Investigation of different decoding algorithms for the automatic generation of alternative language patterns in German to English translation

Bachelor-Thesis

zur Erlangung des akademischen Grades B.Sc.

Melina Zanon

2364383

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Design, Medien und Information
Department Medientechnik

Erstprüfer: Prof. Dr.-Ing. Sabine Schumann

Zweitprüfer: Prof. Dr. Larissa Putzar

Hamburg, 11. 08. 2021

Contents

Abstract

A single sentence in one language can always be translated into another language in multiple ways. Translation is therefore an ambiguous task, that is difficult to realize in an automatic way. This thesis investigates different decoding algorithms for the generation of text with a neural machine translation model. The goal is to generate multiple translations with diverse language patterns. For this purpose, an artificial neural network for German to English translation is implemented and translations are generated using different decoding algorithms. The resulting translations are then analyzed with regard to their quality and diversity.

Zusammenfassung

Ein einzelner Satz in einer Sprache kann immer auf mehrere Arten in eine andere Sprache übersetzt werden. Übersetzung ist daher eine mehrdeutige Aufgabe, die schwer automatisch zu realisieren ist. In dieser Arbeit werden verschiedene Dekodieralgorithmen für die Generierung von Text mit einem neuronalen maschinellen Übersetzungsmodell untersucht. Das Ziel ist es, mehrere Übersetzungen mit unterschiedlichen Sprachmustern zu generieren. Dazu wird ein künstliches neuronales Netz für die Deutsch-Englische Übersetzung implementiert und Übersetzungen mit verschiedenen Dekodieralgorithmen generiert. Die resultierenden Übersetzungen werden anschließend in Bezug auf ihre Qualität und Diversität analysiert.

1 Introduction

In an increasingly globalized world, people often need to communicate with each other even when they do not speak the same language. Therefore, the need for high-quality, fast automatic translation rises. The recent trend in the industry is to develop such translation systems using artificial neural networks, that learn to translate texts from data. While these systems get progressively better at producing fluent and natural language, most of them only offer a single translation of the given text. An often-overlooked fact is that languages are much more complex and variable than that and a text can always be translated in multiple different ways. Which translation is the best in a situation depends on many factors unknown to the system. It would therefore be beneficial if a system could provide alternative translations.

This work focuses on a neural network based translation system, that provides multiple alternative English translations for a German input text. The goal is to investigate, which algorithms work best for generating multiple meaning equivalent translations with alternative language patterns, such as different sentence structures or use of vocabulary.

1.1 Motivation

More than 7,000 languages are spoken in the world (Eberhard et al. 2021). Although the internet is often thought to be a very international space, very few of these languages are represented online. There is a digital language divide (Young 2015). About 77% of all internet users speak the same ten languages (Internet World Stats 2021). The language you speak, therefore limits the information that is available to you online. For example, the English Wikipedia edition has about 6 million articles, while the Danish edition has only about 280,000 (Wikipedia contributors 2021).

Research in natural language processing, that focuses on languages other than English and better machine translation systems, could help people to better navigate the digital world in their own language (Ruder 2020) or help them learn another language. In a lot of use cases, having multiple alternative translations can be even more beneficial. Some translation services, such as *DeepL*, already provide the option to not only translate into English, but to differentiate between American and British English. The same thing could be done for a wide variety of languages all with their own dialects and regional differences. Similarly, a translation system could offer differently structured translations using synonyms or paraphrases, so one can choose the translation that best fits the indented tone and style. This could be helpful for professional use cases, such as writing a formal email, but also for creative writing, such as writing a blog or fiction.

Having multiple translations can also be useful for language learning. The research team of Duolingo, which is an online language-learning platform, realized that and presented a new kind of task, called *Simultaneous Translation and Paraphrasing for Language Education* (STAPLE) (Mayhew et al. 2020).

Applications like Duolingo often combine several types of exercises, like listening or speaking exercises, fill-in-the-blank questions or sentence reconstruction exercises often with multiple choice answers and of course free translation exercises, where the user translates a prompt sentence into their first or second language.

The problem with these free translation exercises is that there are always multiple possible translations for the same sentence. Words often have multiple correct translations, for example the German word "still" can be translates to "silent", "quiet" or "calm". In many languages, such as Italian or Turkish, pronouns can be dropped in a sentence because the conjugation of the verb already implies the pronoun, but they can be used for emphasis. Often the meaning of a sentence does not change when the word order differs, for example "In winter I wear an overcoat." and „I wear an overcoat in winter.", could be translations of the same sentence.

All these different possible translations should be accepted as the right answer. The process of manually finding all the possible translations would be very time consuming. Furthermore, even human translators often translate things differently and do not share the same opinion on which translation is "correct". An interesting, but challenging task for natural language processing is therefore finding an extensive set of acceptable answers for these types of exercises, combining the task of machine translation and automatic paraphrasing.

1.2 Structure

In the following chapters the process of how multiple diverse translations can be automatically generated is elaborated.

First, in chapter 2 the foundations of current approaches to machine translation are explained. This includes training methods and state-of-the-art architectures for artificial neural networks. In chapter 3, the text data used for training and testing the machine translation system is described. Chapter 4 then gives an overview of the algorithms used to generate translations and the methods to evaluate those translations. The implementation and training of the neural network, as well as the generation of translations is described in chapter 5.

A detailed analysis of the generated translations is performed in chapter 6, answering the question which decoding algorithms best generate alternative and diverse outputs.

In the last chapter 7, an assessment of the experiments in this thesis is conducted and an outlook on possible further work is given.

2 Foundations

In order to develop an automatic translation system, a computer needs to have the ability to process natural language on an almost human level. The research field of *natural language processing* (NLP) is considered to be a branch of *artificial intelligence*, because it requires a machine to “understand” language. To achieve this goal machine learning methods and artificial neural networks are used to develop systems that automatically learn to improve with the use of text or speech data.

The following chapter introduces the foundations needed to understand the automatic translation system used in this work. First, an overview of the concept of neural machine translation is given. Next, transfer learning, a specific training technique for machine learning models, is described. After that, the transformer, a state-of-the-art model architecture for NLP, and finally T5, the specific transformer model used for the experiments in chapter 5, are introduced.

2.1 Neural Machine Translation

Machine translation (MT) is a sub-field of natural language processing, which aim is to automatically translate text from a source language into a target language. Since there is not one single right translation, but usually multiple, for any given sentence, it is considered to be one of the more difficult problems in artificial intelligence.

Early machine translation systems operated with a set of human developed rules and are therefore called rule-based machine translation systems. The inherent complexity of natural languages made this approach very difficult, because it required linguistic experts to manually write a very large set of rules and exceptions.

Statistical Machine Translation (SMT) is data-driven and does not require a set of human developed rules. Instead, it aims to learn linguistic information from a set of example translations in a parallel corpus. These systems typically consist of several sub-components, that need to be tuned separately. They are count-based and learn correlations of words and their translations. So-called phrase-based systems, which learn statistics of words occurring together in a sentence (also called n-grams), quickly outperformed the word-based approach and were considered state-of-the-art up until 2016.

While neural networks were sometimes used within SMT, neural machine translation (NMT) is an approach, that uses a single neural network to directly translate from one language to the other. Most NMT models use an *encoder-decoder* structure (Cho et al. 2014). The *encoder* is a neural network, that reads the input sentence and encodes it into an abstract vector representation, called the *context*. The *decoder* is a second neural network, that takes the output of the encoder and translates it into the target sentence. The encoder and decoder are trained jointly as one big network, making neural machine translation an end-to-end system. This new approach achieved comparable results with SMT in 2014 (Bahdanau et al. 2014). In 2016 Google announced the Google Neural Machine Translation system (Wu et al. 2016) and since then NMT is the predominant approach to machine translation in the industry.

2.2 Transfer Learning

Traditionally, supervised machine learning models try to learn one specific task from the ground up with labeled training data from the same domain and with the same distribution as the future data, the model needs to perform that task on. Therefore, these models are only able to generalize well on conditions encountered during training. For producing a similar result on a different task or in a different domain, a new model would need to be trained from the ground up with the same amount of labeled data. Depending on the task or domain it might not always be possible to gather a sufficient amount of data to train such a model.

Transfer learning is the practice of using a model trained on a source task in a source domain to solve another, related target task in a target domain, by utilizing the knowledge the model has already learned.

Definition

The following section is based on the definitions and notations of Pan and Yang. (Pan and Yang 2010)

There are two essential concepts involved in transfer learning: a *domain* and a *task*. A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, with X being $\{x_1, ..., x_n\} \in \mathcal{X}$. A task $\mathcal{T}$ in a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, consists of a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$, that can also be written as the conditional probability distribution $P(Y|X)$, which is learned from the training data, consisting of pairs $\{x_i \in X, y_i \in Y\}$.

With $\mathcal{D} = \{\mathcal{X}, P(X)\}$ and $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$ transfer learning can be defined as follows:

"Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$."(Pan and Yang 2010)

This knowledge can take on different forms and is gained in a so-called pretraining phase. It can later be adapted to the target task in two different forms: *feature-*

extraction and *finetuning* (Ruder 2019). In feature-extraction, pretrained representations are used as the input for a new model and the weights of the pretrained model are frozen. In finetuning the pretrained model is directly trained with new data and the weights are updated.

Transfer learning is predominantly used in the field of computer vision where huge, labeled data sets like ImageNet are available for object recognition. While training a neural network on the task of classifying different objects in pictures, the model learns representations for the general structure and composition of images, such as edges or shapes, before learning more task specific representations, such as a cat.

These lower-level representations of images can be transferred to a lot of other machine learning tasks where the input is images. Not having to learn the general elements an image is made of for every image recognition model, can save a lot of computation, and lowers the amount of data needed for the target task.

This is especially useful when there is limited data available in the target domain or task, but a massive amount in the source domain. For example, in natural language processing this can be applied to train models on big, easily available English data sets and later transfer that knowledge to a language where data is scarce. This is referred to as *cross-lingual learning* (Ruder 2019).

Recent approaches to NLP and NMT involve the unsupervised pretraining of a large neural network on massive amounts of unlabeled data. Often using a language modeling task, where the model is trained to predict the next word given a sequence of words and therefore learning a probability distribution of any given sequence of words. The idea is that the pretrained model “understands” the underlying structure of language and can then be applied to solve different problems, like summarization, question answering or sentiment analysis.

2.3 The Transformer Architecture

Prior to the introduction of the transformer (Vaswani et al. 2017), sequence to sequence (seq2seq) problems like neural machine translation, were solved using recurrent neural networks (RNNs). RNNs can process temporal sequences with information about previous inputs persisting in their internal states. Therefore, they are also sometimes said to have “memory”. This makes them very useful in NLP, where words depend on the context of previous words.

