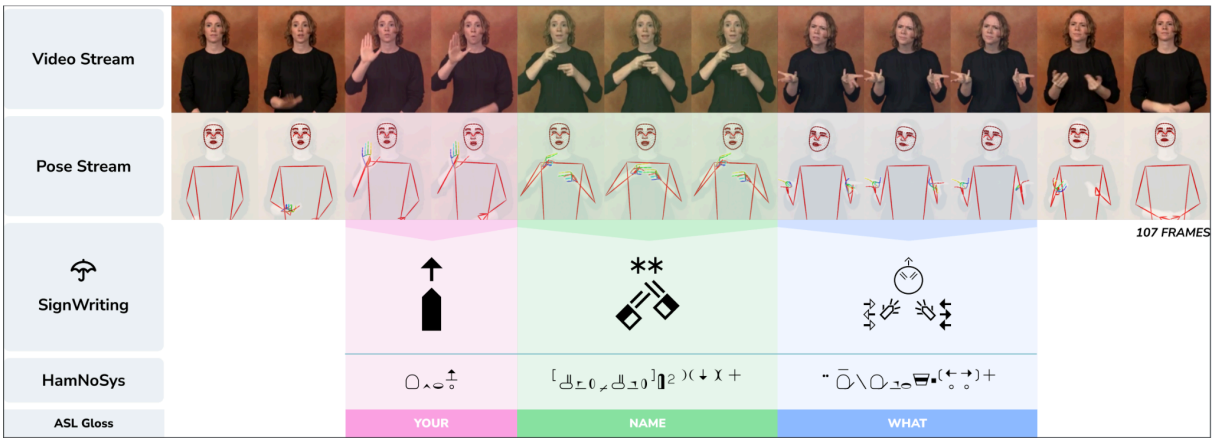

Figure 1.1: English sentence "What is your name?" in different American Sign Language representations (video frames, pose estimations, Written Notations and glosses). Taken from Yin et al. (2021)

The different sub-fields of Sign Language Production are related to the different conversion tasks between language representations, all represented in **Figure 1.2**. They are often grouped depending on the target representation. For example, tasks that go from Glosses or Text representations to Pose or Video are known as Sign Language Production. For a complete definition of all the tasks, see Yin et al. (2021).

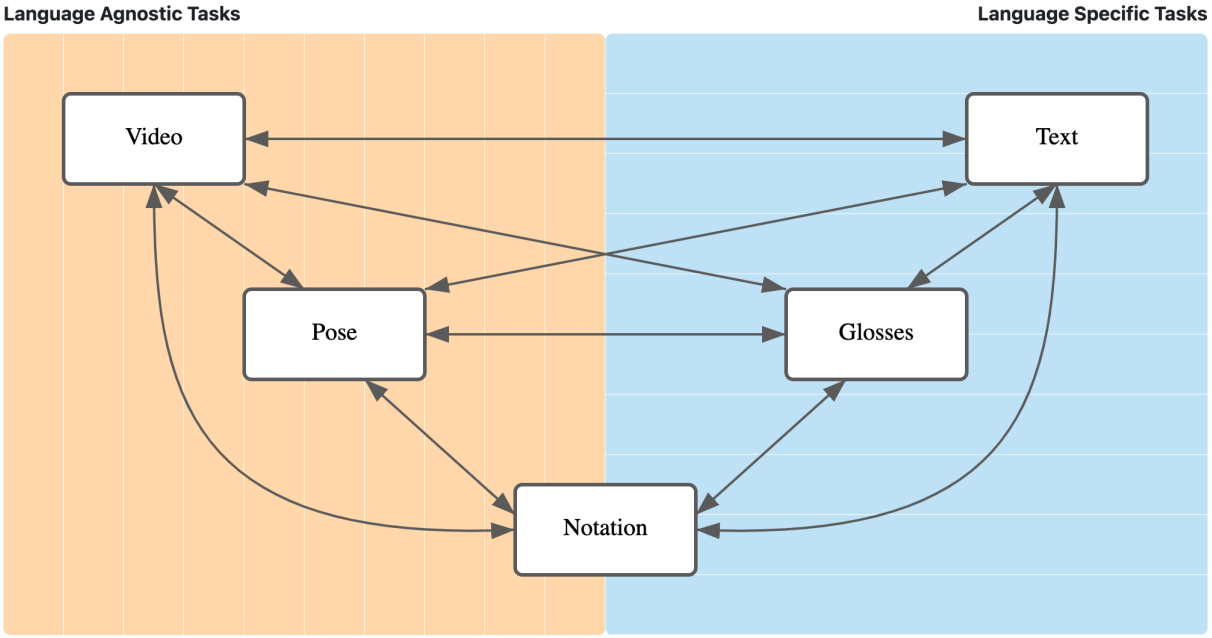

Figure 1.2: Graph representing the different conversion tasks between language representations in Sign Languages. Edges in the orange zone represent Computer Vision tasks, edges in the blue zone represent Natural Language Processing tasks. Edges in both zones require a combination of both. Taken from https://research.sign.mt/

Our research is part of the Sign Language Translation field, and we will specifically focus on the Gloss-to-Text direction for Catalan Sign Language.

1.2 Problem Statement

Sign Languages have their own grammar and rules, not necessarily following the same conventions as their spoken counterparts. Moreover, there is no universally established Sign Language, but more than 200 different SLs, each with its own characteristics[1]. These differences make reusing knowledge gained on other languages not that straightforward.

The amount of data and research available varies from language to language, Catalan Sign Language (LSC) being on the low side for both. The last and only attempt we found at Machine Translation for LSC was done by Sanabre & Cardús (2012), which used a statistical approach and a small corpus collected for this work. This was before the recent rise in popularity of Neural Networks, which are now State-of-the-Art.

There are 2 main reasons for the lack of research on this topic. First, LSC is almost exclusively used in Catalonia, which is small compared to other languages. Second and most important, the amount of data available is extremely limited. The serious collection of annotated data useful for Machine

[1] https://wfdeaf.org/our-work/

Translation has started relatively recently (Barberà et al., 2015), and is unfortunately still not available to researchers at the moment. Nonetheless, in this work, we will attempt to improve the results yielded by the statistical approach, getting the most out of the small parallel corpus that was collected.

1.3 Objectives

As just mentioned, the only documented attempt at Machine Translation for Catalan Sign Language was done by Sanabre & Cardús (2012), on the Text-to-Gloss direction using a statistical method. This was some time ago, and modern techniques are quite different. Thus, the first objective is developing a Neural Machine Translation (NMT) network, this time for the Gloss-to-Text translation task, exploring different configurations to try to find the best one possible. As there are no previous results in this task, we will compare with a simple model that will serve as a baseline.

The other objective is related to the data scarcity problem. NMT works best when there are high quantities of data, and unfortunately it isn't our case. To try to overcome these problems, some techniques can be used to artificially enhance the dataset with synthetic data. This is called Data Augmentation, and it has already been applied in the past using a set of rules to create synthetic glosses (Moryossef et al., 2021). Knowing that, our second objective is to apply these already existing methods and create our own set of rules, attempting to improve the results.

Chapter 2

State of the Art

As mentioned in the previous chapter, the focus of our work will be the translation of LSC glosses to written Catalan. Before defining our research methods, it is important to know what has been done in this field. In this chapter, we will look at some of the already existing literature on Gloss-to-Text translation, and we will also mention some Text-to-Gloss approaches, as they are usually researched together. We will also mention some important projects related to the documentation of the Catalan Sign Language.

2.1 Gloss-to-Text and Text-to-Gloss

In this section, we will mention three different types of approaches in Machine Translation. The first are Statistical Approaches. As seen in the previous chapter, the work done by Sanabre & Cardús (2012) falls into this category. They used the Moses toolkit (Koehn et al., 2007) for the Text-to-Gloss task, improving the results by adding to the annotated data buccal adverbs as well as manual morphemes, topic and punctuation marks.

Then, we also have Rule Based approaches, like the one followed by Porta et al. (2014), where they use rules to transfer the analyzed Spanish dependencies into another dependency tree in Spanish Sign Language (LSE), later used to generate glosses.

Current State-of-the-Art approaches, however, use Neural Machine Translation to achieve the highest quality results. Camgöz et al. (2018) tried different types of Recursive Neural Networks, which would later be outperformed by Yin et al. (2020), whose architecture used Transformers (Vaswani et al., 2017). To combat data scarcity, Moryossef et al. (2021) proposed a set of rules to create augmented data from a monolingual corpus, showing that pretraining the models on this artificial data improved the results on the Gloss-to-Text direction. All the results of the previously mentioned works can be compared between them, as the experiments are performed in the RWTH-PHOENIX-Weather dataset (Forster et al., 2012). Chiruzzo et al. (2022), following the same idea, proposed their own set of rules for LSE, showing improvements on both Text-to-Gloss and Gloss-to-Text translation tasks. These rules were recently expanded upon by Perea-Trigo et al. (2024), with the aim of creating whole synthetic datasets. Other approaches linguistically enhance the inputs by adding Part-of-Speech (POS), dependency and morphological information to the input and using transfer learning for the Text-to-Gloss direction (Egea Gómez et al., 2022). On the Gloss-to-Text direction, adding POS information has also been attempted, by training a POS tagger for LSE glosses, which improved the quality of the translations.

2.2 Catalan Sign Language

We will start this section by mentioning again the work made by Sanabre & Cardús (2012), which collected a corpus of gloss annotated weather predictions. This is the data that we will use in this research. A more extensive corpus is currently being created, as described by Barberà et al. (2015), to document the current state of Catalan Sign Language and be used for different kind of research, like for example the analysis of the grammar and lexicon of LSC or the creation of dictionaries. All advances are being recorded on the *Portal de la llengua de signes catalana*[2] website.

[2] https://lsc.iec.cat/inici/

During our research, we will also use the Catalan Sign Language Grammar (Quer & Barberà, 2020) documented in the SignHub platform[3]. It is still a work in progress, but it will be useful to understand the language and possibly create some LSC specific augmentation rules to improve the quality of our translations.

[3] https://thesignhub.eu/

Chapter 3

Methods

In this chapter, we will define the experiments that will be attempted to achieve the objectives previously defined in **Chapter 1.3**. The followed steps and the taken decisions will be discussed here, and the results of said experiments will be seen later in **Chapter 5** before being commented on in **Chapter 6**.

