ADDING SIMPLICITY TO TRANSLATION

USING COMPRESSED TEXTS AS A PIVOT FOR NEURAL MACHINE TRANSLATION BETWEEN SIGN AND SPOKEN LANGUAGE

MICK FOPPELE

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
AT THE SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
OF TILBURG UNIVERSITY

WORDCOUNT: 8729

STUDENT NUMBER

2003181

COMMITTEE

dr. Dimitar Shterionov
dr. Henry Brighton

LOCATION

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science &
Artificial Intelligence
Tilburg, The Netherlands

DATE

January 14, 2022

ACKNOWLEDGMENTS

The completion of this thesis would not have been possible without the help of dr. Dimitar Shterionov and the opportunity to use the GPU infrastructure of Tilburg University.

ADDING SIMPLICITY TO TRANSLATION

USING COMPRESSED TEXTS AS A PIVOT FOR NEURAL MACHINE TRANSLATION BETWEEN SIGN AND SPOKEN LANGUAGE

MICK FOPPELE

Abstract

Sign language translation generally consists of two translation systems. A Video-to-Gloss translation system and a Gloss-to-Text translation system. This paper describes a new method for Gloss-to-Text translation. We propose a pipeline that consists of a MTS that translates the glosses received from a sign recognition system, to a compressed text and a MTS that translates the compressed text to a full text functioning as a text expander. In this paper the target text will be written German and the source text will be German sign language (DGS). By compressing the target text of the first MTS in the pipeline, we get the target text closer to the source text, which could improve translation performance. The second MTS serves as a text expander, going from compressed to full text. Two types of neural networks will be compared, an LSTM and a Transformer. The data set used is a Deutsche Gebärdesprache (DGS) Corpus consisting of 59,035 sentences. This thesis finds evidence that neither the proposed pipeline of the LSTM nor the proposed pipeline of the Transfomer outperforms their respective baseline. This thesis also finds evidence that the type-token ratio and the moving average type-token ratio decrease after every machine translation, indicating that MTS's struggle to generate diverse output.

1 INTRODUCTION

Sign languages are natural languages used by deaf people and people who suffer from muteness to communicate with others (Sandler & Lillo-Martin, 2006). *"Sign languages are used for everything that spoken languages are - within the family circle, for social interaction, oratory, education, scientific exchange, introspection and dreaming, story-telling, theatre, and poetry"* (Sandler & Lillo-Martin, 2006). Sign languages are not only an important means of communication within the deaf community, but they are also gaining popularity by the public who have difficulty with hearing or communication

(Naresh, Visalakshi, & Satyanarayana, 2020). Globalization and digitization are increasing the amount and pace of interaction between different cultures and countries. This also applies to those who use sign language as their primary way to communicate. To ease the communication between people who are deaf or suffer from muteness and those who can hear and speak, the scientific community has been trying to create ways to automate the translation of sign language to text via machine translation systems (MTS's). These translation systems are generally not of high quality. Those that achieve a high accuracy are often domain specific systems. Every increase in performance, be it speed or accuracy, is useful.

Given a video recording of a message in sign language, the automated translation of sign language to spoken language is a process that involves two major steps. The first step would be the recognition of signs and the translation of these signs into glosses. In this context, glosses are words that correspond to the core meaning of a sign (Konrad et al., 2020). Glosses can be seen as the labels of a certain sign or combination of signs. The second step would be the translation of these glosses to full text. We define the spoken text as shown in the 'Spoken' column of Figure 6 of the appendix [7] as 'full text'. Moryossef, Yin, Neubig, and Goldberg (2021) defines the first step as Video-to-Gloss and the second step as Gloss-to-Text. Both of these steps are processed by a different system. This thesis will have a focus on Gloss-to-Text, hence we will only describe this specific system in more detail. In Gloss-to-Text translation, a MTS trained on a sign language lexicon and a full text lexicon needs to be built. This MTS will then be able to translate glosses to full text. The model of Mehdi and Khan (2002) has been adapted and is shown in Figure 3 of the appendix [7]. This Figure gives a clear structure of the entire process of converting sign language to speech. Boxes are added to show where the Video-to-Gloss and Gloss-to-Text steps occur in this process. A similar format could be applied when converting German sign language, or Deutsche Gebärde Sprache (DGS), to text. In this case, the MTS is trained on a DGS lexicon and a full German text lexicon. Since this paper has a focus on Gloss-to-Text translation, we define a MTS translating glosses directly to full text as an 'end-to-end' translation system. By this definition, the MTS used in the Gloss-to-Text section of Figure 3 in the appendix [7] is an end-to-end translation system.

This paper proposes an extra step in the translation process of Gloss-to-Text. The proposed pipeline consists of a MTS that translates the glosses received from the recognition system, to compressed texts and a MTS that translates these compressed texts to full text. The compression of texts or sentences can be defined as the natural language processing task of abbreviating or shortening the original text corpus while retaining its

underlying meaning and information (Mandya, Nomoto, & Siddharthan, 2014). Compression will be described in more detail in Section 2.1. Using the format of Mehdi and Khan (2002), our pipeline could be displayed as shown in Figure 4 of the appendix [7]. By implementing compression we get the full text closer to its sign language counterpart which could improve the translation process. The inclusion of this step results in the following research question:

> *How does a pipeline of machine translation systems trained on sign language glosses, compressed texts and spoken texts differ from an end-to-end machine translation system for the task of sign to spoken language translation in terms of performance?*

To see whether the inclusion of this extra MTS will actually increase the accuracy of the Gloss-to-Text translation process, we create a baseline using the format of Figure 3 the appendix [7]. To substantiate the main research question we create the following sub-questions:

SRQ1 *How does the performance of a machine translation system pipeline for Gloss-to-Text translation based on LSTM compare to a machine translation system pipeline based on Transformer?*

SRQ2 *To what extent do specialized hyper parameters for spoken language translation with a low resource data set transition to sign language translation with a low resource data set when used in a pipeline and how do these models compare to an end-to-end translation system?*

SRQ3 *How does the lexical diversity of the training data correlate with the end performance of the machine translation system pipeline?*

The first sub-question has a focus on which type of neural network will be used to build the MTS's within the pipeline and the baseline. Which neural networks are taken into consideration will be described in more detail in Section 2.2. The second sub-question has a focus on hyper parameters. We will use the current literature based on low resource spoken to spoken machine translation and adapt these hyper parameters for our models. The literature and parameters are described in more detail in Section 2.4. The third research question has a focus on the correlation of the lexical diversity of the training data with the performance of the MTS pipeline. The concept of lexical diversity and its use will be described in more detail in Section 2.5.

This paper shows that the inclusion of an extra MTS in both methods does not increase the translation accuracy in the field of sign language translation. Other insights include the inability of MTS's to generate diverse

output, the impact of different evaluation metrics and the importance of pre-processing in sign to spoken language translation.

This paper is divided into several sections. Section 2 gives an insight into the relevant literature. The methods, implementation of these methods and the data used to answer the research questions are described in Sections 3 and 4. In Section 5, the results are presented. These results are further discussed in Section 6. In this Section the limitations of this paper are also mentioned. Finally, Section 7 outlines the main conclusions and recommendations for further research.

2 RELATED WORK

A considerable amount of research has been done in the field of automated sign language translation. However, previous research has been mainly focused on using end-to-end translation systems as defined in Section 1. We cover the current literature to each specific topic mentioned in Section 1.

2.1 *Text simplification and compression*

Text simplification is a natural language processing task of simplifying the grammar and structure of a text corpus while retaining its underlying meaning and information (Mandya et al., 2014). Text can be simplified in different ways. One way could be compression. As mentioned before, compression can be defined as the natural language processing task of abbreviating or shortening the original text corpus while retaining its underlying meaning and information (Mandya et al., 2014). Even though compression can be seen as a form of simplification, we will define simplification and compression as two different methods: (i) 'simplification' performs lexical simplification and (ii) 'compression' performs sentence compression (Mandya et al., 2014).

Applications of text simplification and compression are, often but not exclusively, seen in the medical domain. Medical texts are generally difficult to understand, since its target audience are highly-skilled professionals who are prone to use complex language and jargon (Van den Bercken, Sips, & Lofi, 2019). Simplification helps the audience understand a text which would normally require an expert intermediary to understand. Simplification has other uses as well, it can broaden the literature for children, non-native speakers and non-expert readers (Sun, Jin, & Wan, 2021).

Data, or text, compression offers an efficient and reliable way of communication. With current innovations such as Internet of Things (IoT) that provide a continuous connectivity, the amount of data grows exceptionally

(Vaerenbergh & Tourwé, 2021). The rate of growth of data being a lot higher than the rate of growth of technologies makes reduction of data volume in both storage as well as transmission desirable (Uthayakumar, Vengattaraman, & Dhavachelvan, 2018). Both simplification and compression can be applied at the same time (Nisioi, Štajner, Ponzetto, & Dinu, 2017). This type of combination will not be applied on the source sentences for the MTS in this paper due to time limitations. The implementation of simplification and compression is often seen as the main natural language processing (NLP) problem as shown in the examples above. In this paper the we will apply compression as a form of pre-processing on our source data.