Using an encoder-decoder structure, these networks encoded the input sequence into a fixed-length vector from which the decoder translates the sequence. While this worked fairly well on short sentences, the models suffered in performance on longer sentences, since all the input needed to be compressed into this fixed-length representation (Bahdanau et al. 2014). Since RNNs recursively concatenate previous inputs and contexts at each step, the first input loses its impact over time, as it gets smaller and smaller. This phenomenon is called the *vanishing gradient problem* (Hochreiter

1991) and it means, that the model cannot “remember” long-term dependencies well. To fix this problem long short-term memory neural networks (LSTMs) were introduced (Hochreiter and Schmidhuber 1997). This type of recurrent neural network has different gates, that filter what information is passed on and what can be forgotten. Although this model solved the vanishing gradient problem, LSTMs still had other problems typical to RNNs. Mainly that inputs are processed sequentially, which is slow and hard to parallelize to train on multiple GPUs.

Furthermore, the fixed-length context vector passed from the encoder to the decoder was still a bottleneck in these networks, because it was the only information the decoder had to work with. *Attention* is a technique developed to solve this problem (Bahdanau et al. 2014, Luong et al. 2015). It allows the model to amplify the relevant parts of the input sequence in the decoder, in order to give additional information to the context. It improved neural machine translation significantly, by learning how to align words in a given language pair. Amongst other things, this technique made models deal better with the problem of reversed word order, which can be seen in Figure 2.1. The French translation of the English phrase “European Economic Area” is “zone économique européenne”.

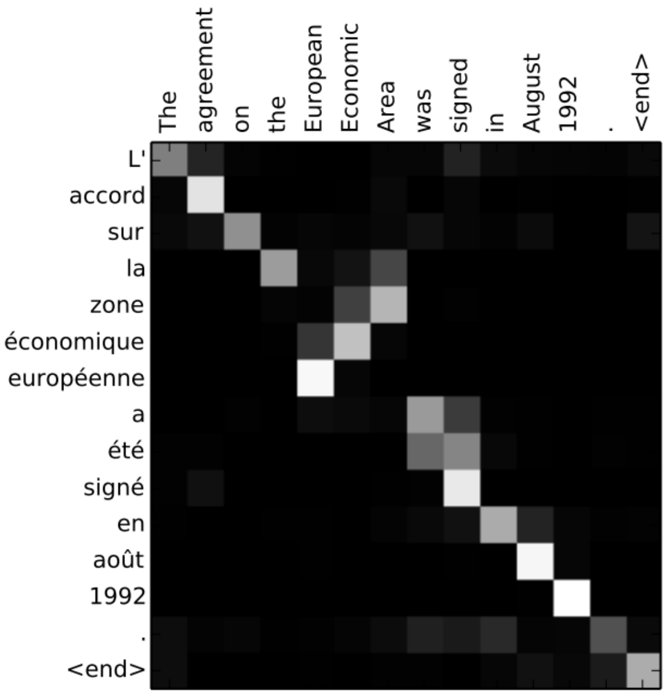


Figure 2.1: Visualization of Attention with reversed word order (Bahdanau et al. 2014)

In their paper “Attention is All You Need” (Vaswani et al. 2017), the authors proposed a new model architecture called the *transformer*. As the title suggest, this model relies solely on the attention mechanism and therefore got rid of all the RNN components in the traditional encoder-decoder structure.

The architecture shown in Figure 2.2, consists of N encoders and decoders stacked on top of each other.

Each encoder has two main components, the multihead-attention block (see 2.3.2) and a simple feed forward network. Both components have an additional layer normalization. The input sequence is first embedded into an n-dimensional vector (see 2.3.1) and then a positional encoding is added. This is necessary since the transformer processes the whole input sequence at once and not sequentially like RNNs. Therefore, the order of the sequence is lost and needs to be encoded into the input embedding to keep the important information of position.

Each decoder has two different multihead-attention blocks, one for the output embedding and one for the context vector from the encoder, and a feed forward network. The output is embedded, and the positional encoding added just as in the input, only that the output is shifted right. This is done so that the decoder does not learn to copy the input it gets, but to predict the next word given the context from the encoder and only the previous decoder outputs. Since the input is not fed in sequentially, the first multihead-attention block is masked, to prevent the decoder from considering future inputs at each position. Lastly, a linear layer and a softmax layer (see 2.3.3) are added on top of the entire decoder stack, to turn the decoder output into probabilities.

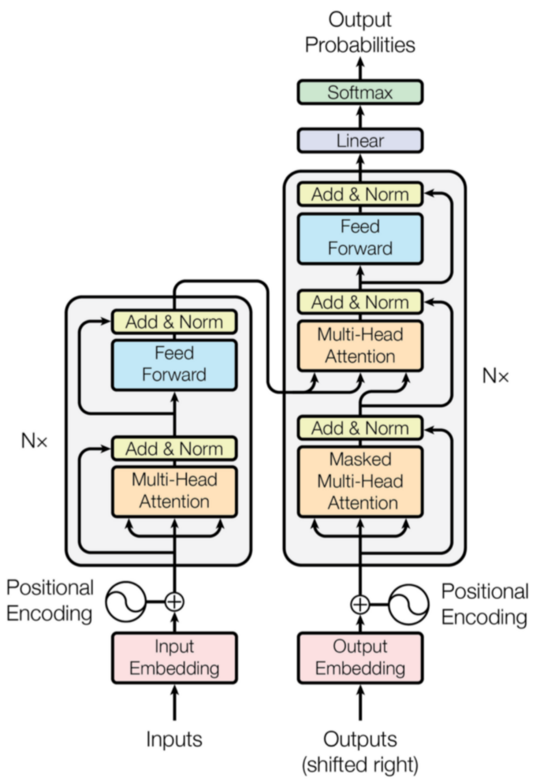


Figure 2.2: The Transformer model architecture (Vaswani et al. 2017)

2.3.1 Word Embeddings

In all NLP tasks text inputs need to be converted into a representation that a neural network can interpret; this essentially means the input needs to be numerical. One way to do that is to convert each word, or each sequence as a bag of words, to a one-hot encoded vector. This vector has the size of the vocabulary $\mathcal{V}$, with each position representing one word. The encoded word is represented by a 1 and all others are 0. This leads to very large and sparse vectors, that are not very useful in practice.

Instead, word embedding is used to learn a lower dimensional, real-valued, representation, called a *word-vector* or *word embedding*. The embeddings can be learned separately, with a specific embedding model like Word2Vec (Mikolov et al. 2013) or GloVe (Pennington et al. 2014) and then fed into another model as the input. This approach is also a form of feature-extraction transfer learning. It can be done either statically, keeping the embedding the same during training, or dynamically, updating the embedding jointly while training to better fit the specific task. Another option is to learn the embedding directly during training in a so-called *embedding layer*.

Word-vectors are not only more efficient features for neural networks, because of the dimensionality reduction making them faster to process, they are also able to encode meaning. They achieve this by embedding words, that are similar, near each other in the vector space. For example, the distance between the vectors for "apple" and "pear" would be smaller than the distance between "apple" and "car". The direction of vectors also seems to encode some meaning, making arithmetic manipulation possible. A famous example for this is subtracting the vector for "man" from "king" and adding the vector of "woman" giving a vector that is close to that of the word "queen".

In the case of the transformer, these word embeddings with the added positional encoding are the input of the first encoder and decoder in the stack. They are learned directly in the embedding layer.

2.3.2 Multihead-Attention

The main component that differentiates the transformer from other seq2seq models is the multihead-attention block. It consists of several attention heads or layers, running in parallel. In the encoder and the first multihead-attention block of the decoder, a method called *self-attention* is used.

The attention mechanism as it is used in LSTMs learns what parts of the input sequence are important to parts of the output sequence. Self-attention learns what parts of the input sequence are important to *itself*. As the meaning of a word depends on the context surrounding it, this mechanism helps to identify that context. It does so by processing each word as a weighted average of the entire sequence.

The particular attention used in the transformer is called *scaled dot-product attention* (see Figure 2.3). The first step of the calculation is to create three vectors from the input embedding or the previous encoder output, called query, key and value. In practice theses vectors are packed together into the matrices Q, K and V for

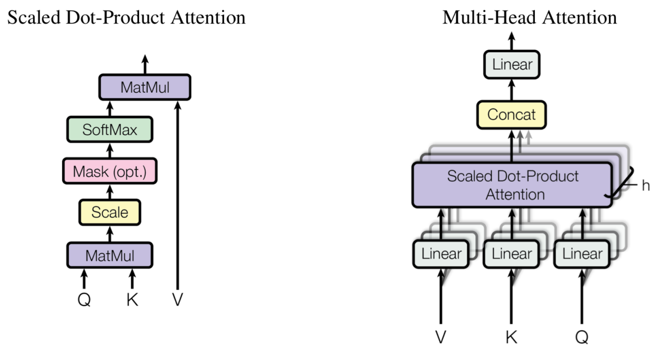


Figure 2.3: Scaled Dot-Product Attention in detail (left) and Multi-Head Attention with several parallel heads (right)(Vaswani et al. 2017)

the entire input sequence. They are calculated by multiplying the input with learned weight matrices W^Q, W^K of size $d_{model} \times d_k$ and W^V of size $d_{model} \times d_v$, which reduces their dimension for more efficient computation.

The next step is to compute a score, that is the dot product of all queries and keys. This can be done with matrix multiplication of Q and the transpose K^T. This score is then scaled by a factor of $\frac{1}{\sqrt{d_k}}$ and a softmax function (see 2.3.3) is applied to obtain weights. These are then multiplied with the values in V. The weighted value matrix is the output of the scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

As the name suggest, *multi-head* attention consists of several of these self-attention mechanisms. They run in parallel, each learning to focus on different parts of the sequence. To ensure that the heads learn different things the weight matrices W^Q, W^K and W^V are randomly initialized and updated during training for each head separately. This allows the model to have multiple different representation subspaces for the input. The output of each attention head is a weighted value matrix, resulting in h different matrices of size $d_{model} \times d_v$. To pass the output to the feed-forward network it needs to be of the dimension d_{model}. The h outputs of the attention heads are therefore concatenated and multiplied by another learned weight matrix W^O of size $hd_v \times d_{model}$ to produce the final output of the layer:

$$\begin{aligned} &\text{MultiHead}(Q, K, V) = \text{Concat}(head_i, ..., head_h)W^O \\ &\text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.2}$$

The self-attention heads of the decoder differ from the ones of the encoder in that they include an additional masking step before the softmax. In this step all positions of the input that come after the current position are set to $-\infty$, to prevent giving the decoder information about the future positions it is trying to predict.

The third use of multi-head attention in the transformer is the encoder-decoder attention, which is similar to the attention mechanism used in other seq2seq models. In these layers the queries come from the previous decoder layer, while the keys and values come from the output of the encoder stack.

2.3.3 The Softmax Function

The output just like the input of a neural network is always numerical. This output needs to be converted into something a human can understand by some activation function. In a classification task for example, this output would be the predicted class label, like "cat". A common way to do that is to normalize the output into a probability distribution over the output classes. These probabilities can be interpreted as how "sure" the network is in its decision that this is indeed a cat.