3.1 Preprocessing and Analyzing the Data

The first step of our experiments is understanding the data we are working with. As mentioned in **Chapter 2**, we will use the LSC parallel corpus collected previously by Sanabre & Cardús (2012), as at the moment, the LSC Corpus Project database isn't available. The data was collected from the *Servei Meteorològic de Catalunya* website[4] and annotated by two native signers. The data was then adapted into a format suitable for the Moses Statistical Machine Translation toolkit (Koehn et al., 2007) and some morphological tags were added. The resulting lines look like this:

zona|zona augmentar-nivell/c|augmentar-nivell .|.

pujant|pujar|vg|vg---- clarament|clarament|d4|d4 arreu|arreu|d4|d4 .|.|x|x

The top sentence is the one annotated in LSC glosses, the source language of our translation system. Each space separated token has two different parts, delimited by "|". The first one is the gloss with morphemes after a "/", like in this example the *augmentar-nivell/c* gloss, with added morpheme *c*. The second part of the token is the same gloss but without any morphemes. For our experiments, the second part of the tokens was used, mainly for two reasons. First, no mapping between the morpheme tag (in this example *c*) and its related morpheme was found in its original paper, as the documented examples are slightly different to ours in terms of the notation of the morphemes. Second, not using the morphemes would simplify the data augmentation attempts that will later be attempted in Chapter 3.4. So for simplicity reasons, we dropped all morphemes. This also includes topic marks, and similar (represented as "!", "?" and "?!" in our data).

The bottom sentence is the original Catalan sentence, the target language of our translation system. Each space separated token has now four parts separated by "|". They are, respectively, the original word, its lemmatized form, the major morphological tag and the complete morphological tag. As the target, we will simply take the original Catalan sentence.

After extracting the sentences, they would be left with this:

zona augmentar-nivell .

pujant clarament arreu .

The resulting sentences are then written into two parallel files, one for LSC and one for spoken Catalan. Each line in the LSC file has its translation in the same line in the spoken Catalan file, hence

[4] www.meteo.cat

the *parallel* name. The scripts that were used to extract the sentences can be found in the GitHub repository[5].

Now that we have the sentences just as we need them, we can further prepare the data. For both the preprocess and the training, we will follow the method provided in the GitHub repository shared by Yasmin Moslem[6].

Sentences that are too low-quality can affect the quality of the translations, so the first step of this further processing will be to filter out these segments. This includes removing empty rows (if there are any), duplicate rows, rows with sentences that have more than 200 words and then shuffling them. After the filtering, we are left with 298 sentences, from the original 410 sentences.

Next, we will separate the data into 3 different datasets, training, development and testing. Due to the limited amount of data that we have, the number of development and testing sentences is less than what we would like. In **Table 3.1** we can see information about the sizes of the different datasets used. In **Table 3.2** we can see which percentage each of the datasets covers of the glosses of other datasets. The coverage of set1 over set2 is computed as

$$coverage = \frac{size(set1 \cap set2)}{size(set2)} \cdot 100$$

	Train	Dev	Test	Total
Sentences	248	25	25	298
Total words	3783	397	376	4556
Unique words	374	143	136	394
Total glosses	2744	296	268	3308
Unique glosses	266	123	106	286

Table 3.1: Sizes of the training, development and test sets

	Train	Dev	Test	Total
Train coverage	100	87.80	92.45	93.01
Dev coverage	40.60	100	61.32	43.01
Test coverage	36.84	52.85	100	37.06

Table 3.2: Percentage of coverage between pairs of sets

The final step is training a sub-wording tokenizer. Word-by-word tokenization is costly in terms of hardware, so the size of the vocabulary is limited. Sub-wording combats this by learning smaller units that are frequently used, called sub-words. This also allows the translation system to also attempt a translation even when finding a new, unknown word. For our experiments, we will use the

[5] https://github.com/LukaChabaud08/LSC-translation
[6] https://github.com/ymoslem/OpenNMT-Tutorial

SentencePiece toolkit (Kudo & Richardson, 2018), more precisely we will train unigram models like the one proposed by Kudo (2018). Once we have trained it, we use them to tokenize our three datasets.

The same processing will be applied to the augmented data that we will create in **Chapter 3.4**, with the difference that the sub-wording model will be trained with the combination of both the original and the synthetic data. Now that we have finally finished preprocessing the data, we can start experimenting with our Neural Machine Translation networks.

3.2 Training the Baseline model

As already discussed in **Chapter 2.2**, Neural Machine Translation is the set of techniques that yields state-of-the-art results at the moment. There exists many frameworks in Python made for implementing neural networks, like PyTorch and TensorFlow, and the one used in these experiments is built on top of both. It is the OpenNMT framework (Klein et al., 2017), which is a toolkit specifically made for Machine Translation tasks. More information can be found in **Chapter 4.1**.

The first translation attempt will be the simplest configuration possible. This is done so that we can compare later models and see if the modifications made in **Chapter 3.3** and **Chapter 3.4** yielded improvements on the translation results or not. The configuration file of the baseline model can be found in **Appendix A**. **Figure 3.1** shows a diagram of the architectures used by OpenNMT.

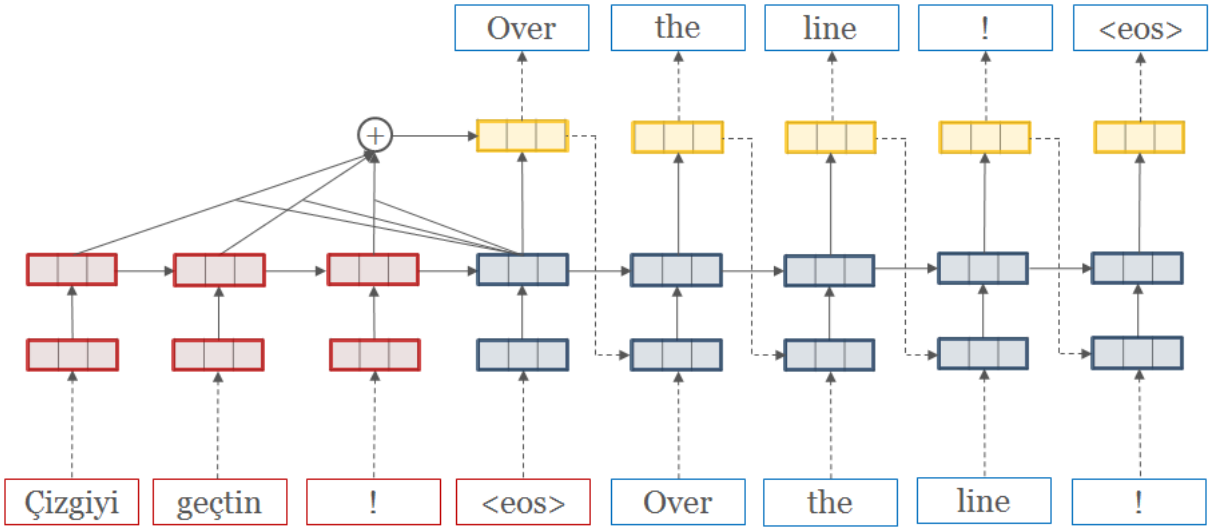

Figure 3.1: Diagram of a Machine Translation model used by OpenNMT. The red words are from the source language, first converted into vectors and then given to the RNN. When finding the <eos> symbol, the target RNN is initialized, applying attention at each time step. It is then combined with the current hidden state to predict the next word. Taken from (Klein et al., 2017).

Early experiments showed that the best results were achieved in the first 4000 epochs, so all following experiments were done with this number of steps at most. You can find these early results in **Chapter 5.1**.

To evaluate the results of the translation, we used the BiLingual Evaluation Understudy metric (Papineni et al., 2002), also known as BLEU. It is the most frequent metric among the research we studied. In our case, we use the SacreBLEU[7] implementation, which offers a standardized, comparable and consistent way to compute the scores (Post, 2018). As an alternative evaluation metric, we also

[7] https://github.com/mjpost/sacrebleu

used the METEOR metric (Banerjee & Lavie, 2005), using the implementation provided in the NLTK[8] library (Bird et al., 2009).

Now that it is well-defined how we will compare the models, we can continue doing further experiments.

3.3 Finding the optimal model

The translation task we are attempting to perform is incredibly low-resource, only having around 300 parallel sentences. Under such conditions, it is even more important to find good hyperparameter values (Zhang et al., 2021). As a starting point, we will use the STMC-Transformer configuration found by Yin & Read (2020), which has also been used in other studies like in Moryossef et al. (2021). This architecture is inspired in the Transformer proposed by Vaswani et al. (2017). The configuration file in can be found in **Appendix B**.

We performed the hyperparameter search by experimenting with different alternative values, usually one above and one below, observing the effects of one hyperparameter at a time. For example, if the *layer's* hyperparameter has a value of 2 in the original configuration, the attempted alternative values could be 1 and 3.

Here in **Table 3.3** you can see which hyperparameters will be examined.

Hyperparameter examined	**Examined values**
Layers	1, 2, 3
Hidden size	256, 512, 1024
Word_vec size	256, 512, 1024
Initial Learning rate	0.1, 0.5, 0.7
Transformer_ff	1024, 2048, 4096

Table 3.3: list of hyperparameters that will be examined with the attempted values. Underlined values are the original values

Another way to improve results is by leveraging word embeddings. Word embeddings are used to represent words in a high-dimensional space, where spatial relationships reflect the similarity between words. OpenNMT-py at the moment only supports word embeddings that are either in GloVe (Pennington et al., 2014) or Word2Vec (Mikolov et al., 2013) format. No publicly available pretrained embeddings could be found in any of these formats, so two alternatives were tried, both using the Gensim library (Řeh uřek et al., 2010). More information about the library can be found in **Chapter 4.4**.

The first alternative we tried was converting the FastText (Bojanowski et al., 2017) embeddings trained by Gutiérrez-Fandiño et al. (2021) into a suitable Word2Vec format. This was done by using the common KeyedVectors structure all the different embedding models can be loaded into. Once the vectors were loaded, they were exported into the desired format and then used on the target language in an OpenNMT model. The results, however, were quite underwhelming, getting much worse results than the baseline model, so no further discussion on this approach will be made.

[8] https://www.nltk.org/

The second alternative, however, is worth mentioning. Again using the Gensim library, we trained our own word embeddings on the data we had available, both for the spoken Catalan and the LSC glosses. These embeddings were then used on the target and source languages, respectively. The scripts used for the conversion and training of the embeddings can be found on the same repository mentioned in **Chapter 3.1**.

After experimenting with all the different variations, a model with all the optimal configuration values was trained and compared to the baseline model. The configuration of said model can be found in **Appendix C**.

3.4 Data Augmentation

To try to combat data scarcity, one of the popular strategies is using data augmentation techniques. Data augmentation consists in the creation of additional artificial data, from other already existing data. This has already been attempted in the past by Moryossef et al. (2021), using a rule-based system to create synthetic gloss data. This data was then used to pretrain a model that would later be fine-tuned on just the real data. The general rules presented in the paper can be found in **Table 3.4**.