2.2 *Machine Translation Systems*

Recent work (Bawa, Kumar, et al., 2021) provides a detailed overview of all the different types of MTS's. Both Bawa et al. (2021) and Östling, Scherrer, Tiedemann, Tang, and Nieminen (2017) show that neural networks currently achieve state-of-the-art results in machine translation. Junczys-Dowmunt, Dwojak, and Hoang (2016), Östling et al. (2017) and Shterionov et al. (2018) prove that neural networks, or neural machine translations systems (NMT), outperform Statistical Machine Translation systems (SMT). Based on these results our focus for building the MTS will be on neural networks. It is to be noted that in the research of Shterionov et al. (2018), the NMT only outperformed statistical machine translation when evaluated by human reviewers. Quality evaluation scores indicate that the statistical machine translation engines perform better. We also want to note that this paper will not use human evaluation due to time and budget limitations.

For machine translation, architectures based on recurrent neural networks (RNN) with attention (Bahdanau, Cho, & Bengio, 2014) have achieved state-of-the-art results since its creation until the introduction of the Transformer (Vaswani et al., 2017). Both approaches are currently being widely employed for building academic as well as commercial MTSs. For this paper, we will compare RNN- and Transformer-based approaches. The basic 'vanilla' RNN (Rumelhart, Hinton, & Williams, 1986) risks both the exploding and vanishing gradients problems (Xu et al., 2020). To fix this problem two new approaches were introduced: (i) Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and (ii) Gated Recurrent Units (GRU) (Cho et al., 2014). Xu et al. (2020) compares the performance of the LSTM and GRU for natural language processing of online reviews. They mention that GRU has one less gate than LSTM which effectively reduces its computation time. The simpler structure of GRU can save a lot of time without sacrificing much performance. When looking at their

results we can see that GRU only outperforms LSTM on a small data set containing long sentences. Since our data set (see Section 3.1) is small with relatively short sentences and computation power should not really be a bottleneck due to the infrastructure provided by Tilburg University, we will opt for LSTM. The infrastructure of Tilburg University has 4 Dell PowerEdge R750 servers. Each server hosts 192GB of RAM with 2 CPUs and 2 GPUs. The CPU's are Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz as mentioned on the site of Tilburg University[1].

As mentioned before, the other approach is the Transformer (Vaswani et al., 2017). The Transformer attains a better BLEU score (Papineni, Roukos, Ward, & Zhu, 2002) than other state-of-the-art models (Vaswani et al., 2017). The combination of the vanishing and exploding gradient problem and the usage of attention-based algorithms often leads to the Transformers outperforming RNN's (Lankford, Alfi, & Way, 2021). Since a Transformer generally requires a lot of data (Lankford et al., 2021; Xu et al., 2020), the LSTM seems to be better suited for a low resource data set. Training machine translation models on pre-processed data has been done before. Mehta et al. (2020) provides proof that models with pre-processed source sentences, in this case simplified sentences, lead to better performance compared to models without pre-processed source sentences. This paper could fill the gap in whether these results could be extended when applying compression in the field of sign language translation.

2.3 *Gloss-to-Text Translation*

Gloss-to-Text translation systems (e.g. DGS to German) have a similar approach as the traditional Text-to-Text translation systems (e.g. English to German) mentioned in Section 2.2. Both translation systems are trained on source and target sentences. The essence is to learn the relationships of the word combinations between both sentences. However, Gloss-to-Text translation systems have an inherent problem. According to Moryossef et al. (2021), when matching a sign language to its spoken counterpart, there seems to be a lower amount of syntactical similarity when compared to the matching of two spoken languages. Moryossef et al. (2021) defines lexical similarity as the overlap of words between multiple lexicons. It is to be noted that in this case, syntactic similarity as defined by Moryossef et al. (2021) can be seen as a form of semantic similarity defined by Harispe, Ranwez, Janaqi, and Montmain (2015) as the similarity of different entities when looked at their semantics, i.e. their meaning. Harispe et al. (2015) gives the clear example of the words tea, toffee and coffee. Both tea and coffee are hot beverages indicating a relatively high semantic similarity score.

[1] https://www.tilburguniversity.edu/current/news/more-news/gpu-computers

The words toffee and coffee look very similar, indicating a relatively high lexical similarity score. The inherent lower semantic similarity between sign and spoken languages decreases translation accuracy of Gloss-to-Text translation systems (Moryossef et al., 2021). By increasing this similarity we aim to increase translation performance.

2.4 *Hyper Parameters*

Well performing MTS's used in natural language processing are usually trained on massive amounts of data. These amounts often exceed millions of sentences to train (Zhang & Duh, 2021). Since sign language is not a spoken language, the costs of creating sample sentences with corresponding videos and glosses are high. As a consequence, sign language data sets rarely exceed 60k sentences. Compared to the earlier mentioned MT tasks, Gloss-to-Text can be defined as a low resource MT problem (Moryossef et al., 2021; Zhang & Duh, 2021). In order to improve natural language processing tasks done by MTS's, multiple suggestions are made to alter the optimized hyper parameters for general usage of these models (Lankford et al., 2021; Sennrich & Zhang, 2019). These suggestions are all based and optimized on low resource Text-to-Text translation tasks. Different types of neural networks require different hyper parameter tuning (Lankford et al., 2021). This paper could fill the gap to see whether the hyper parameter tuning proposed by Lankford et al. (2021); Sennrich and Zhang (2019) for low resource Text-to-Text translation tasks can also be applied in Gloss-to-Text translation tasks.

2.5 *Lexical Diversity*

Malvern, Richards, Chipere, and Durán (2004) mention that there is disagreement within the scientific field about the definitions of lexical diversity and vocabulary richness, which sometimes appear to be synonymous. O'Dell, Read, McCarthy, et al. (2000) defines vocabulary richness as a feature consisting of multiple components. These components include 'lexical variation', lexical sophistication', 'lexical density', and 'number of errors'. Lexical variation, which can be seen as lexical diversity, is here defined as 'the range of vocabulary and avoidance of repetition'. The measure for lexical diversity as defined by O'Dell et al. (2000) is traditionally the type-token ratio (TTR), which can be calculated by comparing the number of different words with the number of total words within a sentence (given that the TTR is computed on a sentence level). This paper will use two different methods for lexical diversity: (i) the basic TTR as defined by

O'Dell et al. (2000) and (ii) the moving average TTR (MATTR) as defined by Covington and McFall (2010).

The loss of lexical diversity has been used as a feature to estimate the quality of MTS's before (Vanmassenhove, Shterionov, & Way, 2019). Since neural networks show an inability to generate diverse output and there appears to be a loss of lexical diversity caused by machine translation (Vanmassenhove et al., 2019), we want to see whether the reduction in lexical diversity impacts the performance of our proposed pipeline by measuring the correlation of the lexical diversity metrics and the performance metrics.

3 METHODS AND METHODOLOGY

Besides the usage of the earlier mentioned LSTM and Transformer methods, we also apply forms of pre-processing and compression on the data. Since compression is a big part of the methodology we apply, it will not be discussed in the pre-processing section but it will have its own separate subsection.

Figure 8 of the appendix [7] shows a flowchart of the entire methodology for the baseline model. The flowchart uses the symbols as defined by Hebb (2012). Both texts are color coded, indicating which text becomes the source text and which text becomes the target text. Figure 9 of the appendix [7] shows a flowchart of the entire methodology for the pipeline model. The same symbols and color coding is applied for the pipeline. In the pipeline flowchart we excluded the compression phase from the pre-processing process to emphasize this phase. It is to be noted that in the pipeline model, the second MTS uses the output of the first MTS as source text. The second MTS is then trained on this output and the target data set of the baseline model. All of the steps shown in Figures 8 and 9 of the appendix [7] are discussed in detail in Section 3.1.2.

3.1 *Data*

The data sets are (DGS) corpora provided by Konrad et al. (2020). We obtained the data sets via the GitHub page of Gijs Thissen[2] who performed a study based on gloss to text translation. The data sets are a watered-down version of the DGS public corpus. The data we use is a merged set that originally consisted of 359,130 and 374,411 fragments of sign language recordings associated with single-gloss annotations. After merging the data sets into one data set and organising the glosses into sequences, it con-

[2] https://github.com/GijsThissen/g-thissen-csai-thesis.

stitutes 59,035 instances. The data set contains timestamps and mouthing signs. Due to time limitations we will not be using these attributes.

3.1.1 *Annotation and Glossing Conventions*

Project notes are provided by Konrad et al. (2020) to get a better understanding of the data. Since sign language is its own natural language it is important to get an understanding of the grammar and spelling of the signs (source) glosses within the scope of this research. Some of the underlying meaning or the combination of signs and mouthed words goes beyond that scope. For example the DGS sign 'SQUARE1$\hat{}$', which is basically signing a square shape with both hands, is frequently used to cover words or meanings such as 'square', 'page', 'letter', 'recipe', or 'map'. The meaning of 'SQUARE1$\hat{}$' depends on the mouthed words as well. Changing the mouthed words can change the meaning of 'SQUARE1$\hat{}$' to 'newspaper', 'visa', 'television', or 'stole' (Konrad et al., 2020). For some glosses the meaning is derived from context. This could be the core reason for semantic dissimilarity between sign and spoken languages as mentioned by Moryossef et al. (2021).