The softmax is a mathematical function, that converts the values of a vector into probabilities. It takes the output values of the last linear layer in a neural network and converts it into positive values summing up to one. The probabilities are proportional to the relative scale of the converted values.

The function $\sigma(\mathbf{x})$ (2.3) takes a vector $\mathbf{x} = \{x_1, ..., x_K\}$ and applies the exponential function to each element x_i and then divides this value by the sum of all the exponential values to normalize it. This ensures that all values sum up to one.

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{2.3}$$

In the transformer, the output of the last decoder is put through a final fully connected linear layer that converts it into a large logits vector with the size of the vocabulary $|\mathcal{V}|$. Each index in this vector corresponds to a word in the vocabulary. The softmax function is then applied to the logits vector to convert these numbers into probabilities, that indicate which words in the vocabulary are likely to be next in the sequence.

2.4 T5 Text-to-text-transfer-transformer

The *Text-to-text-transfer-transformer* (T5) was developed to test the limits of current transfer learning techniques with a variety of NLP tasks (Raffel et al. 2020). It presents a unified framework, that enables every NLP task to be treated as a text-to-text problem, i.e., the input as well as the output is text. This allows for the use of the same model and training procedure for every task, with the added benefit of making different factors in the process more comparable. The goal is to find a setup that works well for multiple different tasks.

The model architecture of T5 does not differ a lot from the original transformer architecture. Instead of using fixed position embeddings, it uses a simplified form of relative position embeddings, which are learned during training. It also uses a simplified layer normalization, that does not have an additive bias. The T5-Base model has 12 encoders and decoders, with each attention mechanism having 12 heads. The matrices Q, K and V have an inner dimensionality of 64. The feed-forward network in each encoder or decoder consists of a dense layer with an output dimensionality of 3072 followed by a ReLU activation function and a second dense layer. Sub-layers and embeddings have a dimensionality of $d_{model} = 768$. The entire model has about 220 million parameters.

The authors of the T5 paper also introduce a new data set called the *Colossal Clean Crawled Corpus* (C4) to pretrain the model on. They base this data set on the text data from April 2019 of the Common Crawl archive, which provides web extracted text by scraping about 20TB of text from HTML files each month. This data was then cleaned to retain only natural language text. This includes removing any text that is source code, by removing every page that contains a curly bracket "{" and every line that contains the word "JavaScript", or placeholder text, containing the words "lorem ipsum". Also, short pages and sentences are filtered out, as well as lines not ending in a terminal punctuation mark, pages containing offensive language and duplicate texts. Finally, an automatic language detector was used to only retain English text. The resulting data set is about 750GB of text.

The T5 models were pretrained for 1 million steps with a batch size of 2^{11}. With a maximum sequence length of 512 tokens, this means the models were pretrained on about 1 trillion tokens. During the pre-training phase they used AdaFactor (Shazeer and Stern 2018), an adaptive gradient-based optimization algorithm, and an inverse square root learning rate schedule, where the learning rate is 0.01 for the first 10^4 warm-up steps and then exponentially decays over time.

For preprocessing the text data the SentencePiece tokenizer (Kudo and Richardson 2018) is used. *Tokenization* is a process that breaks down text into small components, called tokens, like words, parts of words or punctuation. Tokenization algorithms create a vocabulary of tokens, by learning the best way to split text into tokens from a corpus and save these tokens with an id in a look-up table. For T5 a subword tokenization algorithm is used, that is trained on 10 parts C4 data and 1 part other Common Crawl data for each of the other languages used in the paper, to learn a vocabulary of 32,000 subwords. This is done to be able to later finetune the model for translation. The vocabulary used for T5 includes English, German, French and Romanian, which means that it can only process these four languages.

For the source task a *denoising* (also called *masked language modeling*) objective was chosen. Instead of just predicting the next word in a sequence of words, like in causal language modeling, denoising means predicting missing or corrupted tokens in a sequence. The data for this task does not require labels, as it is possible to randomly corrupt tokens in the training data and create corresponding targets automatically. This task teaches the model general-purpose knowledge about language.

Instead of single tokens the T5 objective, shown in Figure 2.4, corrupts spans of consecutive tokens and replaces them with mask tokens. The target to be predicted is a sequence of these corrupted spans, each prefixed by the mask token, that was used to replace it. This makes training faster than when predicting the entire sequence, since the targets are shorter. Specifically, 15% of tokens are corrupted and the average span length is set to be 3.

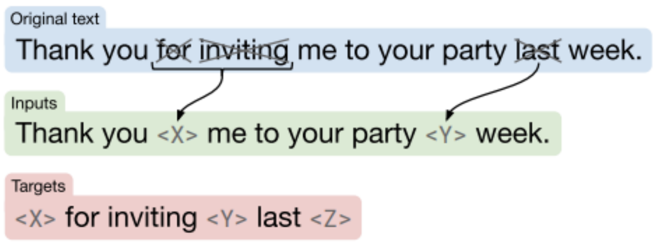

Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.
Inputs
Thank you <X> me to your party <Y> week.
Targets
<X> for inviting <Y> last <Z>

Figure 2.4: The pretrainig objective used in the T5 models (Raffel et al. 2020)

The released pretrained versions of T5 are not only pretrained on the unsupervised denoising objective alone, but are pretrained using a *multi-task pretraining* strategy, where the model is trained on several task simultaneously. This has shown to improve the results of finetuning. Using the text-to-text framework, every input is prefixed with a label like "translate English to German:", or "summarize:" to differentiate the tasks. The C4 data set is then mixed with several task specific data sets and the inputs are randomly sampled from this concatenated data. To account for differences in data set size an artificial limit is set to adjust the proportions. For the T5-Base model the limit is 2,620,000 and additionally translation data sets are cut down to 1 million examples. Since one of the tasks T5 is pretrained on is English to German translation, it might already have a good understanding of translation between these languages and could easily be used to finetune it for German to English translation without a big data set.

There are five different sizes of the T5 model:

- Small $\sim$ 60 million parameters
- Base $\sim$ 220 million parameters
- Large $\sim$ 770 million parameters
- 3B $\sim$ 3 billion parameters
- 11B $\sim$ 11 billion parameters

For all experiments in this thesis the T5-Base model is used.

3 Data Set

The following chapter describes the data being used for developing a German to English translation system. Since the idea for the experiments in this thesis was inspired by the STAPLE data set and task of the Duolingo shared task 2020 (Mayhew et al. 2020), the chapter first describes this data set. Afterwards, the German-English data set that was specifically created for this thesis is described.

3.1 STAPLE Data Set

For the Duolingo shared task 2020 the Duolingo team published five data sets in different languages (Mayhew et al. 2020). They proposed a new task called STAPLE. For the task, English prompts are taken as the input of a machine translation system and a set of plausible translations is generated.

	Train			**Val**			**Test**		
Language	engl.	trans.	ratio	engl.	trans.	ratio	engl.	trans.	ratio
Hungarian	4,000	251,442	62.9	500	27,647	55.3	500	33,578	67.2
Japanese	2,500	855,941	342.4	500	172,817	345.6	500	165,095	330.2
Korean	2,500	700,410	280.2	500	140,353	280.7	500	150,477	301.0
Portuguese	4,000	526,466	131.6	500	60,294	120.6	500	67,865	135.7
Viatnamese	3,500	194,720	55.6	500	29,637	59.3	500	28,242	56.5

Table 3.1: Sizes of the STAPLE data sets (Mayhew et al. 2020)

The published data sets consist of English prompts with several accepted translations, produced by users of the Duolingo online language-learning platform. Its domain is therefore language learner texts, which are relatively short and simple sentences, that usually provide an example of a specific grammatical rule or use of vocabulary in context. The data was sampled from English courses for Hungarian, Japanese, Korean, Brazilian Portuguese, and Vietnamese speakers. The users were given an English prompt and asked to translate it into their respective language. Data was gathered for one month to count how often a translation was given by a user. These counts were then normalized as a learner response frequency weight for each translation in a set of accepted translations of a prompt, so that they all sum up to one. The learner response frequency should indicate more fluent and natural responses, although all responses are technically correct ones. The prompts have different numbers of accepted translations, depending on length and the number of possible meanings.

The resulting data sets each have 500 prompts for the test and validation data set, respectively. The training data set size varies for the different languages and depends on the size of the English course in Duolingo. The average size of the accepted translation set lies between 50 and 350, depending on the language. This leads to a very high coverage of possible translations. All numbers of the data set are shown in Table 3.1.

3.2 German-English Data Set

As the STAPLE data set does not include German, the goal is to create a similar data set with German prompts and several correct English translations while staying in the domain of language learning text. In the shared task the use of additional corpora was allowed and teams that did use that opportunity reported better results when using other data geared towards language learners. Specifically, the Tatoeba corpus was mentioned to be extremely similar to the STAPLE domain.

Tatoeba[1] is a large database of sentences and translations, that provides examples of words in context for language learners. It is collaborative and open; the sentences and translations are provided by users. At the moment Tatoeba supports 405 languages, including German with over 500,000 sentences.

Unlike other popular sentence aligned parallel corpora like Europarl or Wikipedia based corpora, the Tatoeba corpus includes multiple alternative translations. Another option would have been using the OpenSubtitles corpus and making use of alternative movie subtitle translations, which can be useful for language learners since it is mostly dialogue. However, alternative subtitles for an entire movie include a lot of duplicate translations and are often free translations that try to shorten the dialogue according to the time constraints of the scene. Similarly Multi30k, a data set of English and German image descriptions used for translation tasks or automatic image captioning, includes multiple English sentences per image. Of those sentences however only one is directly translated to German, and the other four are independent descriptions. While they all describe the same image, they can not be treated as direct translations. Both these data sets therefore seem too noisy for the task in this thesis and the Tatoeba data is chosen instead.

Taking all German sentences with at least two English translations from the Tatoeba database results in a data set with 29,096 prompts before cleaning. This is several times larger than the largest language sets in the STAPLE data, but there are far less translations per prompt. Most of the sentences have merely two or three translations, resulting in only about 70,000 translations. Nevertheless, the groups that participated in the STAPLE challenge often used only a fraction of the provided

[1]The data from Tatoeba is released under the CC-BY 2.0 FR license and is available at `https://tatoeba.org`.

translations, suggesting that the amount of data could be sufficient to produce comparable results.

Since anyone can contribute sentences to Tatoeba, some steps are taken to clean the data while keeping in mind the language learning use case. Only correct and truly different translations are kept. First, some offensive and inappropriate language is filtered out manually. A lot of the translations only differ in punctuation. If more than two translations are available and two of them are only different in punctuation, the most "direct" translation is kept. For example, if one sentence ends in a period an another in an exclamation mark, the one ending in the same terminal punctuation mark as the German prompt is chosen. Otherwise, if there are only two translations and both are the same except for the punctuation, the whole prompt is removed, as it does not qualify as a set of alternative translations.