Moryossef General Rules

1) Discard all tokens $t \in S$ if $POS(t) \notin \{noun, verb, adjective, adverb, numeral\}$
2) Discard remaining tokens $t \in S$ with probability $p = 0.2$
3) Lemmatize all tokens $t \in S$
4) Apply a random permutation σ to S verifying $\forall i \in \{1, n\}, |\sigma(i) - i| \leq 4$

Table 3.4: Moryossef General Rules (Moryossef et al., 2021). S is the sentence, n is the number of remaining tokens at step 4 and POS is a part-of-speech tagger.

The first data augmentation experiment will be performed using these rules, except for the last one. For the part-of-speech tagging part, we will use the python library SpaCy (Honnibal & Montani, 2017). More information on this library can be found in **Chapter 4.2**.

The synthetic data will be created from three different corpora. The first corpus is the Tatoeba corpus[9], released under a CC-BY 2.0 FR license[10], which we will download from the Opus Corpora website (Tiedemann, 2012). The second corpus we will use is the AnCora-CA corpus (Taulé et al., 2008). The third one will be a small corpus with data collected from the Meteocat weather predictions[11] over a span of 12 days, using their API. This data is similar to the annotated one we have, so we expect to have better results with it than with other corpora, even though the amount of data is not that great. The scripts used can be found in the same repository mentioned in **Chapter 3.1**.

Moryossef et al. (2021) also proposed a set of rules specifically made for the German Sign Language, getting an improvement similar to the general rules over the baseline model. Chiruzzo et al. (2022) attempted the same with Spanish Sign Language (LSE), basing the rules on the LSE grammar and the annotated data they had, and Perea-Trigo et al. (2024) extended this set of rules.

So for our research, we also created a set of rules specific to Catalan Sign Language, with the aim of improving the results yielded by the general augmentation rules. We will base these rules on the

[9] https://tatoeba.org
[10] https://creativecommons.org/licenses/by/2.0/fr/
[11] https://www.meteo.cat/prediccio/general

annotated data we have at our disposition, the rules previously used in other languages and the LSC Grammar (Quer & Barberà, 2020). In **Table 3.5** you can see the different LSC specific augmentation rules that were used in our experiments.

LSC Specific Augmentation Rules
Used General Rules (Moryossef et al., 2021)
1. Remove Determinants 2. Remove Prepositions 3. Lemmatize words
Specific ordering of the glosses, based on LSC Grammar (Quer & Barberà, 2020)
4. Adjectives are signed after the noun they modify 5. Adverbials are signed after the verb they modify 6. Basic order of a sentence is subject-object-verb 7. Quantifiers can go before and after the noun* 8. Temporal adverbials at start of sentence* 9. Frequency adverbials can go anywhere, except negative ones, which go to the end of the sentence*
Corpus Observed Rules
10. dilluns , 19 de novembre de 2007 → després dilluns dia 19 mes novembre any 2007 11. prepirineus → per-sota pirineus 12. oriental → est, occidental → oest 13. Substitute wind names by *vent [direction]* 14. Marejol → mar onades

Table 3.5: Catalan Specific Augmentation Rules

Again, the implementation of these rules can be found in the same repository mentioned in **Chapter 3.1**. Rules marked with a * weren't implemented due to limitations of the Dependency Parser and Part-of-Speech tagger that were used.

Once we have applied the rules on the different datasets, two different strategies were followed. The first one was pretraining the model with a combination of the real and synthetic data, and then fine-tuning on just our real data. The second strategy was very similar to the first one, but instead of using the two datasets, we pretrained the models just on the synthetic data.

The aim of the first strategy was to get the best possible results with data augmentation, seeing also how the model would perform by using an extended dataset with synthetic data. The objective of the

second approach was to explore an alternative to the first one, one that could also show us how training a model with a fully synthetic dataset could perform.

With these final experiments, we conclude the methodology section of the research. In **Chapters 5 & 6**, we will see the results they yielded and discuss what they imply.

Chapter 4

Development: Tools & Technology

Deciding what technologies will be used to achieve the results of our research is a crucial part of the work, as properly choosing the tools can save hours in terms of training times. In this section we will see the frameworks, services and libraries that were used to conduct the experiments, explaining in the process why we chose them.

4.1 OpenNMT

For the translation experiments, we use the open source Neural Machine Translation toolkit *OpenNMT*[12]. This toolkit aims to offer an easy-to-understand framework which allows researchers to develop and compare their neural networks, all while yielding production worthy results both in accuracy and efficiency (Klein et al., 2017).

Currently, *OpenNMT* is implemented in two of the most popular Deep Learning frameworks, *PyTorch* and *TensorFlow*. The one chosen for the experiments was *OpenNMT-py*, as its user-friendliness will make the implementation of the toolkit simpler.

OpenNMT provides scripts to perform the important tasks of the framework. These are the *onmt_build_vocab*, *onmt_train* and *onmt_translate* scripts. These can be tuned with different parameters, which can all be centralized in a single YAML configuration file. We used different configuration files for the different experiments, and the most important ones can be found in the **Appendices** at the end of the paper.

The first phase of the pipelines is to build the vocabulary. This will allow converting the text into a suitable representation for the neural network, which will improve the results. The next phase is training the model. This is where we have to define the architecture of the model and how it will be trained. The resulting translation will largely depend on how we define this phase. The final phase is the translation. The script will output the results depending on the checkpoint we give, and we can then evaluate our model how we want using said results.

On top of the strong points mentioned above, this framework has been frequently used in research about Sign Language Translation, so this will greatly simplify the implementation of our experiments, making use of already existing configurations that have shown to work well in the past.

4.2 SpaCy

SpaCy[13] is an open-source Python library, which offers different high-quality Natural Language Processing tools (Honnibal & Montani, 2017). In this work, we used some of their tools to implement the different Data Augmentation Rules that were defined in **Chapter 3.4**.

The tools come in different packages, depending on the language the models were trained on, the size and the type of the dataset. In our case, we used the model *ca_core_news_sm*, the most efficient of the available models for Catalan which still has high accuracy, trained in a news dataset.

By loading this model, we get a Language object, which will allow us to analyze sentences using the Part-of-Speech (POS) Tagger and the Dependency Parser (DP).

[12] https://opennmt.net/
[13] https://spacy.io/

A POS tagger is a tool that aims to accurately assign POS labels to the different words of a sentence. In our case, we used it to remove Catalan words that belong to grammatical categories which don't exist in Catalan Sign Language, like determinants and prepositions, and to reorder some terms of the sentence in a specific way, like with adjectives and adverbials.

A Dependency Parser, in the context of Natural Language Processing, is a program that provides a syntactical analysis of a sentence, giving a tree that models the grammatical dependencies between words. This was used to reorder the terms in the sentences according to the Subject-Object-Verb structure defined in the LSC Grammar (Quer & Barberà, 2020). It was also used to find which noun or verb the adjectives and adverbials modified, to reorder them as previously mentioned.

Using this library, however, there was no easy way to implement rules 7-9 in **Table 3.5**, as it didn't provide ways to detect some subcategories (like quantifiers, temporal adverbials and frequency adverbials) which were the target of those rules.

4.3 Google Colab

Google Colab[14] is a free Jupyter Notebook service that is hosted by Google. It provides an already set up environment which is suited for machine learning. We will use this service to organize the Machine Translation part of the research in easy-to-understand notebooks[15].

The main reason why we used Google Colab is that they provide GPUs that can be freely used to a certain extent. Using this free tier also ensures that anyone can reproduce the experiments that were done in this research, without having any CPU/GPU limitations. However, its use is tightly tied to Google Drive, which is needed for the storage of the different weights, results, plots and data. That was a manageable issue in our low-resource scenario, but extensions of it could imply having to use another environment or paying for more resources.

4.4 Gensim

Gensim[16] is an open-source Python library designed to represent documents as semantic vectors (Řeh uřek et al., 2010). It does so in a way that is both efficient for the machine and easy to understand for humans.

This library learns these semantic vector representations by using different known unsupervised algorithms that observe the appearance patterns of the different words. In our case, for the creation of our own word embeddings, we used the Word2Vec algorithm (Mikolov et al., 2013), which is one of the supported vectors of OpenNMT. It is as easy as creating a Word2Vec object passing the path to the corpus file as the *corpus_file* argument, and it will automatically train an embedding model. We can then export the obtained vectors and use them in our neural networks.

To store these vectors, Gensim uses an object structure called KeyedVectors. This structure is common to all the learning algorithms that are used, so we used to convert pretrained vectors (Gutiérrez-Fandiño et al., 2021) into the Word2Vec format, by simply loading them in this common structure and then exporting them using the same method as before.

[14] https://colab.google/
[15] The notebooks are based on the ones in https://github.com/ymoslem/OpenNMT-Tutorial
[16] https://radimrehurek.com/gensim/

Chapter 5

Results

In this chapter, we will describe the results obtained by following the methods previously described in **Chapter 3**. The implications of said results in the Sign Language Translation field will later be discussed in **Chapter 6**, before concluding the work and proposing future work in **Chapter 7**.

5.1 Baseline

The first results that were examined came from the baseline model described in **Chapter 3.2**. As already said, this model will serve as a comparison for other models, to find if those approaches yield improvements or not. See in **Figure 5.1** a plot of the BLEU scores that this model got, and in **Figure 5.2** the METEOR scores.

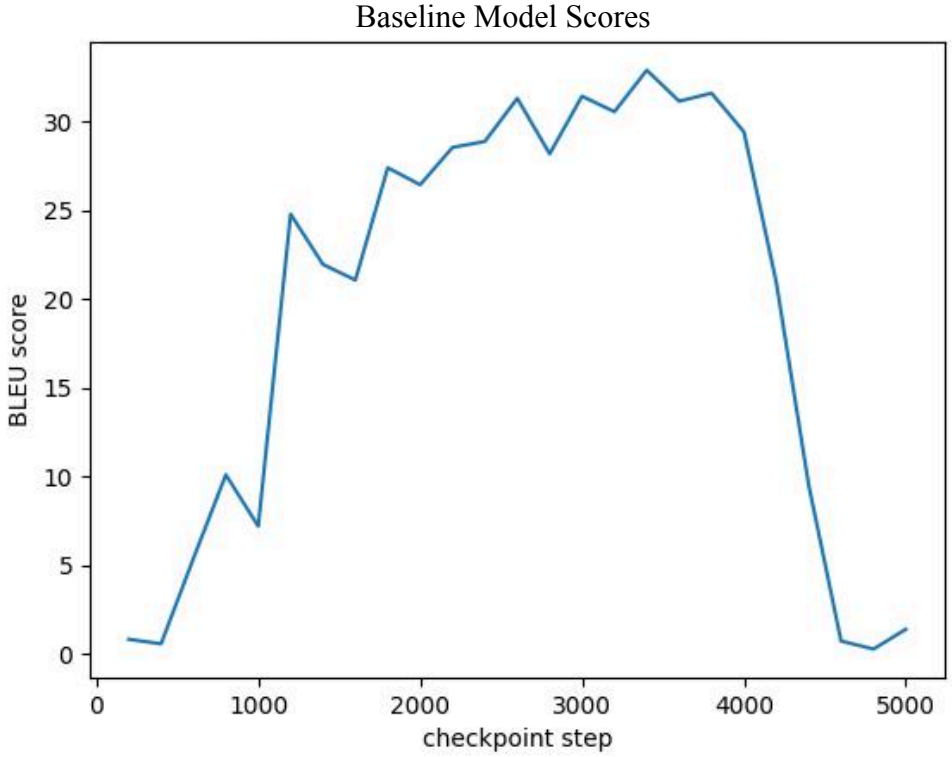

Figure 5.1: Plot of the BLEU scores of the Baseline model

The greatest result is achieved at step 3400, getting a BLEU score of 32.87 points. We can also see that around step 4000 the quality of the translations start decreasing greatly, which is the reason why further experiments were trained only for 4000 steps.