There are multiple types of glosses within the DGS corpus. These glosses are linked in a parent and child relationship. All parent type glosses are followed by a circumflex ($\hat{}$) (e.g. $ORAL$\hat{}$). Glosses without circumflex represent child types. This parent child relationship is defined as: X is the parent type. Y is the child type. Y is a subclass to X. In the DGS corpus these relationships correspond to the type-token matching of signs and mouthed words within the project data set. Since we do not use the mouthed words the circumflex derives no direct meaning in this paper. Among the sign language glosses there are multiple child groups distinguished using a prefix. In the DGS corpus they use a dollar sign followed by the prefix to group the child groups. The types occurring in this data set are described in the bullet points below:

- NAME SIGNS ($NAME)

Sign language makes use of name signs. All names included in this data set are defined by '$NAME'. Since the data set is based on real people all names are kept '$NAME' in order to preserve privacy. For public or well-known individuals the prefix is normally followed by the name (e.g. $NAME-JOE-BIDEN1), however the public DGS corpus does not make this further differentiation and only uses the parent class '$NAME'.

- PRODUCTIVE SIGNS ($PROD)

Within sign language there is an interaction between lexical signs (telling something) and polymorphemic signs (showing something). Polymorphemic, which are called productive signs in the DGS corpus, are small signs built on conventional and non-conventional elements (Konrad et al., 2020). The meaning of these productive signs can only be derived via context. This makes classification and translation of such signs extremely hard. Especially since this DGS is a low resource data set and this type of classification would require a lot of data.

- POINTING SIGNS ($INDEX)

Pointing signs look a lot like pointing gestures. Unlike the earlier mentioned productive signs, pointing signs normally do have a limited range of meanings. In the DGS Corpus the pointing signs are glossed by '$INDEX' followed by a number. Some examples of this would be '$INDEX4' which indicates the signer is using a thumb to sign or '$INDEX-ORAL1' where the signer points to his or her lips. The latter is often used to direct focus on a mouthing sign. Pointing signs are complex since they can be used to sign objects in the area, which again need to be derived from context.

•FINGERSPELLING ($ALPHA)

In DGS, the spelling of words or denoting a single letter is done via fingerspelling. All fingerspelling signs are labelled as '$ALPHA^'. This parent type contains multiple child types such as '$ALPHA1' and '$ALPHA2' which represent one-handed and two-handed signs respectively. For example the gloss related to the one handed signing of the letter 'L' would be '$ALPHA1:L'. Cities, villages or names consisting of a few letters such as 'Iglu' will sometimes be spelled completely. This would be glossed as '$ALPHA1:I-G-L-U'. Note that this example is spelled using one hand.

• INITIALISATION ($INIT)

Initialisation is the process of denoting the first letter of a word. In DGS this is done using the fingerspelling signs ('$ALPHA') we mentioned before. The inclusion of a tilde symbol (~) before a gloss indicates that the gloss is derived from the same type of sign but with a changed parameter (Konrad et al., 2020). This type of sign is often implemented when the signer does not know the regular sign. The deriving of these glosses could trouble the translation process since the machine translation systems need to learn a lot more relations.

• NUMBER SIGNS ($NUM)

In order to communicate large numbers in sign language there needs to be an efficient system in place. In the DGS corpus a '$NUM^' parent class is created. This parent class is again split up in multiple child types (Konrad et al., 2020):

1-10 ($NUM-ONE-TO-TEN1A etc.),
11-19 ($NUM-TEEN1 etc.),
10, 20...90 ($NUM-TENS1 etc.),
100, 200...900 ($NUM-HUNDREDS1),
1000, 2000...10,000 ($NUM-THOUSANDS1).

The project notes gives the example of 1989. This would be segmented into three tokens: $NUM-TEEN1:9 (nineteen), $NUM-ONE-TO-TEN1A:9 (nine), $NUM-TENS1:8d (eighty). For repeated digits there is a special child type of the '$NUM^' type (i.e. '$NUM-DOUBLE1A'). Due to the difference in format used to describe numbers we foresee some trouble in translating numbers. We tried to prevent this problem by writing an algorithm that changes the glossed numbers of the source sentences into integers. However, the development and implementation of such an algorithm would take too much time hence we use the glosses provided by the DGS corpus as input.

• LIST BUOYS ($LIST)

In the DGS corpus enumerating items to a list is communicated by pointing the index finger to the finger of the other hand. The function called 'list' has multiple functions. It can be used to actually make a list of e.g. politicians or colors, but it can also be done to simply enumerate numbers. These signs are glossed in the '$LIST' type. The glosses in the data set get more detailed since it is also possible to show the total items within the list by yet another sign. The fact that such a gloss has that many different meanings makes the training of a MTS harder.

• GESTURES ($GEST)

Other signs that rely a lot on context are gestures. These types of signs are usually unconventional. An example of a gloss labelled as gesture is '$GEST-OFF$^{\hat{}}$'. This is a sign made with the hand palm up and the hand open. The German word 'offene' means 'open'. These types of glosses are often really specific and indicate a movement not visible in either the source or target sentence. The MTS does not see the video recording, only '$GEST-OFF$^{\hat{}}$' and the corresponding target word.

• MOUTHING ($ORAL)

Some words in DGS are purely based on mouthed words, also known as lip reading or speech reading. These signs are glossed as '$ORAL$^{\hat{}}$'. These glosses are linked to a mouthing sign. As mentioned in Section 3.1, we will not be using these signs. The choice to skip this attribute will decrease the performance of our MTS.

The quality and complexity of this data will reduce the performance of our MTS. However, since all the models are trained on the same data, we can still compare their respective performance.

3.1.2 *Pre-processing*

The data sets will be loaded into Python for processing. Google Colab[3] will be used as a Python environment. The training of the systems is done using the GPU infrastructure of Tilburg University. As mentioned in Section 3.1, the data set consists of many fragments. These are originally ordered as shown in Figure 6 of the appendix [7]. In order for the MTS to be trained, the glosses need to be aligned with their respective target sentence as shown in Figure 7 of the appendix [7]. The alignment of the sentences resulted in many empty or NaN values in the source data. All

[3] https://colab.research.google.com/

these values are removed using own code. This code will be freely available on the related GitHub repository of this paper[4].

On a character level we see some inconsistencies. In the example shown below we see that in the source sentence the word Fußball is spelled with 'ss' while in the target sentence the word is spelled with 'ß'.

> *Source (sign) = WEIT1 ICH1* GEHEN1A* MÖGEN3 ICH1* FUSS-BALL2* SPIELEN2*
> *Target (spoken) = Das war so weit. Ich wollte lieber Fußball spielen.*

In the German language both types of spelling are correct. In order for us to maintain consistency, we change all the 'ß' characters to 'ss'. We choose this option due to the brevity of the code. Replacing 'ß' with 'ss' will not result in errors, whereas the replacement of 'ss' by 'ß' in, for example, the word 'wissen' will result in faulty spelling. This process is done using own code. Before removing the ß from the data we check to see whether the gloss '$ALPHA1:ß' or '$ALPHA1:SS' is present in the data set in order to prevent bad translation. Neither '$ALPHA1:ß' nor '$ALPHA1:SS' is present in our data.

After formatting and fixing inconsistencies, the first step in natural language processing is tokenization (Webster & Kit, 1992). By tokenizing a piece of text we make each word or stand alone character (including punctuation marks) a token. Since sign language does not use punctuation marks, and all glosses are already split, the source files are already tokenized. An example of why tokenization is necessary for the target files is the end of a sentence. This can have a punctuation mark. To make sure that these characters are not added to the last word of the sentence, and thus creating a new word, the tokenization algorithm splits this in two tokens as shown in the example below.

> *Pre-tokenized = #Name4 hat mir gesagt, dass sie austreten will, weil ihr Mann gestorben ist.*
> *Tokenized = # Name4 hat mir gesagt , dass sie austreten will , weil ihr Mann gestorben ist .*

The example also clearly shows how this algorithm deals with commas. The implementation is done using a tokenization script[5]. The provided script also cleans the data. After the tokenization and cleaning, we will use a train_test_dev.py script[6] to split the data into six subsets:

[4] https://github.com/MFoppele/ThesisDSS
[5] This script is provided by dr. Dimitar Shterionov
[6] This script is provided by dr. Dimitar Shterionov

-train.src	-train.trg
-test.src	-test.trg
-development.src	-development.trg

The src extension is an abbreviation for 'source' and the trg extension is an abbreviation for 'target'. The test and development subsets contain 1000 randomly selected sentences each. The remaining sentences will be used for training the models. The same process is done for the compressed data sets. This results in the following six subsets:

-train.src	-train_compressed.trg
-test.src	-test_compressed.trg
-development.src	-development_compressed.trg

We opt for this split due to the small amount of data as mentioned in Section 3.1. The train_test_dev.py script used to split the data creates source files with empty lines inside. The MTS interprets these lines as sentences and starts assigning sentences to empty lines. Hence we need to remove the empty lines. This is done using own code. In order to remove these lines we need to remove the corresponding target lines as well, otherwise the src and trg files are not aligned anymore. The removal is done using a loop based on the indices of the src file. These are simply put in a list. Then the list of indices is used to remove the sentences in the trg file. Due to the nature of a for loop, the indices list is looped over in reverse to prevent us from removing the wrong index.