In an additional step all numbers are converted. Learning the numbers is an essential part of learning a language. Therefore, numbers should be converted to their word representations. Taking into account the different conversions for ordinal numbers, years and telephone numbers (e.g., October 20th = "twentieth", 1939 = "nineteen thirty-nine", 911 = "nine-one-one"), numbers are written like they are spoken, as this is how it would be done in a language learning context. For the German prompts, case has to be considered when converting ordinal numbers, for example "20. Oktober" can be "**zwanzigster** Oktober", "am **zwanzigsten** Oktober" or "der **zwanzigste** Oktober" depending on the context.

Some Tatoeba sentences included translations that converted units of measurement into their American or British equivalents. For example, the sentence "Tom hat dreißig Kilo abgenommen" (eng. "Tom lost **thirty kilos**") being translated as "Tom lost **seventy pounds**". Although technically correct, translating the numbers directly is more intuitive. Furthermore, a machine translation system will not learn conversion rates, therefore these conversions are likely to hurt the translation quality. Both German and American/British units of measurement are included in the sentences of the data set, but not as different translations of the same prompt. An allowed exception is the conversion of time, such as conversions of the 12-hour to the 24-hour system (e.g., "achtzehn Uhr" (18:00) being translated to "six o'clock"), or specific differences in German and English customs of telling the time, e.g., "halb drei" (lit. "Half **three**") being translated to "half past **two**". Telling the time is different in every language. Therefore, these conversions should be allowed for clarity.

Train			**Val**			**Test**		
ger.	trans.	ratio	ger.	trans.	ratio	ger.	trans.	ratio
27,613	63,599	2.3	530	1,153	2.2	503	1,151	2.3

Table 3.2: Size of the German-English data set

After these cleaning steps the resulting data set has 28,646 German prompts and 65,903 English translations in total. Like in the STAPLE data, a subset of roughly

500 prompts was taken for validation and testing, respectively. The average sentence length is about 7-8 tokens (including punctuation). The resulting sizes are shown in Table 3.2.

The data set is saved as translations pairs. Each German prompt is thereby repeated as many times as there are alternative translation for this prompt (between 2 and 14 times).

4 Methods

To generate translations with a machine translation model an algorithm needs to decode the model output. Depending on the specific algorithm used, the generated translations can differ considerably. The following chapter first describes the decoding algorithms used in this thesis. Afterwards it gives an overview of different methods used to evaluate machine translation, that will be used to compare the outputs of the different decoding algorithms in chapter 6.

4.1 Decoding Algorithms for Multiple Outputs

The model outputs a probability distribution over the entire vocabulary $\mathcal{V}$ for each token in the output sequence. To generate text from these probability distributions, a search algorithm needs to find a likely sequence of words, that make up the final sentence.

Searching through all possible output sequences to find the most likely one is inefficient, since the number of options grows exponentially at each step. The options for the first word being the size of the vocabulary $|\mathcal{V}|$, the options for the second word being $|\mathcal{V}|^2$ and so forth until the end of the sequence. The vocabulary of large language models is often tens or hundreds of thousands of words. Therefore, the problem is in reality solved with heuristic decoding methods, that find an approximate solution.

This language generation is called *auto-regressive*, as it uses previous outputs to predict the current output at each time step. This means, that the probability distribution of a sequence of words $\mathbf{w} = \{w_1, ..., w_T\}$ can be written as the product of conditional next word distributions $P(w_t|w_1, w_2, ..., w_{t-1}, \mathbf{c})$ at each time step t, with $\mathbf{c}$ being the context word sequence:

$$P(\mathbf{w}) = \prod_{t=1}^{T} P(w_t|w_1, w_2, ..., w_{t-1}, \mathbf{c}) \tag{4.1}$$

A simple and fast solution is to use *greedy search*. As the name “greedy” implies this algorithm greedily takes the word with the highest probability at each time step: $w_t = \operatorname{argmax}_w P(w|\mathbf{w}_{[t-1]}, \mathbf{c})$.

While this successfully picks the highest probability for the current time step, it might produce a sub-optimal sequence overall. This is because it misses high probability words, that are “hidden” behind lower probability words and therefore decreases the likelihood of the entire output sequence. Furthermore, it will always produce the same output with no possible variance, which makes it not suitable for generating multiple outputs.

Beam Search will find an output sequence with an overall higher probability than greedy search. The way it works is that it keeps track of a set of B (the beam-width) most probable candidate sequences. These candidate sequences are called *beams*. It considers the B best next words for each beam in a greedy fashion, making B^2 new candidate sequences. These are then scored, by their overall probability and the B best scored candidates are kept. This process repeats until all beams reach an end-of-sequence token, or a predefined maximum length. This solves the problem of high probability words hidden behind lower probability words and produces higher quality sentences. It can also be used to generate alternative sentences by simply outputting all beams.

Although beam search is a widely used algorithm in NLP problems, it is critiqued to produce generic outputs with a lack of diversity (Vijayakumar et al. 2018). For problems with ambiguous outputs, where there is not one clear, correct answer, the near identical sequences of beam search might not be favorable. While producing different "correct" outputs, they often only differ by one or two words and do not accurately portray the diverse possible solutions to these problems. Sentences also often stem from the same high-valued beam, that is kept earlier in the process, leading to little variety in sentence structure. For NMT that aims at only finding one best translation it might be a good solution, but when searching a variety of alternative translations, one might argue, that beam search does not sufficiently capture the complexity of that task.

Diverse Beam Search is an alternative to beam search with a diversity objective, proposed by (Vijayakumar et al. 2018). It divides the beams into G groups and greedily optimizes each group using beam search, keeping the previous groups fixed. By adding a dissimilarity term, that penalizes the use of tokens that have been used before in other groups, it then modifies the next word probabilities. Forcing the following beam search step to pick other words encourages the exploration of different paths in each group. As this allows for different words to be picked at the beginning of generation it can produce alternative sentence structures. Therefore, this approach might be relevant for alternative translations, giving more interesting variation in structure and vocabulary use.

A different approach to decode the sequence is *sampling*. In which the next word w_t is randomly picked with respect to its conditional probability distribution: $w_t \sim P(w|\mathbf{w}_{[t-1]}, \mathbf{c})$. This is done to introduce randomness into the generation in order to make text less generic and thus more natural. However, without any restrictions, sampling by itself produces nonsensical outputs as the next word can technically be any word in the vocabulary. To combat this a parameter called *temperature* is often used to change the output probabilities of the decoder, increasing high probabilities and decreasing low ones before sampling. While this decreases the chance of nonsensical tokens to be picked, it does not eliminate it entirely.

Top-K Sampling is a very simple solution to eliminate the chance of unlikely words occurring during generation, that was originally proposed for story generation (Fan et al. 2018). At each time step only the k most likely next tokens are considered while sampling. This reduces the amount of unreliable tokens, while still keeping the element of randomness.

A difficulty of top-k sampling is picking the right value for k. If it is small it might exclude good next words and limit the diversity, if it is too big unlikely words might still be included, damaging the quality of generation. Furthermore, probability distributions are different for each time step, but k stays the same. This leads again to exclusion of likely next words when the distribution is flat, i.e. there are many similarly good words, and to inclusion of unlikely words if the distribution is sharp, i.e. there are only few high probability words.

Top-P Sampling is another sampling strategy, that is also called *nucleus sampling* (Holtzman et al. 2019). It can change the probability distribution from which to sample dynamically at each time step, by only considering the *nucleus* of the distribution. The nucleus is where the vast majority of the probability mass lies. Instead of using a fixed value like k to reduce the amount of possibilities, top-p sampling chooses the minimum number of words from the probability distribution whose accumulated probability exceeds a threshold of p. This approach includes all high probability words, while suppressing the unreliable tail of low probability words sufficiently.

It is also possible to combine top-k sampling, top-p sampling and sampling with temperature. Since sampling relies on randomness, both top-k and top-p sampling can be used to generate multiple alternative outputs, by independently sampling multiple times. Therefore, they might be suitable methods for generating alternative translations with a more variable vocabulary.

4.2 Evaluation of Machine Translation

The evaluation of machine translation is a difficult task, that traditionally has been performed by trained human translators. It is therefore a slow and expensive process. To make it faster and easier to compare different systems or different version of the same system, several automatic metrics for MT evaluation have been proposed over the years. The central idea is that to measure the translation performance, it must be determined how close a machine translation is to a professional human translation. Therefore, a numerical score of this “closeness” and a set of high-quality reference translations is needed to evaluate MT quality.

The most widely used measure to compare different MT systems is BLEU (Papineni et al. 2002), even though it is frequently critiqued to not be a good estimation of translation quality. Other metrics have found to have better correlation with human judgment but are not so widespread. This is due to the fact that MT evaluation is complex and task-dependent and therefore an area of research in itself. The BLEU-Score is comparatively simple to compute, language independent and correlates reasonably well with human judgment, which has led to its widespread use among MT researchers (Post 2018). Nevertheless, it lacks specifically in comparability when using more than a single reference translation, which is why it will not be the only metric used in this thesis.

The following section will introduce four MT evaluation methods and the advantages or disadvantages they might have, in order to better understand the results of the multiple output MT system discussed in chapter 6. All of the metrics are based on the comparison of a *candidate translation*, which is the decoded output of the neural network, and one or more *reference translations*, taken from a test data set. They are also all based on the measures of *precision* and *recall*. Precision is the fraction of predictions that were correct out of all of the predictions that were made. For example, when comparing the candidate and the reference sentence, a perfect precision of 1 would mean all of the words in the candidate are also in the reference. It would however not mean that the candidate sentence covers all the words in the reference. Recall is the fraction of predictions that were correct out of the truth, so how many of the possible correct predictions were actually made. A perfect recall of 1 would mean all of the words of the reference were correctly predicted, but it does not say anything about additional wrong words that might have been also predicted.

The $\mathbf{F}_1$-Score is the harmonic mean of precision and recall. Since neither precision, nor recall on its own are good measures of accuracy a balance of the two is often taken:

$$\mathrm{F}_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.2}$$

F_1 is also called balanced F-Score because it weighs precision and recall evenly, but it can be parameterized to give either one more weight.

The main scoring method in the STAPLE Paper is weighted F_1, where recall is weighted by the learner frequency rate of the accepted translations. In this work F_1 will be unweighted as the data set used does not have weights for the target translations. Instead of on the word level the score is here calculated on the sentence level, by comparing the set of true translation with the set of predicted translations. It is calculated for each prompt sentence s with accepted translations set $\mathcal{A}_s$ and predicted translation set $\mathcal{P}_s$. The number of correct predictions, in this case is the number of sentences that intersect between $\mathcal{P}_s$ and $\mathcal{A}_s$. Precision (4.3) is then calculated as the number of correct sentences out of the set of predicted translations and recall (4.4) is calculated as the number of correct sentences out of the set of accepted translation.

$$precision = \frac{|\mathcal{P}_s \cap \mathcal{A}_s|}{|\mathcal{P}_s|} \tag{4.3}$$

$$recall = \frac{|\mathcal{P}_s \cap \mathcal{A}_s|}{|\mathcal{A}_s|} \tag{4.4}$$

These F_1-Scores for each prompt s are then averaged over the entire test data set. Note, that this sentence level calculation means that only exact matches of sentences are counted as correct. To make this easier punctuation and capitalization are ignored. This is therefore only a measure of how many of the provided reference translation the system generates and not a good measure of translation quality. Furthermore, the score depends on how many alternative translations are generated. The following measures in this chapter calculate precision and recall on a word or n-gram level and are therefore more informative on the overall ability of the system to generate alternative translations.