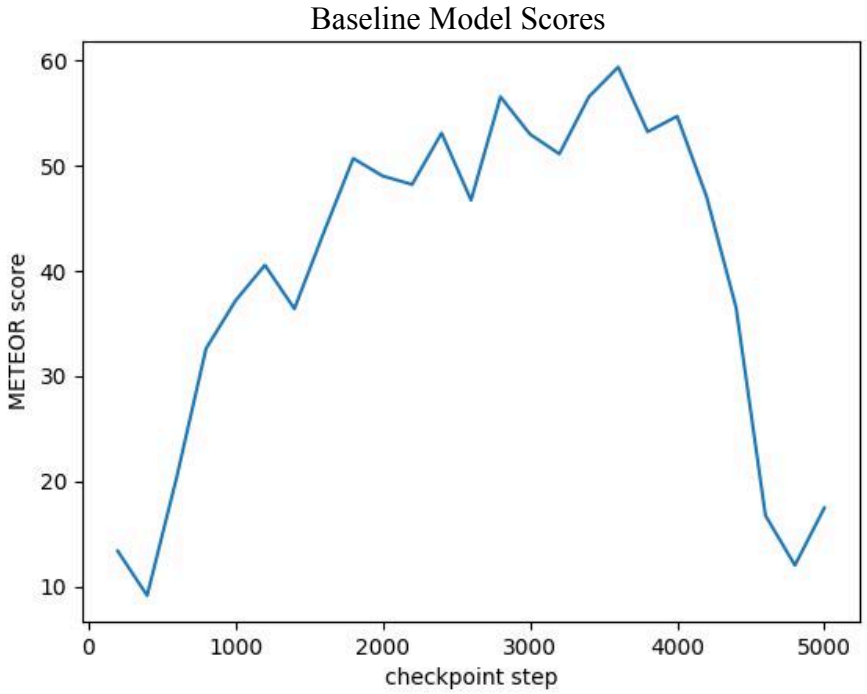

Figure 5.2: Plot of the METEOR scores of the Baseline model

Similar to the BLEU scores, the best results are achieved at epoch 3600, with a METEOR score of 59.37 points. The same decay in the quality of the translations can be observed at around epoch 4000.

5.2 Hyperparameter Optimized Model

As explained in **Chapter 3.3**, the configuration in **Appendix B** was used as a starting point. See in **Figure 5.3** and **Figure 5.4** the BLEU and METEOR scores respectively of this model in comparison to the baseline.

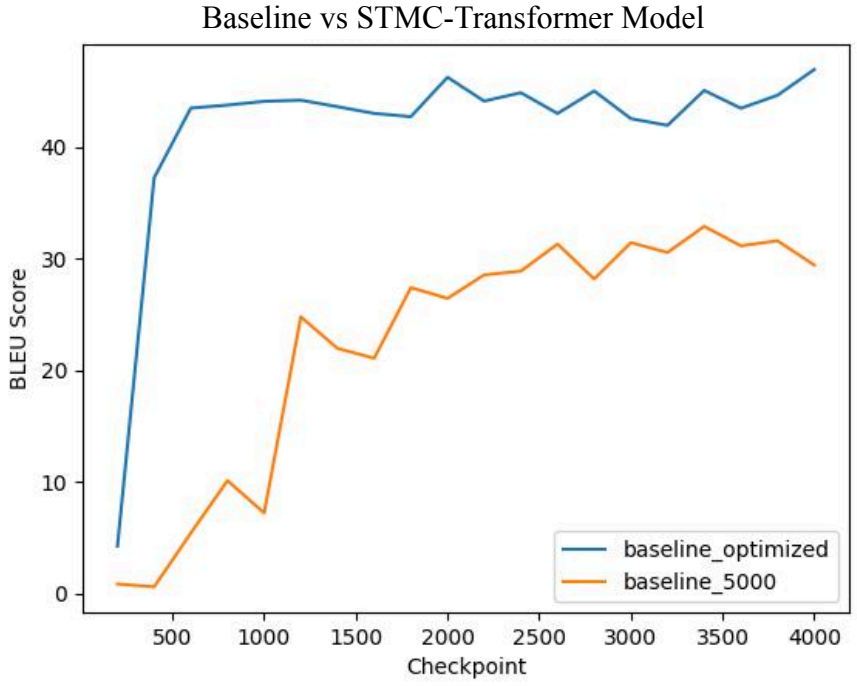

Figure 5.3: Plot of the BLEU scores of the borrowed config model compared to the baseline model

As we can see, this new model yields great improvements compared to the baseline model, converging way earlier, at around epoch 600. The best translations are found at epoch 4000, achieving a maximum BLEU score of ~ 46.93 points.

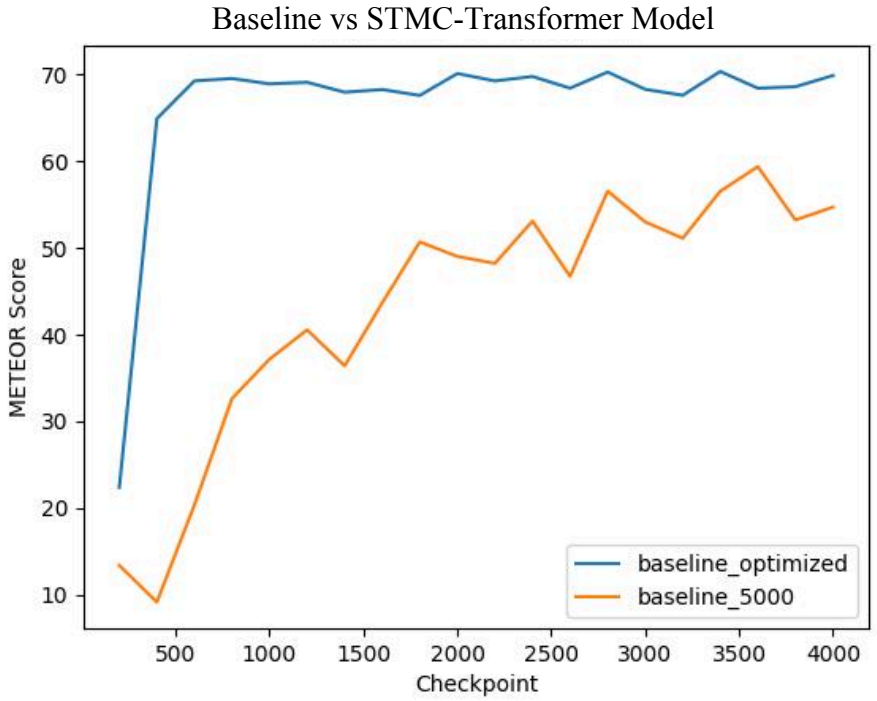

Figure 5.4: Plot of the METEOR scores of the borrowed config model compared to the baseline model

The same converging behavior can be seen with the METEOR scores, that now achieve a max score of ~ 70.724 points at epoch 3400.

Now that we have seen the results of this improved model, let's see how the modification of some of the hyperparameters affected the translation quality, evaluated by with the BLEU metric. As a reminder, these hyperparameters can be found in **Table 3.3**.

# of layers	Max BLEU score	Best Epoch
1	46.227	4000
2	46.925	4000
3	47.710	3600

Table 5.1: Best BLEU scores with different values of the *layers* hyperparameter

In the case of the number of layers for both the encoder and decoder, the more the layers, the better the resulting translation, achieving the best BLEU score earlier with 3 layers.

Word_vec & Hidden size	Max BLEU score	Best Epoch
256	46.365	3000
512	46.925	4000
1024	48.547	3000

Table 5.2: Best BLEU scores with different values of the *word_vec & hidden size* hyperparameters

Increasing the size of the hidden size of the RNN yields an improvement in the scores, achieving a maximum score of 48.547 BLEU points in 3000 epochs.

Init Learning rate	Max BLEU score	Best Epoch
0.1	45.606	3400
0.5	46.925	4000
0.7	47.383	2000

Table 5.3: Best BLEU scores with different values of the *learning rate* hyperparameter

The next hyperparameter is the initial learning rate. The higher this value, the higher the maximum BLEU score gets. With initial learning rate 0.7, it achieves its highest peak in only 2000 epochs.

Transformer_ff	Max BLEU score	Best Epoch
1024	46.925	3200
2048	46.925	4000
4096	47.197	1800

Table 5.4: Best BLEU scores with different values of the *transformer_ff* hyperparameter

The last hyperparameter is the size of the feed forward layers of the transformer. The values 1024 and 2048 get similar results, the first one getting them earlier. However, 4096 has slightly better results and gets them much earlier.

Pretrained embeddings	Max BLEU score	Best Epoch
None	46.925	4000
Only Source	46.379	2400
Only Target	47.295	3000
Both	45.607	3000

Table 5.5: Best BLEU scores with different combinations of pretrained embeddings

Finally, there are the results of using different combinations of pretrained embeddings, explained in **Chapter 3.3**. The best results were achieved by only using the target Catalan embeddings, and using the source embeddings, both alone and in combination with the target embeddings, gave worse results than without using them.

After finding the best hyperparameter values, we tried combining them, achieving the results displayed in **Figure 5.5** and **Figure 5.6**.