The final step of pre-processing is to implement a variation of the Byte-Pair Encoding (BPE) algorithm as introduced by Gage (1994) on all the different data sets. BPE is used to split each word into sub-words (smaller parts of a word) which reduces the vocabulary used to train the NMT. When considering the English words *'taller'* and *'lowest'*, the BPE algorithm will split *'taller'* into *'tall'* and *'er'* and *'lowest'* into *'low'* and *'est'*. When the NMT models learn from the data, they will learn not *'taller'* and *'lowest'*, but they will learn the sub-words *'tall'*, *'er'*, *'low'* and *'est'*. With this new vocabulary the NMT models can recognize (and translate) *'tall'*, *'taller'*, *'tallest'*, *'low'*, *'lower'*, *'lowest'*. A big advantage of BPE is that the NMT models are now able to recognize more words then there are available in the original data set. The NMT models are now able to recognize the adjective, comparative and superlative. Even when only the word 'small' is in the data set, 'smaller' and 'smallest' will be recognized as well. The same principle applies for morphological derivation (Sennrich, Haddow, & Birch, 2015). Morphological derivation is the process of forming a new word by adding a prefix of suffix (Dixon, 2014). Considering the root word

'happy', a prefix (e.g. *'un-'*) could be added to create a new word *'unhappy'*. A suffix (e.g.-*ness*) could be added to create the word *'happiness'*.

It is to be noted that the mentioned BPE examples above are very idealized. Due to the small data set used in this paper, the BPE algorithm evidently has a low amount of sub-words as training input. As a consequence, there might be low lexical cohesion within the data which impedes the BPE algorithm.

3.2 *Compression*

The form of simplification used on the target sentences of the first MTS in the pipeline for both the LSTM and the Transformer will be compression as defined in Section 2.1. This form of simplification is the least computationally expensive and the easiest method to implement. Due to time limitations, compression seems to be the best approach in this NLP problem. Given that we use a 'Deutsche Gebärdensprache' (DGS) corpus and German sign language does not have copula words (Pfau & Quer, 2008), we choose a compression method which removes stop words as defined by the nltk library[7] (e.g. *'ich'*, *'weil'*, *'wir'*, *'deiner'*) as a proxy for copula words. By removing the amount of words in a spoken language, the semantic similarity should increase compared to a sign language. The compression is applied in Python by removing the nltk stopword list from the random assigned target train, test and development sets. We will apply compression before the split algorithm and the letter inconsistency algorithm to keep the evaluation pairwise and reduce computation time. The Evaluation of the compression algorithm, defined as the compression ratio, will be discussed in more detail in Section 4.4.1.

4 EXPERIMENTAL SETUP

Both LSTM and Transformer are implemented using early stopping (Prechelt, 1998). The value is set to 10, this indicates that if the validation perplexity does not decrease for 10 consecutive steps the training process is stopped. Early stopping is done in order to prevent overfitting.

The implementation of the second MTS in the pipeline requires some additional processing of the data. The output files of the OpenNMT framework (Klein, Kim, Deng, Senellart, & Rush, 2017) gives metadata per sentence as shown in the example below:

[2021-11-24 11:10:00,047 INFO]
SENT 6: ['ÜBER1̂', 'ERZÄHLEN4', '$INDEX1*']*

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

PRED 6: Das hat also nichts erzählt .
PRED SCORE: -3.5319

The removal of all the metadata is done using own code based on index number.

4.1 *Implementation of LSTM*

The LSTM is implemented using the OpenNMT framework (Klein et al., 2017). Both the decoder and encoder are RNN's of type LSTM. We implemented the hyper parameters optimized for general use. Besides hyper parameter tuning, Sennrich and Zhang (2019) mention other improvements such as a reduced vocabulary and the implementation of a lexical model introduced by Nguyen and Chiang (2017). The tuning of hyper parameters is done based on the changed vocabulary and the lexical model. However, the biggest improvements in accuracy are obtained when the RNN and CNN architectures are completely replaced with an attention mechanism, thus creating a Transformer (Lankford et al., 2021). We are focused on the hyper parameters and not the inclusion of the additional steps mentioned by Sennrich and Zhang (2019). Due to time limitations and the scope of this paper, we will only test a tuned version of the Transformer. As shown in the results, the LSTM does not have a second run.

4.2 *Implementation of Transformer*

The Transformer is implemented using the OpenNMT framework (Klein et al., 2017) as well. In the first run we implement the hyper parameters optimized for general use. Based on the optimized hyper parameters for low resource data sets mentioned by Lankford et al. (2021), our hyper parameter values are shown in Table 2. Values that differ in run 2 are indicated in bold. It is to be noted that according to Lankford et al. (2021), both 2 and 8 attention heads could work. In their results they mention that a model trained on 55k lines favours 2 attention heads. Since the volume of our data is closest to the benchmark of 2 attention heads we opt for 2 heads as well.

Hyper parameter	run_1	run_2
Learning rate	2.0	2.0
Batch size	4096	**2048**
Attention heads	8	**2**
Feed-forward dimension	2048	2048
Embedding dimension	512	**256**
Label smoothing	0.1	0.1
Dropout	0.1	**0.3**

Table 1: Transformer optimization.
Changed hyper parameters in run 2 are shown in bold

4.3 *Implementation of TTR and MATTR*

In order to check for the correlation of lexical diversity and the MTS pipeline performance, we will measure the lexical diversity of the target test file, the output of MTS 1 and the output of MTS 2 as displayed in Figure 5 of the appendix[7], where LDT denotes Lexical Diversity Test. Using a high interval of tests we can see if lexical diversity is positively or negatively correlated with the performance of the MTS pipeline compared to the baseline. Both the basic TTR and the MATTR are calculated using the lexicalrichness package[8]. The TTR and MATTR will only be calculated on the test sets. We want to calculate these measures for the output of the first MTS and second MTS in the pipeline. Since translating the train data will result in a bias, only the test data will be translated. Hence this set is used in every LDT to maintain consistency.

4.4 *Evaluation metrics*

In order to measure the amount of compression applied on the data, we will use a compression ratio metric provided by Alva-Manchego, Martin, Scarton, and Specia (2019). The compression ratio will be described in more detail in Section 4.4.1. To evaluate the quality of the MTS's, we use BLEU (Papineni et al., 2002) and TER (Snover, Dorr, Schwartz, Micciulla, & Makhoul, 2006). Since both metrics have the same purpose they are described in the same section. These evaluation metrics for performance are described in more detail in Section 4.4.2.

[8] https://pypi.org/project/lexicalrichness/

4.4.1 *Compression ratio*

The compression ratio is our metric to evaluate the amount of compression. It is defined by Alva-Manchego et al. (2019) as *"The number of characters in the simplification divided by the number of characters in the original sentence."* All individual characters (including spaces, periods, commas etc) are used in this calculation. This is done using own code. The compression ratio of each sentence will be appended to a list. Then the average compression ratio will be calculated over this list. In our data set, this resulted in a compression ratio of 0.7010. Due to the nature of this metric, tokenization is not necessary to compute the compression ratio. The decision to compress the sentences before splitting is not impacting the compression ratio.

4.4.2 *Bilingual evaluation understudy (BLEU) and Translation Edit Rate (TER)*

The bilingual evaluation understudy (BLEU) is an algorithm that helps to evaluate automatic machine translation with the central idea of '*The closer a machine translation is to a professional human translation, the better it is*' (Papineni et al., 2002). We choose this metric due to its accurate depiction of translation quality and its extensive use in the field of natural language processing.

We will also evaluate the Translation Edit Rate (TER), defined as '*the amount of editing that a human would have to perform to change a system output so it exactly matches a reference translation*' (Snover et al., 2006). TER seems to yield a higher correlation to human judgement than BLUE (Snover et al., 2006), hence we include this evaluation metric as well.

Both BLEU and TER will be calculated using the sacrebleu package[9]. The output files of the OpenNMT package are not suited for the sacrebleu script. As mentioned before, the output files of the OpenNMT gives metadata per sentence. This metadata alters the BLEU score. Since we are only interested in the predicted (translated) lines we need to filter these lines out. This is done using the same code used to prepare the data for the second MTS in the pipeline. This code will be freely available on the related GitHub repository of this paper[10]. The sacrebleu script will be applied on the predicted sentences and test.trg files.

5 RESULTS

In Section 5.1 the BLEU and TER scores will be described for the first run of the LSTM and Transformer as well as the second run for the Transformer.

[9] https://github.com/mjpost/sacrebleu
[10] https://github.com/MFoppele/ThesisDSS

All results of the final output will be shown in Table 2. The 'NA' in Table 2 and Table 3 indicates that there is no score for that model.

In Section 5.2 the TTR and MATTR are described for LDTs. The values for the 'DGS corpus normal' and 'DGS corpus compressed' are identical in run 2 simply because the same data sets are used in both runs. The 'NA' in Table 4 indicates that there is no score for that model.