BLEU is short for *bilingual evaluation understudy* and was proposed as an automated understudy to human translators by (Papineni et al. 2002). It calculates a score from 0 to 1, although it is often displayed as 0-100, with 1 or 100 being a perfect match to the reference translation. It is not intended to be used on the sentence-level but rather averaged over a whole corpus, since sentence-level scores vary a lot from human judgment and depend heavily on the reference translation provided. The authors claim this effect cancels itself out when averaged over a test data set with translations provided by different translators.

The implementation of BLEU is rather simple. It is based on n-gram precision. The matching n-grams of a candidate translation and a reference translation are counted, disregarding the order in which they appear, and then divided by the total number of n-grams in the candidate translations. One problem of precision is that the repetition of words is rewarded, even when the sequence is highly unlikely.

Given the references “the cat is on the mat” and “there is a cat on the mat” and the candidate translation “the the the the the the the”, unigram precision would be 7/7 because the word “the” also appears in the references.

In order to overcome this problem BLEU uses a *modified* n-gram precision. The idea is that a word should not appear more often in a candidate translation than in any of the references. So, the maximum number of times the word appears in one of the references is counted. In this example “the” appears two times in the first reference. The count of the word in the candidate translation is then clipped by this maximum: $Count_{clip} = \min(Count, MaxRefCount)$. Making the modified unigram precision of this example 2/7.

The clipped n-gram counts are calculated for each sentence in a test data set, summed and then divided by the total number of n-grams in all candidate translations. Giving a modified precision score p_n for the entire test corpus. This is done for all n-grams up to a parameter N. Unigram precision would therefore perform exact word matching to check whether the same words are used, while higher n-grams compare whether or not the sentences have the same structure. All calculated n-gram precisions are averaged by taking the geometric mean, which is equivalent to the average logarithm with uniform weights. This is done because precision decreases exponentially with n, therefore a logarithmic average is used to represent all values fairly instead of a linear average.

While the modified n-gram precision successfully penalizes translations that are too long, by not counting n-grams that do not appear in the reference or clipping n-grams that appear too often, it fails to recognize translations that are too short. Since the denominator is the total number of n-grams in the candidate translation, a really short sentence with two words can have a unigram precision of 2/2 if both words appear in the reference. In this case, even though the references might be much longer, and the system only translated a small portion of the sentence, it would get a perfect score.

Therefore, the final step in calculating BLEU is to multiply the score by a *brevity penalty*, which is computed over the entire corpus so it does not punish short sentences harshly. It is calculated using the test corpus' effective reference length r, which is the sum of all reference lengths best matching the length of their corresponding candidate translation and c which is the length of all candidate translations:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r. \\ e^{(1-r/c)} & \text{if } c \leq r. \end{cases} \tag{4.5}$$

Given all modified n-gram precisions p_n of n-grams up to N, uniform weights w_n set to $1/N$ and the brevity penalty as shown in 4.5, BLEU is calculated as follows:

$$\mathrm{BLEU} = \mathrm{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{4.6}$$

Although BLEU was specifically developed to be used with multiple reference translations, it is in practice often used with only single reference test data sets. Increasing the number of references always increases the BLEU score, since it is more likely to have matching n-grams. Therefore, comparing BLEU scores across different test data sets is difficult, especially when the number of references differs. This fact, the different parameters like N and other practices like different preprocessing of the data have led to the critique that BLEU scores are not comparable and therefore inconclusive. Matt Post proposed in (Post 2018) that researchers should use the same BLEU-scheme with official test data sets and no additional preprocessing to make BLEU a more uniform benchmark. He developed a tool called SacreBLEU[1] that makes it easy to do so.

[1]The SacreBLEU Python module is available at https://github.com/mjpost/sacrebleu

METEOR stands for *Metric for Evaluation of Translation with Explicit word Ordering*. It was developed to explicitly address the weaknesses of BLEU by (Banerjee and Lavie 2005) and has since been continuously updated to achieve better results.

The main weakness of BLEU is that is does not consider recall, which has been shown to play a more significant role than precision in achieving high correlation with human judgment. Recall is the ratio of matched n-grams and the total n-grams in the reference translation. It is an important score, since it shows how well the candidate translation covers the content of the reference. BLEU does not consider it, because it is unclear how to handle recall when considering multiple references. Furthermore, BLEU's n-gram matching only indirectly measures sentence structure with higher order n-grams. It also does not work well on the sentence level, because when n-gram precisions are calculated for a single sentence and there are no n-gram matches for one of the values for N the entire score averages out to zero, even when there are lower order n-gram matches.

METEOR attempts to solve these issues. It uses only unigram matching in three different stages to find the best alignment of the candidate and reference translation. An alignment is defined as a mapping between unigrams, in which every unigram matches with one or zero unigrams in the other sentence. The first stage matches exact unigrams, the second stage matches unigrams after stemming. *Stemming* is the process of reducing a word to its common root form without prefixes, suffixes, or other modifications. E.g., reducing "playing", "plays" and "played" to the root "play". In the third stage words that are synonyms of each other are matched. It uses the WordNet database to find synonyms. Each stage matches only the unigrams, that have not been matched yet, thereby preferring direct matches over stemmed matches or synonyms.

In each stage the largest subset of mappings that fulfills the alignment criteria is selected. If there are several subsets of the same size, the one with the least unigram mapping *crosses* is selected. Given two unigram mappings (t_x, r_x) and (t_y, r_y), with t being a unigram in the candidate translation at position $pos(t)$ and r being a unigram in the reference translation at position $pos(r)$, a cross occurs when the result of $(pos(t_x) - pos(t_y)) \times (pos(r_x) - pos(r_y))$ is negative.

Considering the example in Figure 4.1, the unigram mappings (t_{cat}, r_{cat}) and (t_{mat}, r_{mat}) of the candidate "on the mat the cat sat" and the reference "the cat sat on the mat" cross because $(pos(t_{cat}) - pos(t_{mat})) \times (pos(r_{cat}) - pos(r_{mat})) = (5-3) \times (2-6) = -8$. This *cross* can also be visualized as the red crossing lines shown in Figure 4.1.

From the resulting unigram matches, precision P as the number of matched candidate unigrams divided by the total number of candidate unigrams, and recall R as the number of matched candidate unigrams divided by the total number of reference unigrams, are calculated. An harmonic mean F_{mean} (4.7) is computed that gives more weight to recall with the parameter α.

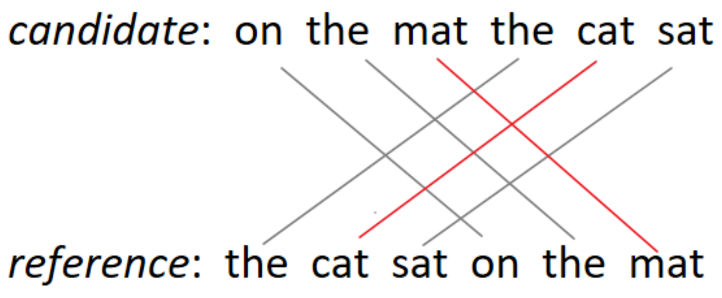

candidate: on the mat the cat sat
reference: the cat sat on the mat

Figure 4.1: Visualization of METEOR's unigram mapping with crosses (own figure based on (Banerjee and Lavie 2005))

$$F_{mean} = \frac{PR}{\alpha P + (1-\alpha)R} \tag{4.7}$$

Lastly a *penalty* is computed, to account for word order. For that the matched unigrams are grouped into *chunks*. These chunks are n-grams that match exactly between reference and candidate translation. The fewest possible number of chunks is selected, so that a perfect translation would result in one chunk and no higher matching n-grams would result in as many chunks as there are unigram matches. The penalty is then calculated so that fewer chunks have a small penalty, and more chunks have a higher penalty:

$$Penalty = \gamma \times \left(\frac{NumberOfChunks}{NumberOfUnigramMatches}\right)^{\beta} \tag{4.8}$$

The final METEOR Score is calculated as follows :

$$\text{METEOR} = F_{mean} \times (1 - Penalty) \tag{4.9}$$

If there are multiple references, the scores are calculated for each reference and the best one is chosen. For calculating METEOR for an entire test data set, precision, recall and the penalty are aggregated over all sentences before the final score is calculated. Since 2005 METEOR has been updated several times. The current version 1.5 (Denkowski and Lavie 2014)[2], includes a fourth stage, that matches paraphrases, includes more advanced text normalization applied before scoring, has language specific function word lists with a parameter that gives more weight to non-function words, as well as additional weights for precision and recall. All parameters are tuned to maximize correlation with human judgment instead of using fixed values.

[2]An official Java implementation of METEOR 1.5 is available at `https://www.cs.cmu.edu/~alavie/METEOR/`

ROUGE stands for *Recall-Oriented Understudy for Gisting Evaluation* and is a package of several recall-based measures. It was developed by Chin-Yew Lin (Lin 2004) for the evaluation of automatic summarization but can be applied to MT as well (Lin and Och 2004). Similar to METEOR it was inspired by BLEU and attempts to solve its issues, like the lack of recall, having no direct measure of sentence structure and not working on the sentence level.

The following section introduces only a part of the available metrics in the ROUGE package, explicitly the ones that have shown to be effective in machine translation.

The ROUGE-L metric is an F-Score variation, where precision and recall are based on the *longest common subsequence* (LCS) of a candidate and reference translation. A sequence Z is a subsequence of another sequence X, when the elements of Z appear in the same relative, but not necessarily continuous order as in X. For example $[A, B, D]$ is a subsequence of $[A, B, C, D]$. If two sequences X and Y both have the same subsequence, it is called a *common subsequence*. For the calculation of ROUGE-L the candidate and reference translation are considered to be sequences of words. The longer the LCS of these sequences, the more similar they are. Precision P is then calculated as the length of the longest common subsequence $LCS(X, Y)$ of the candidate sequence X and the reference sequence Y, divided by the length of X. Recall R is the same $LCS(X, Y)$ divided by the length of Y:

$$P = \frac{LCS(X,Y)}{|X|} \tag{4.10}$$

$$R = \frac{LCS(X,Y)}{|Y|} \tag{4.11}$$

The ROUGE-L score, like all of the ROUGE scores, is then the standard parameterized F-Score usually with a weight that favors recall. If there are M references the best score out of a set of $M - 1$ references is chosen M times, each time leaving out a different reference. These scores are then averaged for the final score. Since LCS compares in-sequence co-occurrences of words that are not necessarily consecutive, it measures word order and sentence structure with a flexible approach that allows for words in between not to match. Considering the example reference “the cat killed the bird” and two alternative candidate translations “the cat kills the bird” and “the bird kills the cat”, the LCS of the first candidate is [“the”, “cat”, “the”, “bird”], while the second is [“the”, “cat”] or [“the”, “bird”]. This would give the first translation a ROUGE-L score of $4/5 = 0{,}8$ and the second a score of $2/5 = 0{,}4$, when using a balanced F-Score, clearly showing the first translation to be the better one. In comparison, both sentences would have the same BLEU score with $N = 2$, since they both have four unigram matches (“the”, “cat”, “the”, “bird”) and two bigram matches (“the cat”, “the bird”).