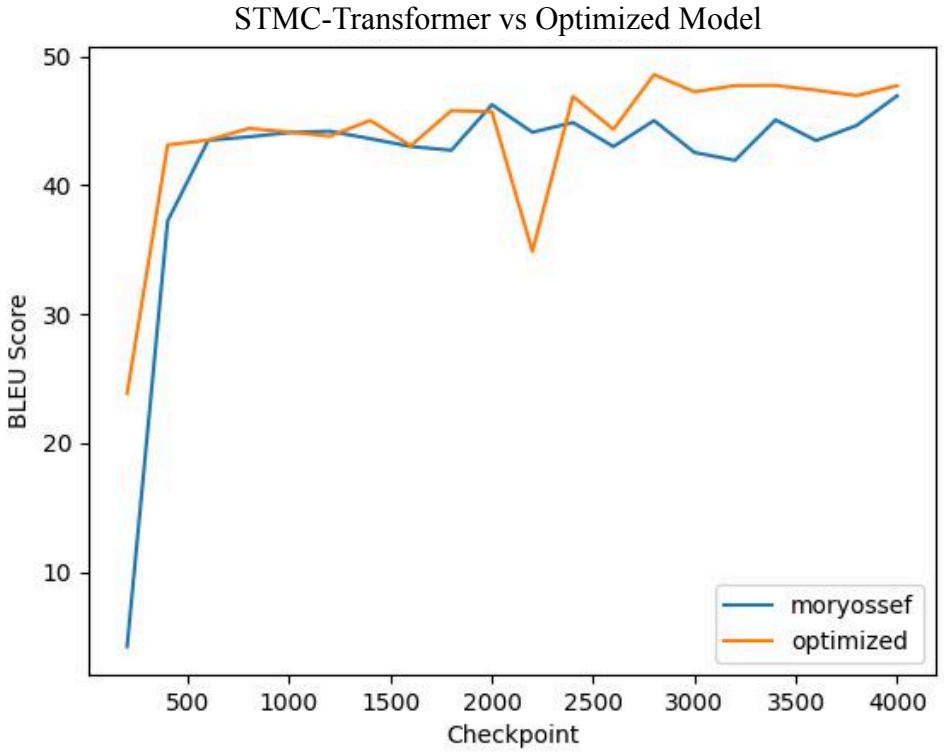

Figure 5.5: Plot of the BLEU scores of the optimized model compared to the borrowed config model

A slight improvement of the BLEU scores can be seen with the new model in comparison with the old one, achieving the highest score at epoch 2800, with a peak of 48.568 points. This is the highest score we found yet. We can also see that it learns faster in the starting steps.

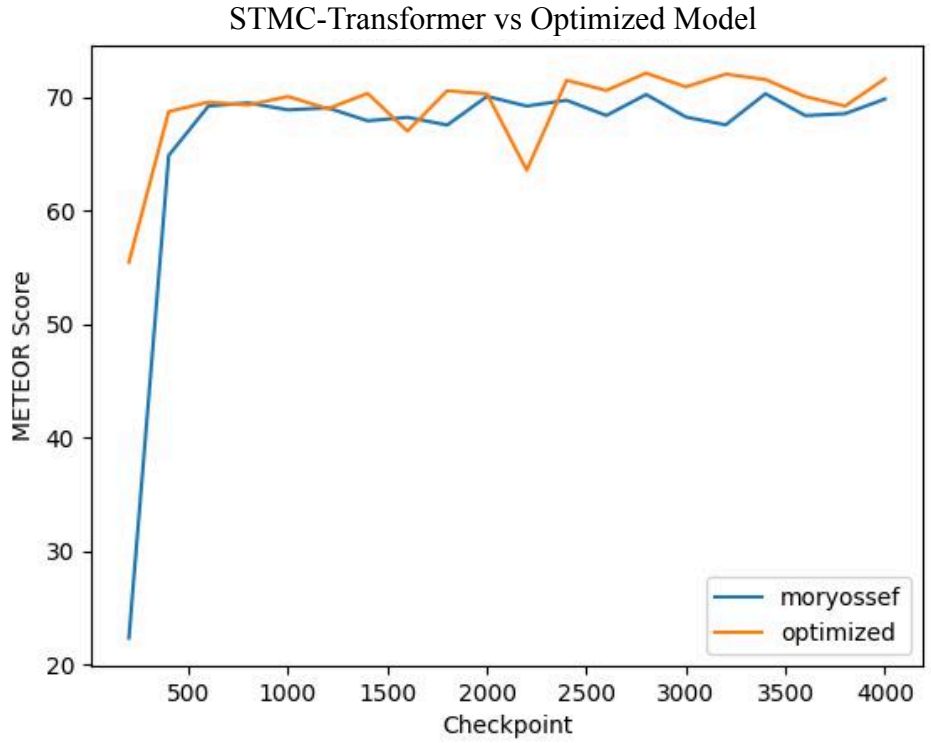

Figure 5.6: Plot of the METEOR scores of the optimized model compared to the borrowed config model

The same behavior can be observed with the METEOR scores, which now achieve a peak of 72.563 points at epoch 2800.

5.3 Data Augmented Model

The following figures show how the different Data Augmentation approaches affected the translation quality. First, we will see the results of the General Augmentation rules (see **Table 3.4**).

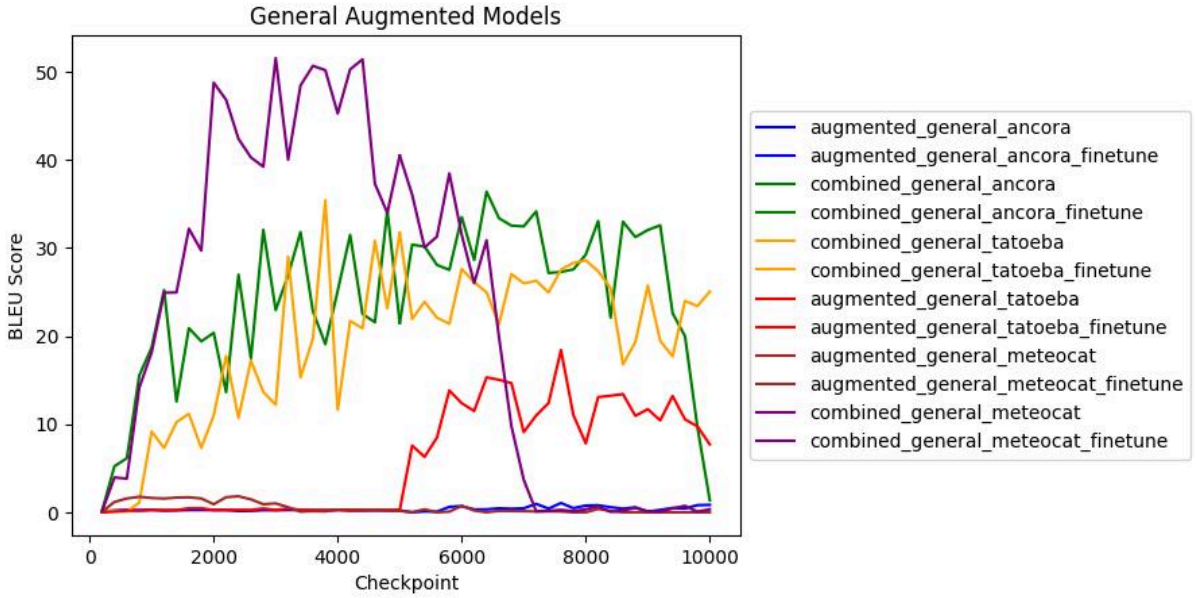

Figure 5.7: Plot of the BLEU scores achieved by the General Augmented Models. *Augmented* models are pretrained on just the synthetic data, *combined* models with the combination of synthetic and real

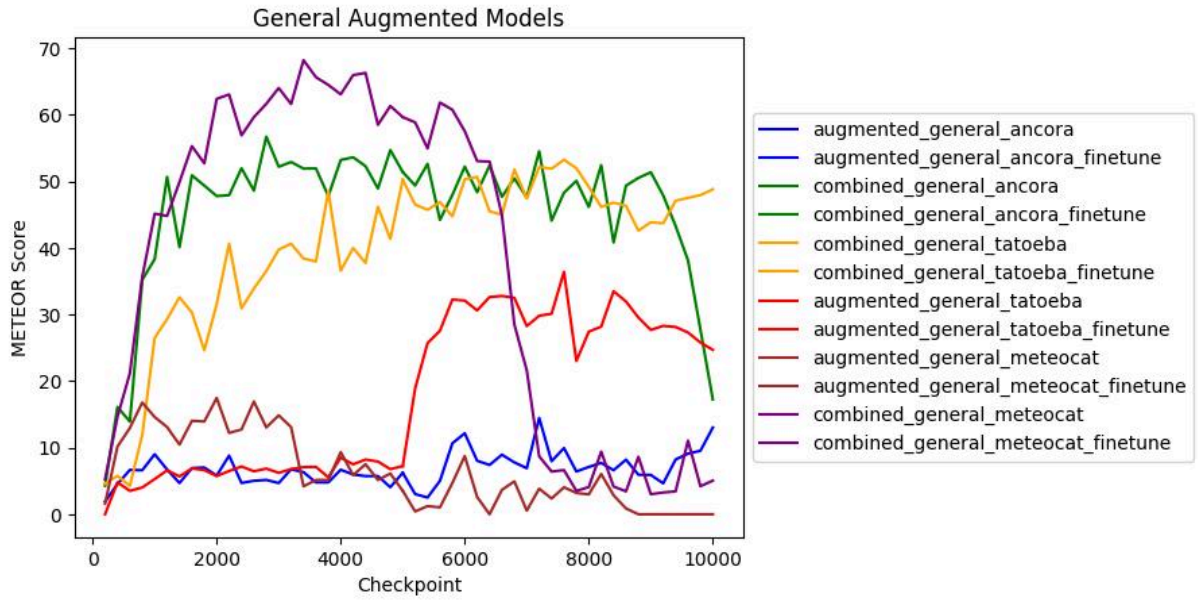

Figure 5.8: Plot of the METEOR scores achieved by the General Augmented Models. *Augmented* models are pretrained on just the synthetic data, *combined* models with the combination of synthetic and real

The best results with the General Augmentation rules are achieved by the model pretrained on the combination of synthetic MeteoCat glosses and the real data. This model performed significantly better than the others.

Now we will see the results yielded by our LSC Augmentation rules, defined in **Table 3.5**.