5.1 *Performance*

Model	run_1 (BLEU/TER)	run_2 (BLEU/TER)
LSTM Baseline	1.2025 / 98.0305	NA
LSTM Pipeline	0.4807 / 102.2522	NA
Transformer Baseline	2.4522 / 97.6840	**2.6023 / 95.5138**
Transformer Pipeline	0.2236 / 101.2857	**0.4965 / 100.8480**

Table 2: Results of run_1 and run_2.
The better performing run is highlighted in bold.

For both the LSTM as well as the Transformer, we see that the baseline outperforms the pipeline significantly on BLEU score and TER score. Even after fine tuning the hyper parameters of the second run of the Transformer, the pipeline did not outperform the baseline.

When comparing the baselines of the two different neural networks in run 1, we see that the Transformer outperforms the LSTM in terms of BLEU score. The difference in the TER score is still in favour of the Transformer but the difference has become relatively smaller. This is a result that contradicts prior research and our expectations since an LSTM generally performs better on a low resource data set compared to a Transformer (Xu et al., 2020). The LSTM pipeline outperforms the Transformer pipeline based on BLUE score but not on TER score. This indicates that both the TER and BLEU evaluation metric can favour a different model in terms of performance.

The adaptation of the hyper parameters for Transformers trained on low resources data sets defined by Lankford et al. (2021) seems to improve the performance of our Transformer as well. Both the BLEU and the TER score improve in the second run compared to the first run. This is in line with the findings of Lankford et al. (2021). The difference in increase between the pipeline and baseline is minimal, about 0.1501 BLEU for the baseline and about 0.2729 BLEU for the pipeline. After tuning the Transformer, the Transformer pipeline of the second run outperforms the LSTM pipeline of the first run. Where the BLEU score seems to increase a lot (relatively

speaking), the TER score seems to improve marginally. In Figure 1 all the models are ordered from best performing to worst performing based on BLEU score. B1 stands for Baseline run 1, B2 stands for Baseline run 2, P1 stands for Pipeline run 1 etc.

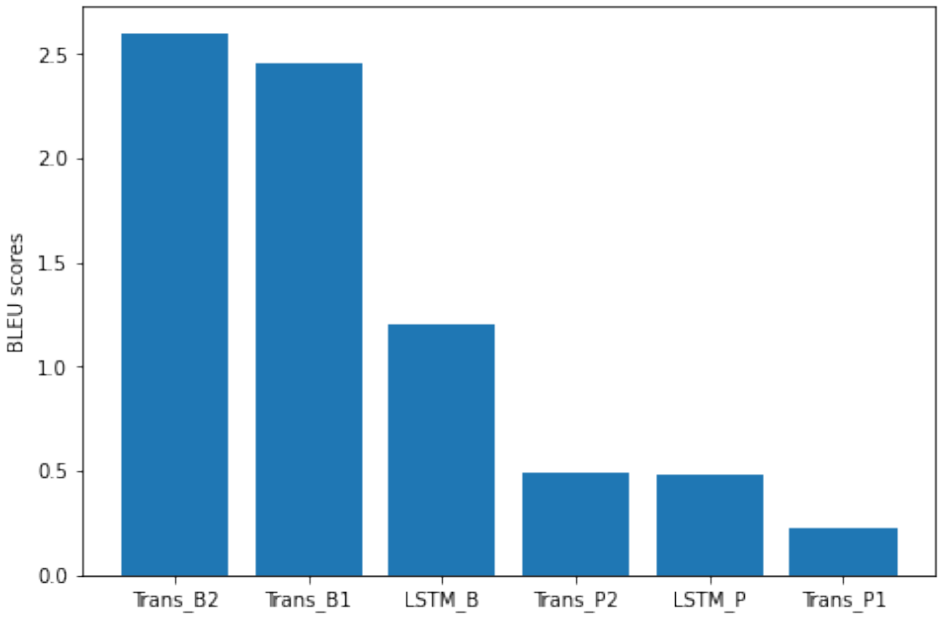

Figure 1: BLEU scores all models

In the first run, the first MTS in the LSTM pipeline, achieved a BLEU score of 0.8535 and a TER of 97.6556. These scores are calculated based on the output of this particular system and the compressed test data set. For the first MTS in the Transformer pipeline, a BLEU score of 2.5663 and a TER of 96.8974 were achieved based on the same compressed test data set. In the second run, the first MTS in the Transformer pipeline achieved a BLEU score of 2.5943 and a TER of 95.5461 based on the same compressed test data set. The performance increase for this MTS compared to the first run is extremely minimal. Most of the performance increase is achieved at the second MTS. The results of each MTS in the pipelines are shown in Table 3.

MTS	run_1 (BLEU/TER)	run_2 (BLEU/TER)
LSTM 1	0.8535 / 97.6556	NA
Transformer 1	2.5663 / 96.8974	2.5943 / 95.5461

Table 3: Results of each MTS within the pipeline

The BLEU score of the first MTS in each pipeline does not indicate the accuracy of the full pipeline, hence these results are not shown in Table 2. We choose to measure this accuracy to see which MTS reduces or improves the accuracy of the pipeline the most and to calculate the correlation of BLEU and TER with TTR. When taking a closer look at the exact predictions we see that numbers are not translated correctly both ways. A number is sometimes translated as a word and a word is sometimes translated as a number.

The individual pipeline translation systems seem to train faster than the baseline translation systems. The pipeline translation systems use a reduced vocabulary to train which reduces the complexity of the translation system itself. However, it is to be noted that when using this as an argument for efficiency it does not hold since the pipeline uses two MTS's. Comparing the time used to train the full baseline and the full pipeline will give the edge to the baseline.

5.2 *TTR and MATTR*

Data	run 1 (TTR/MATTR)	run 2 (TTR/MATTR)
DGS corpus normal	0.249 / 0.932	0.249 / 0.932
Baseline LSTM output	0.089 / 0.829	NA
Baseline Transformer output	0.228 / 0.913	0.153 / 0.789
DGS corpus compressed	0.409 / 0.97	0.409 / 0.97
Pipeline LSTM output 1*	0.092 / 0.778	NA
Pipeline LSTM output 2*	0.008 / 0.605	NA
Pipeline Transformer output 1	0.401 / 0.961	0.192 / 0.74
Pipeline Transformer output 2	0.02 / 0.732	0.014 0.652

Table 4: Lexical Diversity of run 1 and run 2.
* Output 1 denotes the output of the first MTS in the pipeline, Output 2 denotes the output of the second MTS in the pipeline

Both the TTR and MATTR decrease with every translation step, which is in line with the claim of Vanmassenhove et al. (2019) that MTS's have an inability to generate diverse output. We see that the Transformer outperforms the LSTM drastically in both the baseline as well as the pipeline scenario based on TTR and MATTR. Both metrics decrease after every MTS, but it is remarkable that the Transformer baseline and the first model of the Transformer pipeline see a small reduction in these metrics. These models seem to be able to produce a more diverse output than the LSTM. The second MTS in both pipelines seems to favour certain sentences

and predict these sentences many times. A high amount of repetition, which means a low diversity in output, reduces the TTR and MATTR drastically.

The TTR and MATTR move in the same direction, just by other degrees. After computing Spearman's rank correlation between the both lexical diversity metrics and MTS evaluation metrics, we see that only the TER and TTR have a significant correlation of -0.8857 at $p < 0.05$. The other correlations and p values are shown in Table 4. Even though the combinations of TER/MATTR and BLEU/TTR are not significant at $p < 0.05$, it is interesting to see that they have the exact same correlation in opposite directions.

Combination	Correlation	p-value
BLEU and TTR	0.7714	0.0724
BLEU and MATTR	0.6571	0.1562
TER and TTR	-0.8857	0.0188
TER and MATTR	-0.7714	0.0724

Table 5: Spearman rank correlations

In Figure 2 the trends of the TTR within the pipelines are shown. The x-axis represents the MTS output where the origin denotes the TTR of the starting test set.