Another measure called ROUGE-W uses a weighted LCS, to improve the score of consecutive matches over nonconsecutive matches. The length L of the consecutive word matches in a LCS is counted as L^{weight} instead of just L in the calculation of precision and recall. It is therefore stricter regarding sentence structure than ROUGE-L and rewarding the use of the same word order as the reference.

The ROUGE-S score uses skip-bigram co-occurrences to calculate precision and recall. A skip-bigram is a pair of words in a sentence, that are in their sentence order, with an arbitrary gap in between. Using the same example given earlier, the sentence "the cat killed the bird" has 10 skip-bigrams: ("the cat", "the killed", "the the", "the bird", "cat killed", "cat the", "cat bird", "killed the", "killed bird", "the bird"). This is equal to the combination $C(n,k) = \frac{n!}{(n-k)!k!}$ with $k = 2$ and n being the length of the sentence.

Given a candidate sentence X and a reference sentence Y, $SKIP2(X,Y)$ as the number of skip-bigram matches between the two, $C(|X|,2)$ and $C(|Y|,2)$ as the number of all skip-bigrams in the candidate and reference respectively, precision and recall are calculated as follows:

$$P = \frac{SKIP2(X,Y)}{C(|X|,2)} \tag{4.12}$$

$$R = \frac{SKIP2(X,Y)}{C(|Y|,2)} \tag{4.13}$$

ROUGE-S counts consecutive matches as well as flexible matches with words in between. It is therefore sensitive to word order and overall sentence structure. For more efficient computation in longer sequences the gap that is allowed to be between the words of a skip-bigram is usually limited by a parameter d_{skip}. Candidate sentences that only have unigram matches and no skip-bigram matches get a ROUGE-S score of zero. To still count those sentences an extension called ROUGE-SU, that additionally counts unigram matches, is sometimes being used.

5 Experiments

The following chapter outlines the design of the experiments in this thesis. First, the technical implementation and the tools used are described. This includes the framework for finetuning the T5 model, the data preparation process and the implementation and specifications of the four decoding algorithms that are investigated in chapter 6. Then, the training process and the chosen hyperparameters are described.

5.1 Implementation

The experiments for finetuning the T5 model, the decoding algorithms, as well as all evaluation metrics except for METEOR are implemented in Python. The model is built with *PyTorch Lightning* (Falcon 2019), an open-source Python library, that provides a light-weight implementation of the *PyTorch* deep learning framework (Paszke et al. 2019). It is intended to help researchers spend less time on engineering and more on research, while still keeping all the functionality of PyTorch.

The T5 model checkpoint that is used to initialize the PyTorch model and the pretrained SentencePiece tokenizer for T5 are provided by the *Hugging Face Transformers* library (Wolf et al. 2020). The model architecture with all of the layers and the pretrained weights are stored in this checkpoint, which makes it easy to build on for finetuning.

input	Wie heißt dein jüngerer Bruder?
tokenized	[Wie, heißt, de, in, jüng, er, er, Bru, der, ?, </s>, <pad>, <pad>]
tensor	[2739, 13172, 20, 77, 30683, 49, 49, 9465, 588, 58, 1, 0, 0]
atn. mask	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]

Table 5.1: Example of data preparation with maximum token length = 13

The data is loaded as DataFrames with the *pandas* library (McKinney 2010). It is then prepared as a PyTorch Dataset, by splitting the lines into inputs and targets, which are then tokenized to subwords and converted to PyTorch Tensors. *Tensors* are a data structure similar to vectors and matrices, that can run on GPUs. In order to give all inputs and targets the same length, they are padded to a maximum token length of 110 for the German sentences and 100 for the English translations. In addition to the input id tensors an attention mask is generated for each input and target, that indicates which tokens should be taken into account and ignores padding tokens. Unlike the original T5 implementation, no prefix like “translate German to English:” is added to the input. Since this is the only task in which the input is not English, the model recognizes the task without the prefix. It is therefore omitted to keep the input sequences short. An example of the data preparation process is shown in Table 5.1.

A PyTorch Lightning DataModule is implemented to load data sets for training, validation, and testing. These data sets are then split into batches for training. The examples in the training data are shuffled, while the ones for validation and testing are kept in order.

The training is implemented with the PyTorch Lightning Trainer module, that automatically takes care of logging, multiple GPU support and batch iteration. For more efficiency two additional callbacks are implemented in the trainer. The checkpoint callback ensures that only the best checkpoint is saved. For that the validation loss is monitored during training and a checkpoint is saved only if it is better than the previous checkpoint. This saves the unnecessary storing of big checkpoints for every training epoch. In another callback, early stopping is implemented, which stops training if the validation loss does not improve for more than three epochs. This makes training much faster.

The four different decoding algorithms described in Section 4.1, are implemented using the *generate* function from the Hugging Face transformers library. For each German input sentence three alternative English translations are generated from the model output. In addtition to the test data from the Tatoeba corpus, translations are generated from the WMT14 *newstest2014* German data. This is a common data set for testing NMT and is one of the recommended data sets in the SacreBLEU library to report BLEU scores on. It is also the same test data the authors of the T5 paper used to report BLEU scores for English to German translation. The data set contains 3,003 sentences of the news domain.

The input sentences are tokenized and padded to a max length of 180 before they are fed into the model. The maximum output length is set to 150 tokens. From the model output ids are generated using the different decoding algorithms. These ids are then detokenized skipping special tokens like padding tokens or end-of-sequence (EOS) tokens, to form a text string again. After all the translations are generated, they are stored in TSV files, so they can be analyzed and scored later.

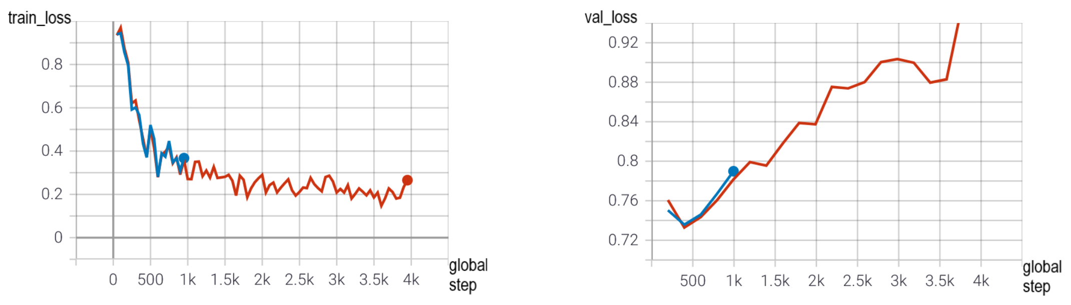
For the beam search algorithm, the beam width B is set to 10 and early stopping is implemented to stop generation when all beams reached an EOS token. For the implementation of diverse beam search the number of groups is set equal to the number of beams $G = B = 10$, to have the most possible diversity in the groups. The penalty for tokens used in other groups is set to 1.5. In the sampling methods k is set to 10 for top-k sampling and p is set to 0.9 for top-p sampling.

Each decoding algorithm generates 10,518 translations, 9,009 for the WMT14 data set and 1,509 for the German-English data set created from the Tatoeba corpus.

5.2 Training

The hyperparameters chosen for this training are based on the finetuning experiments in the T5 paper but are modified for the data and the hardware available. Adafactor is used as the optimization algorithm with a constant learning rate of 0.001. Instead of a batch size of 128 a batch size of 64 is used to fit on the available hardware. To save some computation the maximum sequence length is shortened to 110 tokens, which is equal to the longest sequence in the data set. This makes the batches much smaller and easier to process. Only the best checkpoint according to validation is used for translating.

The training is done on a Linux virtual machine running in a Docker container on the university server of the HAW Hamburg. It is using five NVIDIA Quadro P6000 graphic cards. During training and validation German inputs are repeated, but each time a different English translation is given as the target. By explicitly showing the model alternative translation it will hopefully be able to produce more alternative outputs.



(a) Training loss **(b)** Validation loss

Figure 5.1: Development of loss in the model versions 1 (blue) and 2 (red) (Screenshot Tensorboard)

Two versions of the model were trained, one with early stopping and one without. The first version stopped training after 5 epochs and about 1,000 steps, because the validation loss was lowest in the second epoch. The second version trained for longer, but the validation loss did only increase with more training. This is to be expected when training such a large model on a relatively small amount of data. The model overfits easily and memorizes the training data instead of learning to generalize, which leads to decreased performance on new data. This phenomenon can be seen in Figure 5.1, where the training loss (see 5.1a) slowly decreases over time, but the validation loss (see 5.1b) increases quickly after an initial drop.

The validation loss was lowest for both version on global step 397. Version 2 reached a slightly better value of 0.7329 than version 1 with 0.7358. Therefore, the generation of translations will be done with the best checkpoint of version 2.

With the computational power of five GPUs and a batch size of 64, each epoch took about five minutes. Version 1 trained for 25 and version 2 trained for 100 minutes.

6 Results

The goal of the experiments in this thesis is to investigate, which decoding methods work best for generating multiple meaning equivalent translations with alternative language patterns, such as different sentence structures or use of vocabulary. To evaluate the translations of each decoding algorithm, the outputs are scored with different metrics. The following chapter first analyzes these scores and what they might indicate about the generated translations. Afterwards, examples are shown for each method and the quality and diversity of the output is discussed.