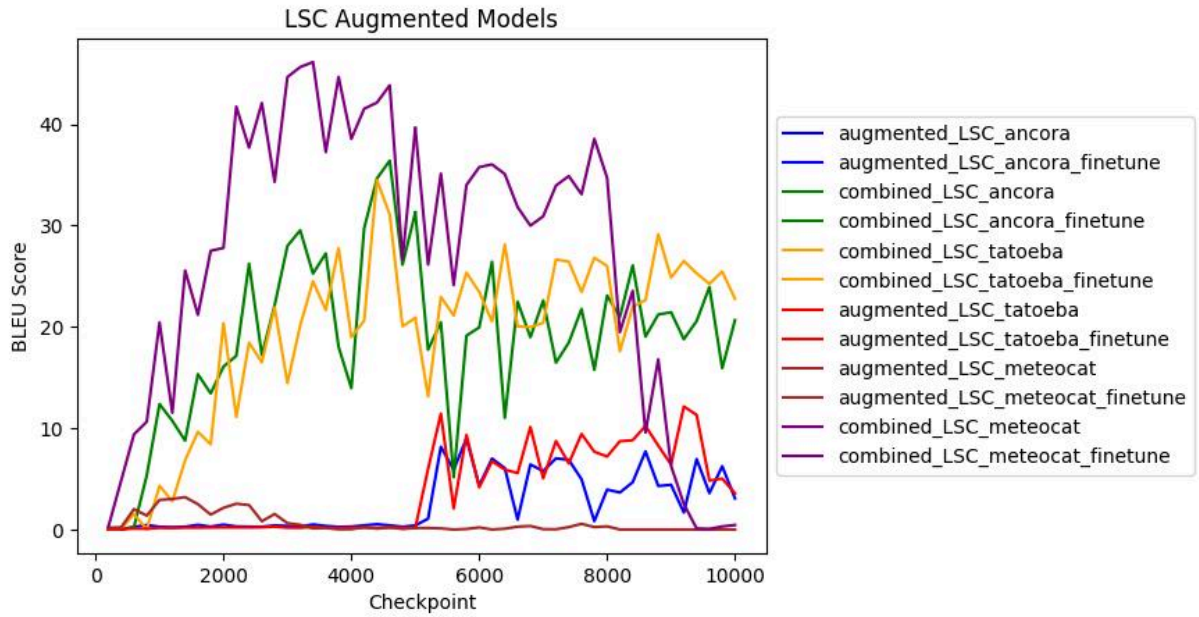

Figure 5.9: Plot of the BLEU scores achieved by the LSC Augmented Models. *Augmented* models are pretrained on just the synthetic data, *combined* models with the combination of synthetic and real

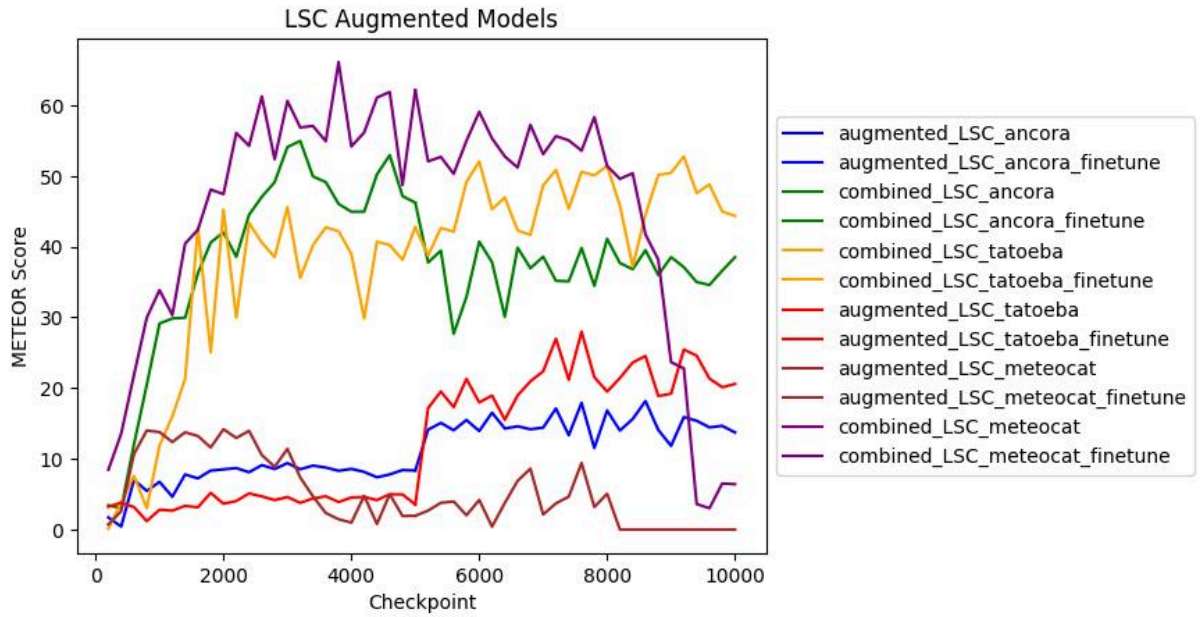

Figure 5.10: Plot of the METEOR scores achieved by the LSC Augmented Models. *Augmented* models are pretrained on just the synthetic data, *combined* models with the combination of synthetic and real

A behavior similar to the one with the General Augmentation rules can be seen in **Figures 5.9-5.10**. The MeteoCat synthetic glosses give the best results in the pretraining part when combining them with the real data. This time, however, the translation quality is better maintained on the fine-tuning phase

Finally, in the plots below, we will see the best data augmented models in comparison with the baseline model.

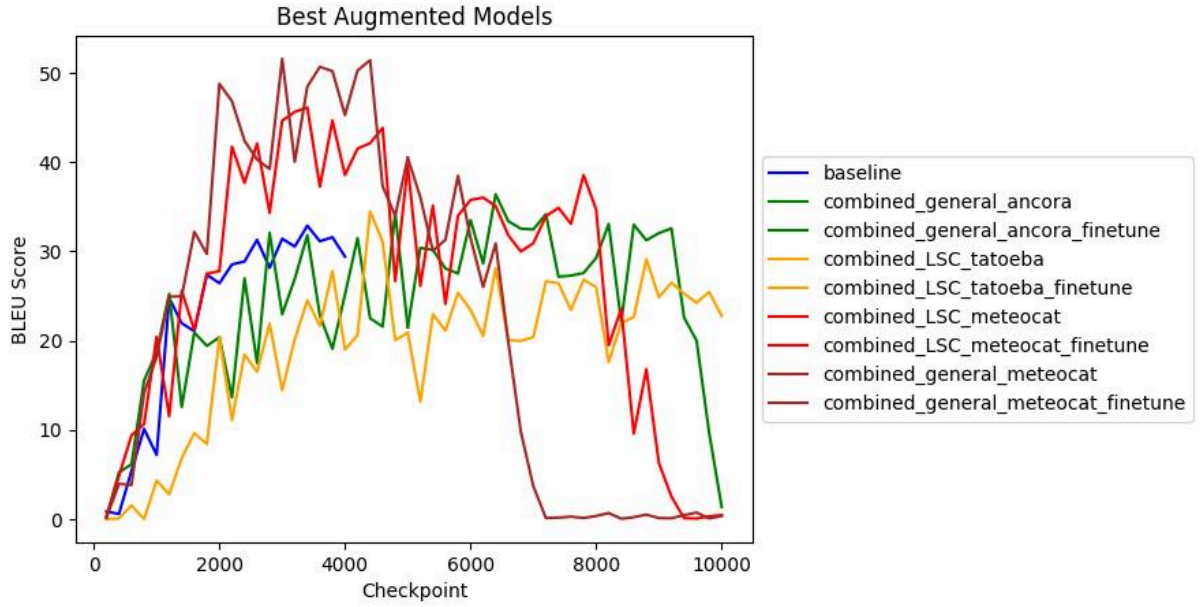

Figure 5.11: Plot of the BLEU scores achieved by the Best Augmented Models. *Augmented* models are pretrained on just the synthetic data, *combined* models with the combination of synthetic and real

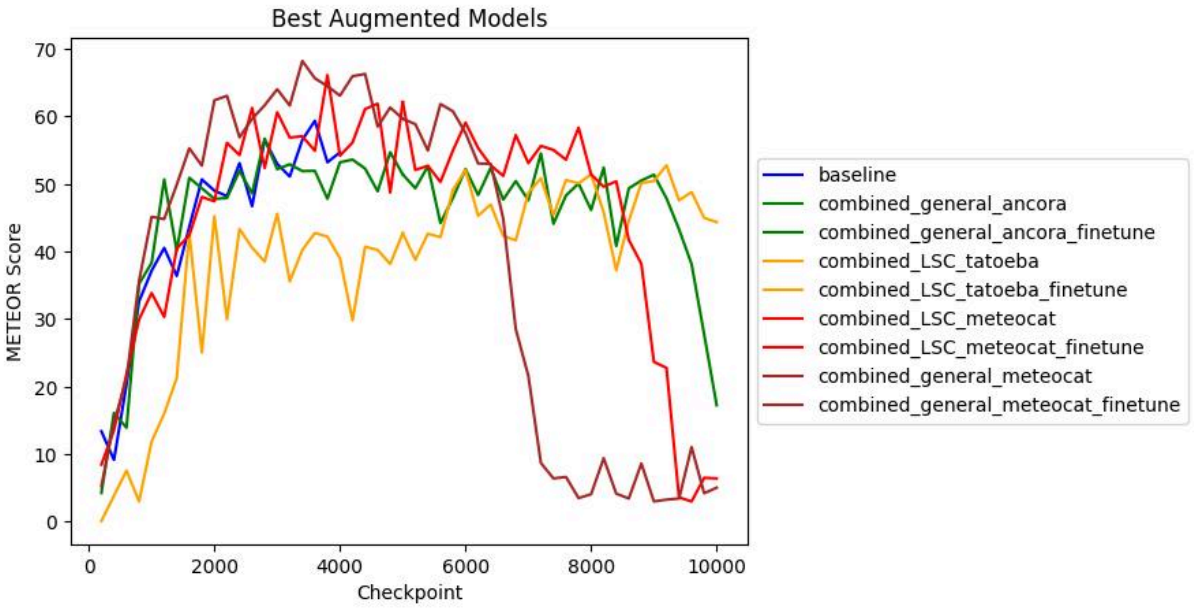

Figure 5.12: Plot of the METEOR scores achieved by the Best Augmented Models. *Augmented* models are pretrained on just the synthetic data, *combined* models with the combination of synthetic and real

In the plots, we can see that both in terms of BLEU and METEOR scores, the best results are achieved by the model pretrained on the synthetic Meteocat glosses obtained with the General Augmentation rules, when combined with the real data. The quality of the translations is much better than the one yielded by the baseline model, yielding a top score of 51.56 BLEU, the highest we got of all the work.