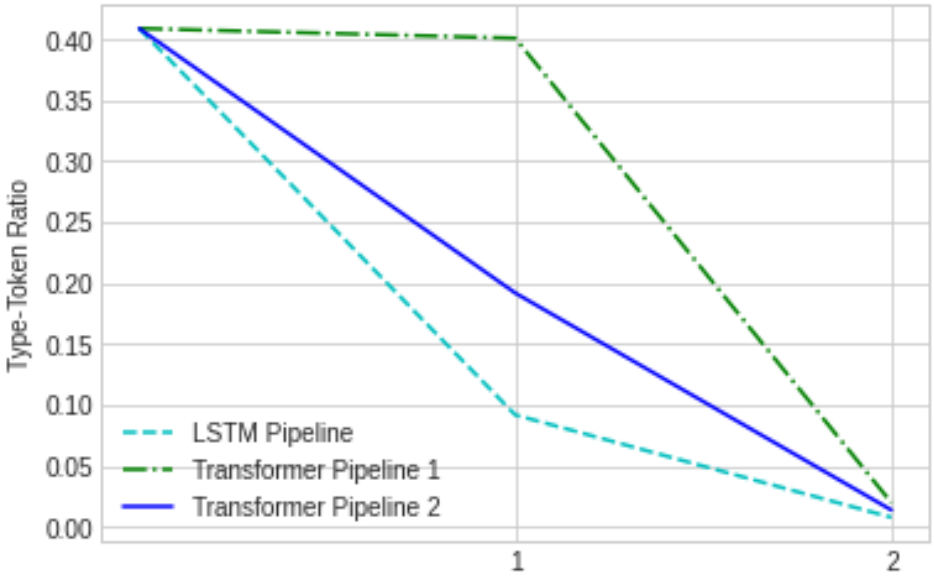

Figure 2: TTR all models

As shown in Figure 2 we see that, in contrast to the other models, the second run with the Transformer has an almost linear decrease across the pipeline. The difference between the first MTS in the pipeline of the first and second run is large but diminishes after the second MTS. The increased performance in the second run of the Transformer resulted in lower TTR and MATTR scores. This means that, even though the output of the system in run 2 is less diverse, it is more accurate. It seems that the Transformer with tuned hyper parameters performs differently than all the other models in terms of diverse output in combination with accuracy.

6 DISCUSSION

The goal of this paper is to see whether the creation of a translation pipeline by including an extra MTS trained on compressed data would improve the machine translation process. This is done via multiple sub questions consisting of different neural networks, different evaluation metrics for these networks and hyper parameter tuning of these networks. In the first subsection we will provide our interpretation of the results and in the second subsection we will acknowledge some limitations of this paper.

6.1 *Interpretations*

All baseline models (Transformer run 1, Transformer run 2 and LSTM run 1) outperform their pipeline counterpart, indicating that the inclusion an extra MTS in this format does not improve machine translation performance. It is to be noted that we expected rather low BLEU scores and high TER scores when comparing these models to operating machine translation models for spoken-to-spoken language. The low BLEU and high TER scores have multiple reasons. The inherent problem of Gloss-to-Text translation as mentioned by Moryossef et al. (2021) makes MTS's built for sign language less accurate than MTS's built for spoken-to-spoken translation. The quality and volume of the DGS corpus reduces performance even more. Besides these two problems based on data, there was a low amount of time and resources available when writing this paper. However, the goal of this paper was not to create an optimal performing MTS for sign language translation, but to make relative comparisons between the baseline and the pipeline.

Recent work has shown that current MTS's fail to generate diverse output (Vanmassenhove et al., 2019). These results are in line with our findings as mentioned in Section 5.2. This might be one of the reasons that a pipeline does not work. The more MTS's are included, the more repetitive the input data gets for the second MTS. The results of our pipelines show that the second MTS does get extremely repetitive in terms of output. The performance of the pipeline could perhaps be improved by creating a more advanced text expander. Models developed for text expansion could perhaps outperform a MTS used for text expansion. An improvement in this phase of the pipeline could have big impact since the second MTS in the pipelines seems to perform poorly. However the first MTS's in the pipelines already underperforms compared to the baseline. MTS's capable of generating a more diverse output could improve the performance of the pipeline since they reduce the chance of repetition in the second MTS.

The second MTS could, in its current form, be improved by changing the volume of its training data. In this paper we trained the second MTS on the same data volume as the first MTS. In contrast to the first MTS is the second MTS, which is used as a text expander, not limited to glossed data. The second MTS could be trained on a data set consisting of millions of sentences. Such a data set could be created by compressing a massive amount of German sentences using own code. As mentioned before this improvement of the text expander could improve both performance and output diversity of the pipeline.

The LSTM pipeline outperforms the Transformer pipeline based on BLUE score but not on TER score. As mentioned before, does the TER score

not only seems to yield a higher correlation to human judgement than BLEU (Snover et al., 2006), but also seems to yield stronger correlation with lexical diversity compared to BLEU. This could indicate that the pipeline of the Transformer in the first run is indeed more accurate than the LSTM in the first run. Further research could be done to see why these two metrics favour different models.

When looking at the first and second run of the Transformer, we see that the performance increase for the first MTS in the second run is extremely minimal, hence most of the relative performance increase is achieved in the second MTS. This is an interesting result since the performance of the pipeline massively drops as soon as the translation of the second system is executed. This would imply that the hyper parameter tuning has the biggest impact on the text expander and not on the first MTS.

The inability to translate numbers correctly might have to do with the semantic dissimilarity described in Section 2.3. To overcome this problem, the source and target sentences of the first MTS need to get more similar in terms of semantics or the volume of training data needs to increase a lot. The '$NUM' parent type has a lot of child types. Some of these types only appear once in the entire data set. The format used by the DGS corpus to portray numbers weakens the performance even more. As mentioned in Section 3.1.1, a larger number such as 1989 would be segmented into three tokens: $NUM-TEEN1: 9 (nineteen), $NUM-ONE-TO-TEN1A:9 (nine) and $NUM-TENS1: 8d (eighty) (Konrad et al., 2020). In the target sentences there would only be '1989'. In order for the MTS to recognize the combination of the three tokens as '1989', it would need a lot more data. Especially since a number like $NUM-TENS1: 8d (eighty) can appear as a stand alone number '80' instead of being part of e.g. '89'. The example mentioned above would result in three glosses together meaning the same number (1989) in spoken German. Pre-processing of the numbers could increase performance of both the pipeline and the baseline.

The results of the main research question are rather surprising. To get a better understanding of why these results are not in line with our expectations we ran some additional tests after the results came in. These test were done in order to check the semantic similarity between the source sentences and the compressed target sentences. The tests were done using the same nltk library in Python as mentioned before. To our surprise we notice that the removal of stop words decreases the semantic similarity between the source and target sentences. A decrease in translation performance for MTS trained on two languages with a lower semantic similarity is in line with the findings of Moryossef et al. (2021). The results of the test are shown in Table 6

Language pair	Semantic similarity
DGS and German	0.0206
DGS and Compressed German	0.0174

Table 6: Semantic similarity

6.2 *Limitations*

The mouthing signs attribute mentioned in Section 3.1 adds meaning to the signs gestures. The exclusion of this attribute lowers the potential accuracy of both the baseline and the pipeline. The data set is built with the help of many different translators. As mentioned by Konrad et al. (2020), translators can use different signs and combinations to represent the same word. This ambiguity reduces the accuracy of MTS trained on this data set as well.

There are more advanced forms of compression and simplification available. Some of these more advanced models could increase the semantic similarity between sign and spoken language.

There are better evaluation metrics available such as human evaluation. Research with more time and resources could implement this type of evaluation. A different evaluation metric for machine translation can impact the results drastically as shown by the results of (Shterionov et al., 2018) and the difference in TER and BLEU in this paper. Using more evaluation metrics could help to better understand the details of what happens within the pipeline.

7 CONCLUSION

This paper provides an empirical analysis to see how a pipeline of MTS's trained on sign language glosses, compressed texts and full texts differs from an end-to-end MTS trained on sign language glosses and full texts for the task of sign to spoken language translation in terms of performance. The results suggest that the inclusion of an extra MTS makes the translation process not only slower in both training and computation time but also less accurate when evaluated by the BLEU and TER scores. The results complement earlier findings (Vanmassenhove et al., 2019) about the inability of MTS's to generate a diverse output. This paper also provides proof that the specialized hyper parameters for Transformers trained on low resource

data sets can be applied in both spoken-to-spoken and sign-to-spoken translation.

Further research should implement different forms of compression or simplification to see if the target texts can get closer to the structure of sign language. There should be more focus on semantic similarity between the source and target languages. This could improve performance of the pipeline and perhaps make the pipeline outperform the baseline. Another aspect that can be improved is the implementation of mouthing signs and the pre-processing of for example the numerical ('$NUM') data in order to see whether this could improve the pipeline. By pre-processing these '$NUM' glosses, the semantic similarity between DGS and German should increase as well. Not only the implementation of more attributes but also a better understanding of the data in general is favourable. A thorough understanding and more pre-processing of the glosses could further increase performance.

REFERENCES

Alva-Manchego, F., Martin, L., Scarton, C., & Specia, L. (2019, November). EASSE: Easier automatic sentence simplification evaluation. 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* (pp. 49–54). Hong Kong, China: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D19-3009 doi: 10.18653/v1/D19-3009

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bawa, S., Kumar, M., et al. (2021). A comprehensive survey on machine translation for english, hindi and sanskrit languages. *Journal of Ambient Intelligence and Humanized Computing*, 1–34.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Covington, M. A., & McFall, J. D. (2010). Cutting the gordian knot: The moving-average type–token ratio (mattr). *Journal of quantitative linguistics*, *17*(2), 94–100.

Dixon, R. M. (2014). *Making new words: Morphological derivation in english*. Oxford University Press, USA.

Gage, P. (1994). A new algorithm for data compression. *C Users Journal*, *12*(2), 23–38.

Harispe, S., Ranwez, S., Janaqi, S., & Montmain, J. (2015). Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, *8*(1), 1–254.