6.1 Metrics

Metric	**Beam Search**	**Diverse Beam Search**	**Top-K Sampling**	**Top-P Sampling**
F1-Score	**32.80**	27.76	24.26	26.32
Precision	**29.03**	25.48	25.25	**28.16**
Recall	**38.96**	31.76	24.96	26.56
BLEU (all)	**53.43**	48.53	46.42	**50.04**
BLEU (first)	**60.26**	**57.78**	46.55	48.43
BLEU WMT (all)	**19.55**	**17.37**	13.72	14.59
BLEU WMT (first)	**19.72**	**18.55**	13.77	14.58
METEOR	**44.51**	41.38	40.88	**42.61**
ROUGE-L	**63.88**	58.23	60.03	**62.25**
ROUGE-W	**49.87**	45.55	46.95	**48.72**
ROUGE-S	**42.48**	37.22	38.60	**40.79**
ROUGE-SU	**48.63**	43.16	44.85	**46.96**

Table 6.1: Scores of the decoding methods (scaled to be between 0-100 for readability)

All metrics shown in Table 6.1 except for METEOR, for which the official Java implementation with pre-tuned parameters is used, are implemented using Python. For the calculation of BLEU, the SacreBLEU python library is used with the standard parameter of $N = 4$, mixed case evaluation, exponential smoothing and the WMT standard "13a" tokenization as suggested in "A Call for Clarity in Reporting BLEU Scores" (Post 2018). Since BLEU is very sensitive to differences in the data set, the scores are also reported on the WMT14 *newstest2014* data set for comparability to other systems. ROUGE scores are calculated using PyRouge[1] from the *rouge_metric* library. For all ROUGE scores the balanced F-Score (F_1) is reported. The weight for ROUGE-W is set to 1.2 and the skip gap d_{skip} is set to 4 for ROUGE-S and ROUGE-SU. Sentences are tokenized by white space.

All scores reported on the German-English test data from the cleaned Tatoeba data are scored using all the available references, which range from 2 to 8 references per prompt.

Beam Search scored the best in all metrics. It reached a high F_1-Score and a very high recall, which suggest it covered the given reference translations the best. It means 39% of the references were exactly reproduced. Assuming the references are correct, this score should indicate how many translations were at least correct. The actual number of correct translations is presumably higher, because only exact reference matches are counted. Since three translations are generated and most sentences in the data have two references, it is more likely to have a high recall over high precision.

The F_1-Score reached is comparable to the first ranking Japanese system or fifth ranking Portuguese system assessed in the STAPLE paper. Although Portuguese is closer to German since both are Indo-European languages and it is therefore a better analogy. Note that this is only a very loose comparison, considering that other data was used, and the score is not weighted in this case.

BLEU-scores for the non WMT data are very high, because BLEU increases with multiple references. When comparing all translations vs only the first output of beam search, the BLEU score drops significantly (see Table 6.1). This suggests the quality of the lower scored beams is less in comparison to the highest scoring beam. Although this effect seems to cancel itself out when longer sentences are used like in the WMT data. This can be explained by the lack of variety in beam search leading to sentences with only one or two different words. These small differences have less of an impact in longer sentences but can hurt the score when the sentence only has a few words.

Although a BLEU score of 19.55 on the WMT14 data is comparatively low, the finetuned T5-Base model for example reached a score of 30.9 in English to German translation on the same data, it is still an adequate score considering only 66.000 examples were used for finetuning (vs. 4.5 million in T5) and the training data is not of the same domain as the test data. The sentences in newstest2014 are considerably longer and more complicated with much more news specific vocabulary.

Beam Search also got the highest METEOR score with 44.51. This means, that the candidate translations have a lot of matching unigrams with the references. Out of roughly 12.000 words 10.000 were matched, this is a lot when considering that not all words can be matched since translations do not always match word to word. Especially in German, where nouns are often concatenated, one noun would match

[1]A fast Python implementation of full ROUGE metric available at https://github.com/li-plus/rouge-metric

with several words in English, but the calculation of METEOR only allows for one match. Almost all words were matched in the first stage (the exact stage) and only a small portion of synonym and paraphrase matches were found. Before adding the penalty to account for word order, the F-Score of METEOR was about 81, but the score suffers from the fragmentation penalty. The matches were grouped into 2.495 chunks for the 1.509 sentences, which averages to 1.7 chunks per sentence. This suggest, that while the vocabulary use in the sentences was good, the overall structure of the sentences was different from the references, or some non-matching words in between resulted in more chunks and therefore a higher penalty.

The difference in structure can also be seen when comparing the different ROUGE scores. While ROUGE-L is always fairly high ROUGE-W is always about 14 points lower. This means that there are a good number of words in the same relative order as the reference, but they are not consecutive. So, the structure is not exactly like in the reference or single words in between do not match. ROUGE-S is even more sensitive to word order, the score is lower. The counting of additional unigram matches results in a ROUGE-SU score about 6 points higher.

Diverse Beam Search scored the worst in all ROUGE scores, but second best in all BLEU scores. Like normal beam search, it has a high recall, covering about 32% of the references.

Although, the BLEU scores are still high, DBS has the largest drop of 9 points when comparing only the first and all three translations. The quality seems to decrease even more than normal beam search. It is also the only decoding method with a significant drop when comparing the first and all translations of the WMT data. This means the second and third translations are more dissimilar from the reference. This proves that the output of DBS is significantly more diverse, than that of beam search. It hurts the overall scores of the method because the references have mostly the same sentence structure and a different, although correct structure would result in a lower score. This also explains the low ROUGE scores because they measure word order.

The METEOR score for DBS is only slightly better than the lowest score. It reaches an F-Score of 77. It has less word matches than beam search but more chunks (2.727) leading to higher fragmentation penalty. It has also less exact matches in the first stage and more paraphrase matches in the last stage, suggesting DBS produces more alternative vocabulary.

Top-K Sampling has the lowest scores overall, but scores higher than DBS in the ROUGE scores. It has the lowest F_1-score due to having a low recall score. Precision is higher than recall unlike the previous methods. A high precision means all generated translations are in the set of references. This is unlikely because for most sentences the number of generated translations is higher than the number of references. Since it is nevertheless the case here, it indicates that there might be repetition in the translations. Repetition is rewarded by precision (see 4.2).

When sampling independently from the same probability distribution, there is a chance to sample the same exact sequence several times, especially when the distributions are sharp. Looking at the generated output of the top-k sampling method, out of the 503 translated sentences 174 had repeated sentences in their set of translations, which is about 35%. Out of those sets with repetition, 146 times two equal sentences were generated and 28 times all three translations were the same.

The repetition also explains why there is no drop in the score when comparing only the first to all three translations in the BLEU score, since a significant number of translations are the same. The BLEU scores are also low compared to the other methods indicating that none of the generated candidates match well with the references. This is especially the case for the WMT data, where sampling scores the lowest. This can be due to length or more complicated vocabulary which apparently suffers when allowing to sample from potentially low probability words.

The Top-k sampling method also has the lowest METEOR score, due to having the highest number of chunks of 2,831 and lowest F-Score of 76.5. Although it matched more words than diverse beam search.

In the ROUGE scores top-k sampling beats DBS, suggesting that while the overall quality of the translations is worse, the sentence structure might be similar to the ones of the references. This could be due to high probability function words still matching, even if the content words do not.

Top-P Sampling scored surprisingly well across the metrics. Although it is also a sampling method, it achieves significantly better scores than top-k sampling. Especially the BLEU, METEOR and ROUGE scores are comparable to the ones of beam search.

It also has higher precision than recall, due to repetition. Top-p sampling repeated itself even more than top-k, 207 translation sets had repetitions (about 41%). Two duplicate sentences were generated 165 times and three equal sentences 42 times.

The BLEU score of all three translations (50.04) is better than the one of a single output translation (48.43). None of the other methods improve in that way. This could be due to the random sampling sometimes producing bad translations, but the error seems to average out when sampling more times. Although this only seems to be the case for shorter sentences, since both WMT BLEU scores are almost the same.

The METEOR score is similar to beam search. The F-score reached is 78.5 and 2.636 chunks are grouped together. The number of matched words in all stages is also very similar to the ones in beam search. Furthermore, all ROUGE scores are about as high as in beam search. Suggesting word order to be similar to the reference.

Based solely on the metrics, beam search is the best decoding method, followed by top-p sampling. Although one has to keep in mind, that the data used for testing does not have an extensive number of references and the ones provided often differ only slightly. This might explain the good scores of beams search, since it often produces very similar outputs.

Diverse beam search on the other hand does indeed seem to generate diverse outputs. It especially loses in scores that measure word order strictly and do not allow deviation from the reference.

The sampling methods perform worse on the WMT data, they do not seem to work well with longer sentences. The high repetition also makes them less suitable for generating alternative outputs. Nevertheless, the outputs of top-p sampling resemble the ones of beam search and it might pose a good alternative.

6.2 Discussion

Given the information of the scores, a few assumptions can be made regarding the diversity of sentences, translation quality, sentence structure and vocabulary use. In the following section these assumptions are discussed with the help of example translations that support them.

Sentence Structure

While all decoding algorithms should be able to produce different sentence structures in theory, the differences in ROUGE scores suggest, that diverse beam search in particular produces different structures. Beam search is often critiqued for producing very similar outputs and the high scores it achieved suggest that the structure of the outputs is similar to the ones of the references and have therefore less variety. The sampling methods reached good ROUGE scores even when the other scores were low, also indicating matches in sentence structure.

Looking at Table 6.2 we can see clearly that all algorithms are able to start the sentence in different ways, starting either with “There is a car” like the reference, or with “A car is”. However, DBS is the only method that produced a sentence starting with the second part of the sentence “In front of the church” and thereby reversing the sentence structure. This is also a more direct translation of the original German sentence.

The example also shows that the outputs of beams search and top-p sampling are almost identical, which was also indicated by the evaluation metrics. Top-k sampling on the other hand, produces slightly different outputs, some of which are good alternatives (the use of “vehicle” instead of “car”), and some include capitalization errors.

Another example shown in Table 6.3, indicates that beam search might indeed be the least flexible, when it comes to sentence structure. Every output starts with “I don’t” and only the translations for “only” and “study” differ. Diverse beam search manages to produce three completely different beginnings and both sampling methods have two different beginnings and also sometimes end the sentence with “but French as well” instead of “but also French”.

German	Vor der Kirche steht ein Wagen.
References	*There is a car in front of the church.* *There's a car in front of the church.*
BS	There is a car in front of the church. A car is in front of the church. A car is standing in front of the church.
DBS	A car is in front of the church. There is a car in front of the church. In front of the church is a car.
Top-K	a car is in front of the church. A car is in front of the church. There's a vehicle in front of the church.
Top-P	A car is standing in front of the church. A car is in front of the church. There's a car in front of the church.

Table 6.2: Example 1: Sentence Structure

German	Ich lerne nicht nur Englisch, sondern auch Französisch.
References	*I not only study English, but I also study French.* *I not only study English, but I study French, too.* *I'm not only studying English, but I'm also studying French.*
BS	I don't only study English, but also French. I don't just study English, but also French. I don't only learn English, but also French.
DBS	I don't only study English, but also French. I am not only learning English, but also French. Not only do I study English, but also French.
Top-K	I do not just study English, but also French. I am not only studying English, but also French. I don't just study English, but French as well.
Top-P	I do not just study English, but also French. I am not only studying English, but also French. I don't just study English, but French as well.