	BLEU↑	METEOR↑
Baseline Model	32.87	59.37
General Augmented Tatoeba	18.41	36.42
General Combined Tatoeba	35.40	53.25
LSC Augmented Tatoeba	12.14	27.98
LSC Combined Tatoeba	34.48	52.78
General Augmented AnCora	1.05	14.47
General Combined AnCora	36.39	56.72
LSC Augmented AnCora	8.97	18.17
LSC Combined AnCora	36.39	54.96
General Augmented Meteocat	1.81	17.47
General Combined Meteocat	**51.56**	**68.22**
LSC Augmented Meteocat	3.19	14.18
LSC Combined Meteocat	46.12	66.13

Table 5.6: Best scores of the different Data Augmented Models. Bold results are the best ones achieved

Chapter 6

Discussion

Before concluding the work, we will take a look back on the objectives defined in **Chapter 1.3**. The first one was attempting an alternative approach to the one by Sanabre & Cardús (2012), by using Neural Machine Translation and tuning the hyperparameters, as done in **Chapter 3.2** and **Chapter 3.3**. The second one was performing Data Augmentation by using already existing rules, also creating our own, as done in **Chapter 3.4**. We will discuss if the results were satisfactory or not, trying to understand the reasons and implications it might have on the field.

6.1 Hyperparameter Optimized Model

As we don't have any reference results in the Gloss-to-Text translation task on this dataset, we will have to get creative to evaluate our research. The other known research done on this dataset was the one by Sanabre & Cardús (2012), which studied the Text-to-Gloss direction. In their experiments, they achieved a top BLEU score of 48.77 points, so this will serve as a rough reference of the scores that can be achieved in this dataset, but unreliable for the comparison of our results, being from a different task. We also have to keep in mind that BLEU is a parametrized metric, meaning that the comparison of these scores isn't fair unless the parameters are the same (Post, 2018).

The BLEU scores could then be analyzed without comparing them to others, following the rough guidelines shown in **Figure 6.1**. As we can see in the figure, we start considering the translations to be of good quality on the 40-50 BLEU range.

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Figure 6.1: Interpretation guidelines for BLEU scores. Taken from Google Cloud AutoML Documentation[17]

We will also use the scores yielded by the baseline model, which gave scores of 32.87 in terms of BLEU and 59.37 in terms of METEOR. These will be a key indicator of the improvements we get with the different experiments.

With that said, the best results we got during the optimization of the baseline were ~ 48.57 in terms of BLEU and ~ 72.56 in terms of METEOR. It also got an improvement of around +1.64 BLEU over the STMC-Transformer (Yin & Read, 2020) implementation. This was achieved by combining the best hyperparameter values that were found during the search performed in **Chapter 3.3**.

[17] https://cloud.google.com/translate/automl/docs/evaluate

The results could surely be improved by trying all the possible combinations with the values we chose to study, but this strategy showed results in this work, getting us the best scores out of all the experiments. There is not enough evidence, however, to affirm that this would always be the case if applied to other research.

Our own pretrained embeddings improved the results in only one case, when using them on the target side of the translation. Using better quality word embeddings could improve the translation quality even further. Even though the gloss embeddings didn't yield better scores, it could be attempted again when the LSC Corpus (Barberà et al., 2015) is released.

Multiple of the optimal hyperparameter values implied the increase in size of the neural network, which combined could reduce the performance of the model, increasing the time it takes to train the model and translate the sentences. Depending on the hardware, it would be wiser to reduce the number of some hyperparameters, like the *transformer_ff* to 2048, keeping the *hidden_size* and *word_vec_size* at 1024 which showed great results (see **Table 5.2**).

6.2 Data Augmented Model

As we can see in **Figures 5.7** to **5.10**, the models pretrained on just the synthetic data from the three datasets fail to learn meaningful information for the translation of the real data. In a 0-shot setting, the pretrained part of the model doesn't achieve a BLEU score higher than 1, except the one using the MeteoCat glosses, explaining why the scores after fine-tuning don't come near the ones achieved by the baseline model.

For the Tatoeba and AnCora datasets, the models pretrained on a combination of synthetic and real data gave results closer to the baseline. If we look at the best BLEU scores only, as reported in **Table 5.6** for both datasets augmented using both sets of rules, we can see that they all achieve some improvement over the baseline model, the best ones being those with the augmented AnCora data. However, if we look at **Figures 5.11** and **5.12**, we can see that they are more unstable than the baseline, with multiple peaks, so the latter would be considered more consistent. Also, in terms of METEOR scores, the baseline scores are still better.

For the MeteoCat dataset, things get interesting, as it gave arguably the best results out of the entire research. These models outperformed even the hyperparameter optimized model in terms of BLEU scores, without being optimized themselves, achieving results of 51.56 BLEU. In **Figure 5.11** and **5.12** we can also see that the General Augmentation rules performed better than the LSC Specific Augmentation rules we came up with.

To try to understand better why the best models worked better, we performed further analysis of the augmented datasets. Different information on the datasets that were augmented can be seen in **Table 6.1**. Furthermore, the coverages on the annotated are shown in **Table 6.2**.

	Tatoeba	AnCora	Meteocat
Sentences	2852	3059	154
Glosses General	2519	4437	242
Glosses LSC	2614	3899	252

Table 6.1: Sizes of the augmented datasets

		AnCora	AnCora +Train	Tatoeba	Tatoeba +Train	MeteoCat	MeteoCat +Train	Train
General	**Train**	39.85	100.0	31.95	100.0	19.92	100.0	100.0
	Dev	52.85	93.50	40.65	89.43	26.02	90.24	87.80
	Test	47.17	96.23	36.79	93.40	30.19	94.34	92.45
LSC	**Train**	38.72	100.0	33.08	100.0	21.80	100.0	100.0
	Dev	51.22	93.50	42.28	89.43	30.08	90.24	87.80
	Test	47.17	96.23	39.62	93.40	34.91	94.34	92.45

Table 6.2: Coverage of the datasets with the combination of the data augmented glosses and the real glosses.

As we can see from the tables, the MeteoCat dataset is both the smallest and the one with less coverage over the training, development and testing sets. However, it was the best, and there could be multiple reasons for that. The better results could be tied to the coverage of certain words of the weather domain which may not be covered by the other datasets. Another explanation could be the syntactic difference between common and weather related sentences, where impersonal phrases are frequent.

Further analysis could be performed in future research, as well as attempting the combination of the hyperparameter optimized model with the data augmentation techniques. For now, we will conclude by emphasizing the importance of the augmented dataset being from the same domain as the real annotated data.

Chapter 7

Conclusions & Future Work

In this work, we attempted for the first time to apply Neural Machine Translation techniques between Catalan Sign Language and written Catalan. We have also done the first documented attempt at LSC Gloss-to-Text translation, providing a comparison for future research on the same dataset, while also providing methods that can easily be reproduced on other corpora that might be released in the future, like the LSC Corpus.

Additionally, we searched the best values for some of the hyperparameters of the architecture, improving the quality of the translations by combining the optimal configurations. Lastly, we attempted Data Augmentation with the use of rules, applying the already existing General Augmentation rules provided by Moryossef et al. (2021), also creating our own set of rules based on the LSC Grammar written by Quer & Barberà (2020). Although the two sets of rules achieved significant improvements over the baseline, our set of LSC Specific rules didn't achieve better results than the General ones.

There are many directions future research on the topic could take. For starters, the Text-to-Gloss translation task could be explored, applying the same methods we applied in this work. The two directions could be studied simultaneously, analyzing additional factors like the sub-wording. The augmentation part of the research could also be improved upon, by revisiting and expanding the LSC augmentation rules we came up with. With the help of experts in the domain, a more rigorous set of rules could be proposed, possibly improving the results of future translation systems and help with the creation of synthetic datasets like done by Perea-Trigo et al. (2024) . Other approaches could also be followed, like leveraging transfer learning like done by Egea Gómez et al. (2022) or using linguistic features to enhance the inputs like McGill et al. (2023).

Overall, the future of research on Catalan Sign Language will be deeply tied to the evolution of the LSC Corpus Project. The greatest limitation to the current research is the extremely limited amount of data available. Even though the results can be good in the current dataset, they would generalize poorly due to the lack of data, and the small amount being from the weather domain would hinder said generalization even more. Thus, the publication of a larger, more general collection of data would greatly help to achieve better translations. This would get us closer to the construction of a full Machine Translation pipeline, which will some day allow easy, accurate and non-intrusive communication between the Deaf and Hard of Hearing community and the non-signing population.

Bibliography

Banerjee, S., & Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In J. Goldstein, A. Lavie, C.-Y. Lin, & C. Voss (Eds.), Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization (pp. 65–72). Association for Computational Linguistics. https://aclanthology.org/W05-0909

Barberà, G., & Quer, J. (2012). The meaning of space in Catalan Sign Language (LSC) Reference, specificity and structure in signed discourse.

Barberà, G., Quer, J., & Frigola, S. (2015). Primers passos cap a la documentació de discurs signat: el projecte pilot de constitució del corpus de la llengua de signes catalana. *Treballs de Sociolingüística Catalana*, 287–302. https://api.semanticscholar.org/CorpusID:148346234

Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information.

Camgöz, N. C., Hadfield, S., Koller, O., Ney, H., & Bowden, R. (2018). Neural Sign Language Translation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7784–7793. https://doi.org/10.1109/CVPR.2018.00812

Chiruzzo, L., McGill, E., Gómez, S. E., & Saggion, H. (2022). Translating Spanish into Spanish Sign Language: Combining rules and data-driven approaches. *Proceedings of the Fifth Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2022)*, 75–83.

Egea Gómez S, Chiruzzo L, McGill E, Saggion H. (2022). Linguistically enhanced text to sign gloss machine translation. In: Rosso P, Basile V, Martínez R, Métais E, Meziane F, (eds). Natural Language Processing and Information Systems, 27th International Conference on Applications of Natural Language to Information Systems, NLDB 2022 Proceedings; 2022 June 15-17; Valencia, Spain. Cham: Springer; 2022. p. 172-83. DOI: 10.1007/978-3-031-08473-7_16

Forster, J., Schmidt, C., Hoyoux, T., Koller, O., Zelle, U., Piater, J., & Ney, H. (2012, June). *RWTH-PHOENIX-Weather: A Large Vocabulary Sign Language Recognition and Translation Corpus*.

Gutiérrez-Fandiño, A., Armengol-Estapé, J., Gonzalez-Agirre, A., Carrino, C. P., de Gibert, O., & Villegas, M. (2021). *Catalan Word Embeddings in FastText*. Zenodo. https://doi.org/10.5281/zenodo.4522041

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Klein, G., Kim, Y., Deng, Y., Nguyen, V., Senellart, J., & Rush, A. M. (2018). *OpenNMT: Neural Machine Translation Toolkit*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., & Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *CoRR*, *abs/1701.02810*. http://arxiv.org/abs/1701.02810

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., & Herbst, E. (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* (pp. 177-180). Association for Computational Linguistics. http://www.aclweb.org/anthology/P07-2045

Kudo, T. (2018a). Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates.

Kudo, T., & Richardson, J. (2018b). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In E. Blanco & W. Lu (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 66–71). Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-2012

McGill, E., Chiruzzo, L., Gómez, S. E., & Saggion, H. (2023). Part-of-Speech tagging Spanish Sign Language data and its applications in Sign Language machine translation. *Proceedings of the Second Workshop on Resources and Representations for Under-Resourced Languages and Domains (RESOURCEFUL-2023)*, 70–76.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space.