Hebb, N. (2012). Flowchart symbols defined. *BreezeTree Software*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Junczys-Dowmunt, M., Dwojak, T., & Hoang, H. (2016). Is neural machine translation ready for deployment? a case study on 30 translation directions. *arXiv preprint arXiv:1610.01108*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., & Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proc. acl.* Retrieved from https://doi.org/10.18653/v1/P17-4012 doi: 10.18653/v1/P17-4012

Konrad, R., Hanke, T., Langer, G., Blanck, D., Bleicken, J., Hofmann, I., ... Schulder, M. (2020). *Meine dgs – annotiert. öffentliches korpus der deutschen gebärdensprache, 3. release / my dgs – annotated. public corpus*

of german sign language, 3rd release [languageresource]. Universität Hamburg. Retrieved from https://doi.org/10.25592/dgs.corpus-3.0 doi: 10.25592/dgs.corpus-3.0

Lankford, S., Alfi, H., & Way, A. (2021). Transformers for low-resource languages: Is féidir linn! In *Proceedings of the 18th biennial machine translation summit (volume 1: Research track)* (pp. 48–60).

Malvern, D., Richards, B., Chipere, N., & Durán, P. (2004). *Lexical diversity and language development*. Springer.

Mandya, A. A., Nomoto, T., & Siddharthan, A. (2014). Lexico-syntactic text simplification and compression with typed dependencies. In *25th international conference on computational linguistics.*

Mehdi, S. A., & Khan, Y. N. (2002). Sign language recognition using sensor gloves. In *Proceedings of the 9th international conference on neural information processing, 2002. iconip'02.* (Vol. 5, pp. 2204–2206).

Mehta, S., Azarnoush, B., Chen, B., Saluja, A., Misra, V., Bihani, B., & Kumar, R. (2020). Simplify-then-translate: Automatic preprocessing for black-box translation. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 8488–8495).

Moryossef, A., Yin, K., Neubig, G., & Goldberg, Y. (2021). Data augmentation for sign language gloss translation. *CoRR*, *abs/2105.07476*. Retrieved from https://arxiv.org/abs/2105.07476

Naresh, P., Visalakshi, R., & Satyanarayana, B. (2020). A study on sign language recognition-a literature survey. *ICDSMLA 2019*, 745–752.

Nguyen, T. Q., & Chiang, D. (2017). Improving lexical choice in neural machine translation. *arXiv preprint arXiv:1710.01329*.

Nisioi, S., Štajner, S., Ponzetto, S. P., & Dinu, L. P. (2017). Exploring neural text simplification models. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 85–91).

O'Dell, F., Read, J., McCarthy, M., et al. (2000). *Assessing vocabulary*. Cambridge university press.

Östling, R., Scherrer, Y., Tiedemann, J., Tang, G., & Nieminen, T. (2017). The helsinki neural machine translation system. *arXiv preprint arXiv:1708.05942*.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the association for computational linguistics* (pp. 311–318).

Pfau, R., & Quer, J. (2008). On the syntax of negation and modals in catalan sign language and german sign language. In *Visible variation* (pp. 129–162). De Gruyter Mouton.

Prechelt, L. (1998). Early stopping-but when? In *Neural networks: Tricks of*

the trade (pp. 55–69). Springer.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Sandler, W., & Lillo-Martin, D. (2006). *Sign language and linguistic universals*. Cambridge University Press.

Sennrich, R., Haddow, B., & Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Sennrich, R., & Zhang, B. (2019). Revisiting low-resource neural machine translation: A case study. *arXiv preprint arXiv:1905.11901*.

Shterionov, D., Superbo, R., Nagle, P., Casanellas, L., O'dowd, T., & Way, A. (2018). Human versus automatic quality evaluation of nmt and pbsmt. *Machine Translation*, *32*(3), 217–235.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006, August 8-12). 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* (pp. 223–231). Cambridge, Massachusetts, USA: Association for Machine Translation in the Americas. Retrieved from https://aclanthology.org/2006.amta-papers.25

Sun, R., Jin, H., & Wan, X. (2021). Document-level text simplification: Dataset, criteria and baseline. *arXiv preprint arXiv:2110.05071*.

Uthayakumar, J., Vengattaraman, T., & Dhavachelvan, P. (2018). A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University-Computer and Information Sciences*.

Vaerenbergh, K. V., & Tourwé, T. (2021). Distributed data compression for edge devices. In *Ifip international conference on artificial intelligence applications and innovations* (pp. 293–304).

Van den Bercken, L., Sips, R.-J., & Lofi, C. (2019). Evaluating neural text simplification in the medical domain. In *The world wide web conference* (pp. 3286–3292).

Vanmassenhove, E., Shterionov, D., & Way, A. (2019). Lost in translation: Loss and decay of linguistic richness in machine translation. *arXiv preprint arXiv:1906.12068*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in nlp. In *Coling 1992 volume 4: The 14th international conference on computational linguistics.*

Xu, P., Yang, W., Zi, W., Tang, K., Huang, C., Cheung, J. C. K., & Cao, Y. (2020). Optimizing deeper transformers on small datasets: An application on text-to-sql semantic parsing. *arXiv preprint arXiv:2012.15355*.

Zhang, X., & Duh, K. (2021, August). Approaching sign language gloss translation as a low-resource machine translation task. In *Proceedings of the 1st international workshop on automatic translation for signed and spoken languages (at4ssl)* (pp. 60–70). Virtual: Association for Machine Translation in the Americas. Retrieved from https://aclanthology.org/2021.mtsummit-at4ssl.7

APPENDIX

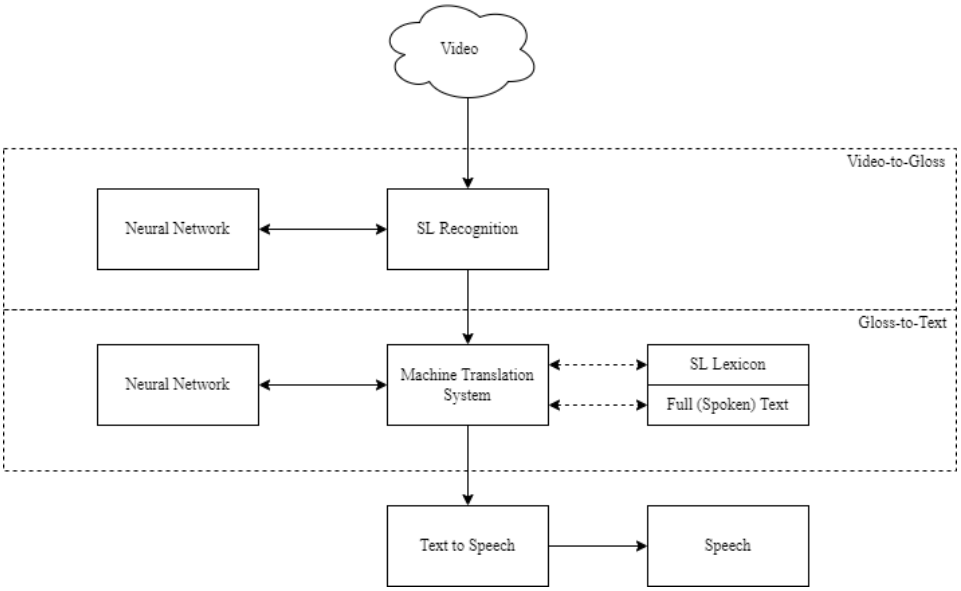

Figure 3: Baseline

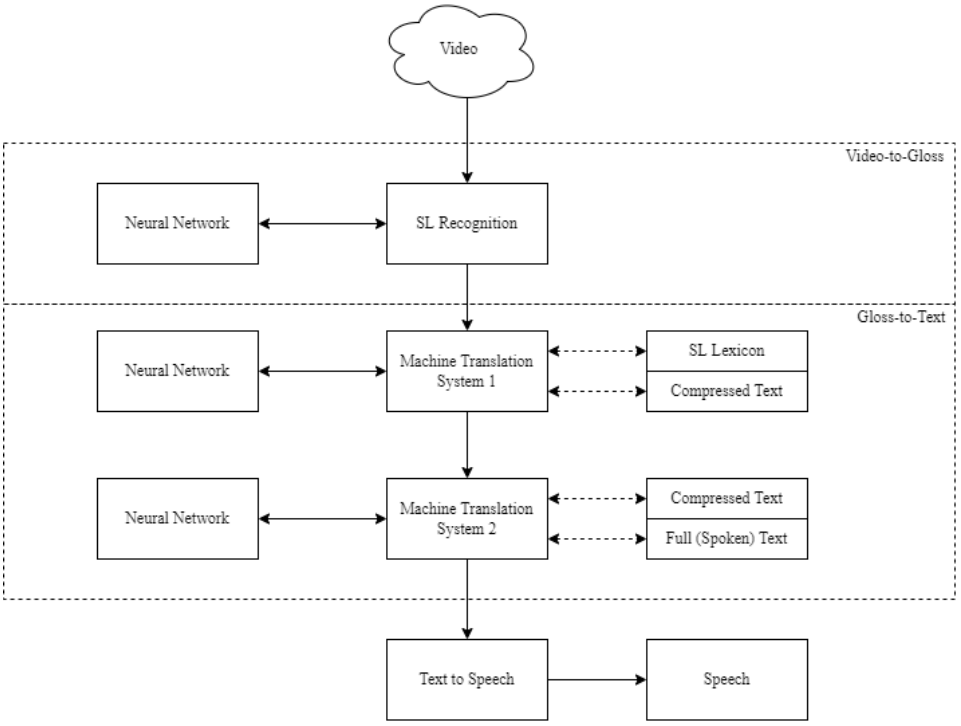

Figure 4: Proposed Pipeline

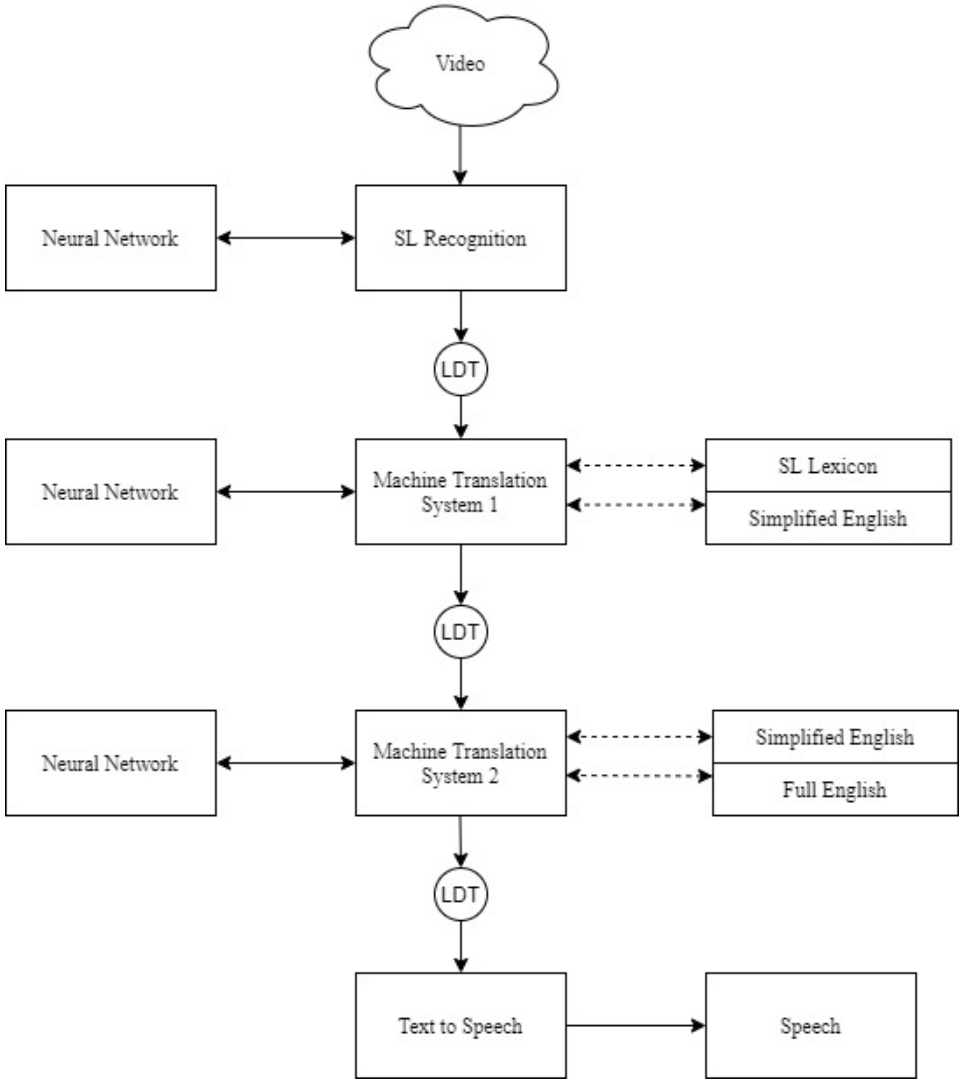

Figure 5: Proposed Pipeline Including LDT

	Sign	Spoken
0	NaN	Erinnerst du dich noch daran, wie wir damals z...
1	WISSEN2B^	Erinnerst du dich noch daran, wie wir damals z...
2	NaN	Erinnerst du dich noch daran, wie wir damals z...
3	$GEST-AUFMERKSAMKEIT1^	Erinnerst du dich noch daran, wie wir damals z...
4	NaN	Erinnerst du dich noch daran, wie wir damals z...
5	BEISPIEL1*	Erinnerst du dich noch daran, wie wir damals z...
6	NaN	Erinnerst du dich noch daran, wie wir damals z...
7	FUSSBALL2	Erinnerst du dich noch daran, wie wir damals z...
8	NaN	Erinnerst du dich noch daran, wie wir damals z...
9	VERGANGENHEIT1^*	Erinnerst du dich noch daran, wie wir damals z...
10	NaN	Erinnerst du dich noch daran, wie wir damals z...

Figure 6

0	"Doch, das ist gut.	[nan, $GEST-NM^, nan, GUT1*, nan]
1	##X8X war dann Schluss damit. In meinem Leben ...	[nan, $NUM-EINER1A:2, nan, $NUM-ZEHNER2A:8d*, ...
2	#195x wurde ich eingeschult, #195x war meine E...	[nan, ICH1, nan, $NUM-TEEN3:9, nan, $NUM-HUNDE...
3	#195x.	[nan, $ORAL^, nan]
4	#Damals in Name5 zum Fußballfest.	[nan, $INDEX1*, nan, FUSSBALL5B*, nan, FEST-FE...
5	#Deine Heimat ist also Name1.	[nan, nan, nan, nan, nan, nan, nan, ICH1, nan,...
6	#Deine Heimatregion Name1 ist interessanter al...	[nan, nan, nan, HEIMAT2*, nan, nan, nan, $NAME...
7	#Der heißt Name2/ #Name2.	[nan, $INDEX1, nan, HEISSEN4*, nan, $NAME, nan...
8	#Der kleine Name2 war damals bei mir, da er kr...	[nan, UND6*, nan, $INDEX1*, nan, $NAME, nan, K...
9	#Er ist einfach zu x.	[nan, $NAME, nan]
10	#Er ist so x, es fällt ihm zu schwer.	[nan, $NAME, nan, $INDEX1, nan, $GEST-ABWINKEN...

Figure 7

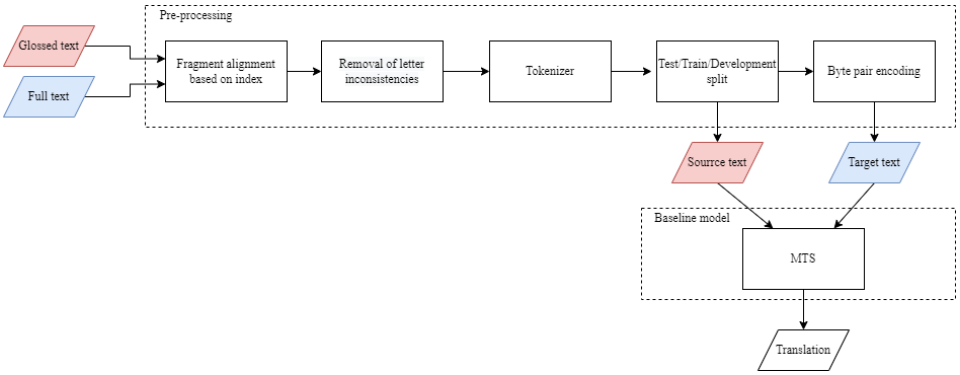

Figure 8: Flowchart Baseline model

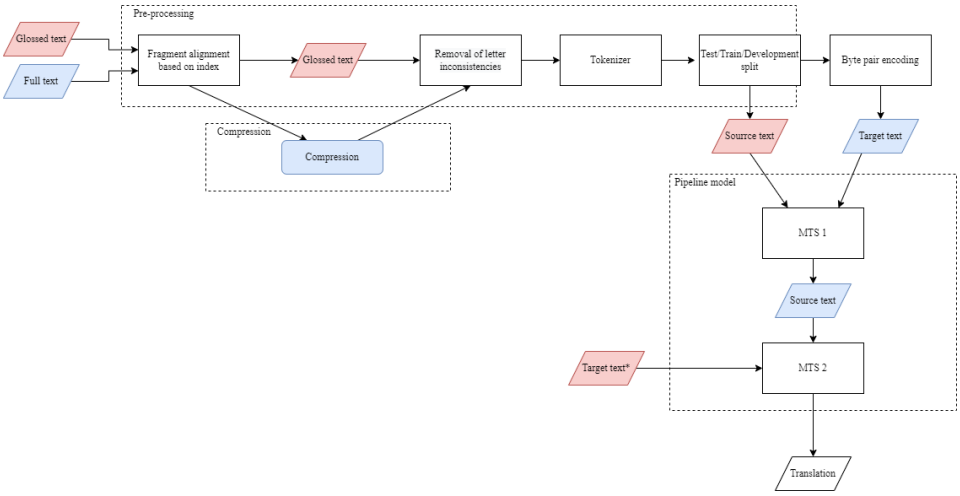

Figure 9: Flowchart Pipeline model
Target text* refers to the target set from the baseline model