Moryossef, A., & Goldberg, Y. (2021). Sign Language Processing. https://sign-language-processing.github.io

Moryossef, A., Yin, K., Neubig, G., & Goldberg, Y. (2021). *Data Augmentation for Sign Language Gloss Translation*.

Núñez-Marcos, A., Perez-de-Viñaspre, O., & Labaka, G. (2023). A survey on Sign Language machine translation. In *Expert Systems with Applications* (Vol. 213). Elsevier Ltd. https://doi.org/10.1016/j.eswa.2022.118993

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–318. https://doi.org/10.3115/1073083.1073135

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. http://www.aclweb.org/anthology/D14-1162

Perea-Trigo, M., Botella-López, C., Martínez-del-Amor, M. Á., Álvarez-García, J. A., Soria-Morillo, L. M., & Vegas-Olmos, J. J. (2024). Synthetic Corpus Generation for Deep Learning-Based Translation of Spanish Sign Language. *Sensors*, *24*(5). https://doi.org/10.3390/s24051472

Porta, J., López-Colino, F., Tejedor, J., & Colás, J. (2014). A rule-based translation from written Spanish to Spanish Sign Language glosses. *Computer Speech and Language*, *28*(3), 788–811. https://doi.org/10.1016/j.csl.2013.10.003

Post, M. (2018). A Call for Clarity in Reporting BLEU Scores. In O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, L. Specia, M. Turchi, & K. Verspoor (Eds.), *Proceedings of the Third Conference on Machine Translation: Research Papers* (pp. 186–191). Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-6319

Quer, J., & Barberà, G. (2020). *A Grammar of Catalan Sign Language (LSC)*. https://thesignhub.eu/grammar/lsc

Řeh uřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New for NLP Frameworks*, 45–50.

Sanabre, G. M., & Cardús, T. B. (2012). *Desenvolupament d'un sistema de traducció automàtica estadística cap a la llengua de signes catalana.*

Taulé, M., Martí, M. A., & Recasens, M. (2008). AnCora: Multilevel Annotated Corpora for Catalan and Spanish. *Proceedings of 6th International Conference on language Resources and Evaluation. http://www.lrec-conf.org/proceedings/lrec2008/pdf/35_paper.pdf*

Tiedemann, J. (2012). Parallel Data, Tools and Interfaces in OPUS. In N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, & S. Piperidis (Eds.),

Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12) (pp. 2214–2218). European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *CoRR*, *abs/1706.03762*. http://arxiv.org/abs/1706.03762

Yin, K., Moryossef, A., Hochgesang, J., Goldberg, Y., & Alikhani, M. (2021). Including Signed Languages in Natural Language Processing. In C. Zong, F. Xia, W. Li, & R. Navigli (Eds.), *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)* (pp. 7347–7360). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.570

Yin, K., & Read, J. (2020). *Better Sign Language Translation with STMC-Transformer*.

Zhang, X., & Duh, K. (2021). Approaching Sign Language Gloss Translation as a Low-Resource Machine Translation Task. In D. Shterionov (Ed.), *Proceedings of the 1st International Workshop on Automatic Translation for Signed and Spoken Languages (AT4SSL)* (pp. 60–70). Association for Machine Translation in the Americas. https://aclanthology.org/2021.mtsummit-at4ssl.7

Appendices

Appendix A: Baseline Model configuration file

```
## Where the samples will be written
save_data: models/baseline_5000
## Where the vocab(s) will be written
src_vocab: models/baseline_5000/baseline_5000.vocab.src
tgt_vocab: models/baseline_5000/baseline_5000.vocab.tgt

feat_merge: "concat"
# Prevent overwriting existing files in the folder
overwrite: True

# Corpus opts:
data:
    corpus_1:
        path_src: data/meteo.lsc-filtered.lsc.subword.train
        path_tgt: data/meteo.cat-filtered.cat.subword.train
        transforms: [filtertoolong]
    valid:
        path_src: data/meteo.lsc-filtered.lsc.subword.dev
        path_tgt: data/meteo.cat-filtered.cat.subword.dev
        transforms: [filtertoolong]
# Train on a single GPU
world_size: 1
gpu_ranks: [0]

# Where to save the checkpoints
save_model: models/baseline_5000/baseline_5000
save_checkpoint_steps: 200
train_steps: 5000
valid_steps: 200

# Tensorboard
tensorboard: true
tensorboard_log_dir: logs

# Batching
bucket_size: 144
batch_type: "tokens"
batch_size: 256

# Seed
seed: 7
```

Appendix B: STMC-Transformer Model configuration file

```
 ## Where the samples will be written
save_data: models/baseline_optimized
## Where the vocab(s) will be written
src_vocab: models/baseline_optimized/baseline_optimized.vocab.src
tgt_vocab: models/baseline_optimized/baseline_optimized.vocab.tgt

feat_merge: "concat"
# Prevent overwriting existing files in the folder
overwrite: True

# Corpus opts:
data:
    corpus_1:
        path_src: data/meteo.lsc-filtered.lsc.subword.train
        path_tgt: data/meteo.cat-filtered.cat.subword.train
        transforms: [filtertoolong]
    valid:
        path_src: data/meteo.lsc-filtered.lsc.subword.dev
        path_tgt: data/meteo.cat-filtered.cat.subword.dev
        transforms: [filtertoolong]
# Train on a single GPU
world_size: 1
gpu_ranks: [0]

# Where to save the checkpoints
save_model: models/baseline_optimized/baseline_optimized
save_checkpoint_steps: 200
train_steps: 2600
valid_steps: 200

# Architecture
layers: 2
word_vec_size: 512
hidden_size: 512
heads: 8
encoder_type: transformer
decoder_type: transformer
position_encoding: true
transformer_ff: 2048
dropout: 0.1

tensorboard: true
tensorboard_log_dir: logs

# Seed
seed: 7

# Optimization
bucket_size: 72
batch_size: 128
accum_count: 3
max_generator_batches: 2
optim: adam
adam_beta2: 0.998
decay_method: noam
warmup_steps: 3000
learning_rate: 0.5
max_grad_norm: 0
```

```
param_init: 0
param_init_glorot: true
label_smoothing: 0.1
```

Appendix C: Optimized Model configuration file

```
 ## Where the samples will be written
save_data: models/baseline_optimized
## Where the vocab(s) will be written
src_vocab: models/baseline_optimized/baseline_optimized.vocab.src
tgt_vocab: models/baseline_optimized/baseline_optimized.vocab.tgt

feat_merge: "concat"
# Prevent overwriting existing files in the folder
overwrite: True

# Corpus opts:
data:
    corpus_1:
        path_src: data/meteo.lsc-filtered.lsc.subword.train
        path_tgt: data/meteo.cat-filtered.cat.subword.train
        transforms: [filtertoolong]
    valid:
        path_src: data/meteo.lsc-filtered.lsc.subword.dev
        path_tgt: data/meteo.cat-filtered.cat.subword.dev
        transforms: [filtertoolong]
# Train on a single GPU
world_size: 1
gpu_ranks: [0]

# Where to save the checkpoints
save_model: models/baseline_optimized/baseline_optimized
save_checkpoint_steps: 200
# keep_checkpoint: 1
train_steps: 4000
valid_steps: 200

# Architecture
layers: 3
word_vec_size: 1024
hidden_size: 1024
heads: 8
encoder_type: transformer
decoder_type: transformer
position_encoding: true
transformer_ff: 4096
dropout: 0.1

# Early stopping
#early_stopping: 3
#early_stopping_criteria: accuracy ppl

tensorboard: true
tensorboard_log_dir: logs

# Seed
seed: 7

# Embeddings
tgt_embeddings: <path_to_target_embeddings>
```

```
embeddings_type: word2vec

# Optimization
bucket_size: 72
batch_size: 128
accum_count: 3
max_generator_batches: 2
optim: adam
adam_beta2: 0.998
decay_method: noam
warmup_steps: 3000
learning_rate: 0.7
max_grad_norm: 0
param_init: 0
param_init_glorot: true
label_smoothing: 0.1
```

Appendix D: Pretraining configuration file

```
## Where the samples will be written
save_data: models/augmented
## Where the vocab(s) will be written
src_vocab: models/augmented/augmented.vocab.src
tgt_vocab: models/augmented/augmented.vocab.tgt
feat_merge: "concat"
# Prevent overwriting existing files in the folder
overwrite: True

# Corpus opts:
data:
   augmented:
      path_src: <path_to_src_augmented_training_data>
      path_tgt: <path_to_tgt_augmented_training_data>
      transforms: [filtertoolong]
   valid:
      path_src: <path_to_src_augmented_dev_data>
      path_tgt: <path_to_src_augmented_dev_data>
      transforms: [filtertoolong]
# Train on a single GPU
world_size: 1
gpu_ranks: [0]

# Where to save the checkpoints
save_model: models/augmented/augmented
save_checkpoint_steps: 200
train_steps: 5000
valid_steps: 200

# Tensorboard
tensorboard: true
tensorboard_log_dir: logs

# Batching
bucket_size: 144
batch_type: "tokens"
batch_size: 256
```

Appendix E: Fine-tuning configuration file

```
## Where the samples will be written
save_data: models/finetune
## Where the vocab(s) will be written
src_vocab: models/finetune/finetune.vocab.src
tgt_vocab: models/finetune/finetune.vocab.tgt
feat_merge: "concat"
# Prevent overwriting existing files in the folder
overwrite: True

# Corpus opts:
data:
    meteo:
        path_src: <path_to_src_training_data>
        path_tgt: <path_to_tgt_training_data>
        transforms: [filtertoolong]
    valid:
        path_src: <path_to_src_dev_data>
        path_tgt: <path_to_src_dev_data>
        transforms: [filtertoolong]
# Train on a single GPU
world_size: 1
gpu_ranks: [0]

# Where to save the checkpoints
save_model: models/finetune/finetune
train_from: models/augmented/augmented_step_5000.pt
save_checkpoint_steps: 200
# keep_checkpoint: 1
train_steps: 10000
valid_steps: 200
# early_stopping: 4

# Tensorboard
tensorboard: true
tensorboard_log_dir: logs

# Batching
bucket_size: 144
batch_type: "tokens"
batch_size: 256

#both_embeddings: glove_dir/glove-sbwc.i25.vec
# to set src and tgt embeddings separately:
#src_embeddings: glove_dir/glove-sbwc.i25.vec
# tgt_embeddings: ...

# supported types: GloVe, word2vec
#embeddings_type: "GloVe"

# word_vec_size need to match with the pretrained embeddings dimensions
#word_vec_size: 300
```