Table 6.3: Example 2: Diverse Sentence Structures

Vocabulary Use

Words can have multiple meanings or have synonyms or can be otherwise worded differently depending on the context. Some translations, although less likely according to the probability distribution of the NMT output, are good alternatives. Given that the sampling methods rely somewhat on randomness, they should have more variety when it comes to vocabulary use. Beam search, being a greedy algorithm is less likely to pick low probability words. This is also true for Diverse beam search, but the diversity penalty might force the algorithm to choose lower probability words. The METEOR score directly measures synonym and paraphrase matches and DBS had much more of them than beam search, suggesting a difference in vocabulary use.

German	Ich war erstaunt, als ich erfuhr, dass ich gewonnen hatte.
References	*I was amazed to learn I'd won.* *I was amazed to learn I had won.*
BS	I was astonished when I learned that I had won. I was astonished when I realized that I had won. I was astonished when I learned that I won.
DBS	I was surprised when I learned that I had won. When I realized that I had won, I was surprised. I was shocked when I realized I had won.
Top-K	I was astonished when I learnt that I had won. I was thrilled when I discovered that I had won. I was surprised when I found out that I had won.
Top-P	I was astonished when I heard that I had won. I was astonished when I found out that I won. I was astonished when I learned that I won.

Table 6.4: Example 3: Variety in Vocabulary Use

The third example (Table 6.4), shows a variety of different translations for the words “erstaunt” (eng. “amazed”) and “erfuhr” (eng. “learned”). Beam search has little variation. It always uses “astonished” and only replaces “learned” with “realized” once. Diverse beam search also uses “learned” and “realized”, but also varies between “surprised” and “shocked”, of which the former is a good synonym, and the latter might be acceptable in this case but has usually a more negative meaning. Similarly, the use of “thrilled” in the top-k output is an acceptable but somewhat more positively worded translation. Top-k uses the most different translations, producing three different words for both “amazed” and “learned”. The output of top-p is very similar to that of beam search, but has more variety in the second half, also using “found out” and “heard” as alternatives.

Of course, too much variety in the generated words can also hurt the translation quality, when wrong translations are chosen. Especially top-k sampling, having the lowest scores, is probably affected by such errors.

German	Russisch ist meine Muttersprache.
References	*Russian is my mother tongue.* *Russian is my native language.*
BS	Russian is my mother tongue. Russian is my native language. Russian is my first language.
DBS	Russian is my mother tongue. Russ is my mother tongue. Romanian is my mother tongue.
Top-K	French is my mother tongue. Russian is my mother tongue. It is my native language.
Top-P	Russian is my native language. Russian is my native language. Russian is my mother tongue.

Table 6.5: Example 4: Good and Bad Vocabulary Variation

The example shown in Table 6.5 is a relatively short sentence, that does not have too many different translations. The only possible variety is in the translation of "Muttersprache", which can be either "mother tongue", "native language", "first language" or also "native tongue".

Beam search is able to accurately display this variation, while all other algorithms fail to do so. The diversity penalty in diverse beam search seems to force the algorithm to pick different beginnings for the sentence leading to wrong translations of "Russian". Top-k samples from unlikely words and produces "French" or simply "it" instead of "Russian". While Top-p sampling produces the two references perfectly, it fails to produce a third valid translations and instead repeats itself.

Repetition

Since the goal was to create alternative translations, the repetition of outputs, which is a big problem in the sampling methods as indicated by the high precision, is damaging. It is likely to happen in short sentences, where the probability distribution is sharp, like the example in Table 6.6.

Since there is so little possibility for different translations, none of the algorithms produced really good outputs. Although beam search at least has three different outputs, while all others repeated themselves. Repetition is not such a big problem in DBS only 42 sentences were repeated in total. Usually, the diversity penalty should punish repetition, but if there are no good words to choose from DBS starts generating strange outputs, like repeated punctuation marks or additional white spaces. The sampling methods have a very high chance to just sample the same thing two or even three times.

German	Der Lehrer kam herein.
References	*The teacher came in.* *The teacher walked in.*
BS	The teacher came in. The teacher got in. The teacher came inside.
DBS	The teacher came in. The teacher came in. The teacher came in..
Top-K	The teacher came in. The teacher came in. The teacher came in.
Top-P	The teacher came in. The teacher came inside. The teacher came in.

Table 6.6: Example 5: Repetition

Quality

While all algorithms except for top-k sampling scored fairly well across the Tatoeba data, the BLEU scores for the WMT data are quite low. This can have several reasons. First, the data set only contains one reference sentence, which affects the score negatively due to less possible matches. The sentences are also much longer and more complicated, than the data the model was trained on. Furthermore, they are of a different domain and might contain out of vocabulary words, that the model has never seen before.

Nevertheless, the WMT sentences might provide a good overview of the overall translation quality of the different decoding methods, showing which method generates fluent sentences that also make sense. The sampling methods scored significantly lower on this data set, which suggest they are less likely to provide good translations when it comes to longer sentences or complicated vocabulary. The beam search methods, on the other hand, seem to produce a somewhat better quality, even though the diverse beam search output gets less reliable in the second and third translation.

The example in Table 6.7 shows translations of a sentence from the WMT data set. To correctly capture the meaning of the sentence the translation should include that the euro crisis is not caused by national governments creating budget deficits, except in Greece.

Beam search produces fluent sentences, that almost reproduce the meaning. It only fails to describe that the national governments are the ones causing budget deficits and not suffer from it. The same is true for diverse beam search, except for the last sentence, which has a completely different meaning and does not even mention national governments or budget deficits. This supports the assumption that the quality of translations decreases with more outputs in DBS.

The sentences of top-k sampling change the meaning even further, instead of "deficits" it produces "constraints" and "shortage", which are not the same thing. The second sentence even suggest a "monetary" crisis was prevented in Greece, which is a completely different meaning. Additionally, the output is less fluent as it contains grammatical errors, like using "Greek" instead of "Greece" and missing words in "only in Greece was the case".

German	Die Krise des Euro hat wenig damit zu tun, dass nationale Regierungen exzessive Haushaltsdefizite verursachen – das war nur in Griechenland der Fall.
References	*The crisis of the euro has little to do with national governments running excessive budget deficits - that was true only of Greece.*
BS	The crisis of the euro has little to do with national governments suffering from severe budget deficits – that was only the case in Greece. The Euro crisis has little to do with national governments suffering from severe budget deficits – that was only the case in Greece. The crisis of the euro has little to do with the fact that national governments suffer from severe budget deficits – that was only the case in Greece.
DBS	The crisis of the euro has little to do with national governments suffering from severe budget deficits – that was only the case in Greece. The Euro crisis has little to do with national governments suffering from severe budget deficits – that was only the case in Greece. There is little to be worried about the crisis of the euro, as it only happened in Greece.
Top-K	The crisis of the euro has little to do with national governments enduring severe budgetary constraints – only in Greece was the case. The monetary crisis has little to do with the failure of national governments to deal with the shortage of budgets – the only thing that was done in Greek was to prevent it from happening. The crisis of the euro has nothing to do with the national governments struggling with budget deficits – this was only the case in Greece.
Top-P	The Euro crisis has little to do with the national governments suffering from high budgets — it was only Greece's fall. The Euro crisis doesn't have much to do with national governments declaring extreme budget deficits, as in Greece. The crisis of the euro doesn't mean that national governments are suffering from deficits in budgets – this was just the case in Greece.

Table 6.7: Example 6: Translation quality in complicated sentences

Top-p sampling produces sentences that are a little more fluent and better reproduce the meaning, then the ones of top-k sampling. In the first sentence, the wrong translation of the German word "Fall", which can mean either "case" or "fall", is chosen. Also “high budget” is almost the opposite of “budget deficit” and therefore wrong. The second and third translations are better, although “as in Greece” is oddly worded.

All methods have advantages and disadvantages. Beam search has the overall best quality, but it is less diverse. It is very good at producing multiple high-quality sentences, that differ slightly in wording, use abbreviated and unabbreviated forms correctly (e.g. “you’re” and “you are”), change the vocabulary used a little and sometimes even provide some variety in sentence structure. Diverse beam search produces similar outputs at first but then provides much more variation in the next outputs. It is especially good at producing different sentence structures but has the disadvantage of decreasing quality when there are not many options to word things differently.

The advantages of both methods could be combined by running the diverse beam search algorithm with a different set of parameters (see Table 6.8). Setting $B > G$, allows for beam search to run normally within a group, while the diversity penalty ensures differences between the groups. This might produce a set of translations with both the little nuances in wording of beam search and the different sentence structures of diverse beam search.

German	Von diesem Buch gibt es noch keine französische Ausgabe.
References	*This book still hasn’t been translated into French.* *There’s still no French translation of this book.*
DBS	There’s no French edition of this book yet. There is no French edition of this book yet. There’s no French edition of this book. This book has no French edition. There is no French edition of this book. There’s no French edition of this book yet! This book isn’t yet translated into French. This book isn’t translated yet. This book isn’t available in French yet.

Table 6.8: Example 7: Diverse Beam Search with $B = 9$ and $G = 3$

Top-k sampling has the most creative vocabulary use, which can be very good in some cases, but hurts the quality in others. Either the words are translated wrong, or it produces a completely nonsensical output. Top-p sampling on the other hand, is surprisingly good and often produces very similar outputs to beam search. It even has some more creative vocabulary use and variety in sentence structure. Occasionally this hurts the translation quality, especially on longer sentences, but works very well on shorter sentences like the one used on Tatoeba. Both sampling methods have the disadvantage of having very high repetition when sampling independently multiple times. This could be improved by penalizing tokens that have been sampled before, or otherwise force the algorithm to produce different outputs.

7 Conclusion

In this thesis a neural machine translation model, that can produce multiple alternative translations, has been developed. With the help of transfer learning the transformer model could be trained with a relatively small amount of data and in a short amount of time. During training, the model was explicitly shown alternative translations for the given input, in order to produce variance in the output. Although the translation quality of the network does not compare with other networks of similar size on standard test data, that has longer sentences and is of a different domain, the model has shown to be able to produce correct alternative translations on short sentences of the same domain as the training data.

The analysis in this thesis has shown, that the algorithm chosen to decode the model output, has a noticeable effect on the translations produced. This can be seen in a variety of different automatic scoring metrics for the evaluation of machine translation, as well as in human evaluated examples.

The specific algorithms investigated in this thesis all produce different variations within a set of generated translations. Beam Search produces the best quality sentences overall, but they only differ slightly in word choice and structure or produce the same sentence with and without contractions. Diverse beam search shows promising results in producing different sentence structures but suffers in quality when it exhausted the possible word orders, resulting in either grammatical or semantic errors. Top-k sampling has the most creative vocabulary use, which can result in interesting and less direct translations, but most of the time it results in nonsensical translations, that do not convey the same meaning and are syntactically incorrect. Top-p sampling is very comparable to beam search and often produces similar sentences. Although it is somewhat less accurate, it is more diverse when it comes to word choice and structure.

In future work, other decoding methods, that have not been included in this thesis could be investigated. The sampling methods can also be combined or altered with a parameter like the temperature to further improve their output. Furthermore, penalties during sampling or a filtering method could be implemented to solve the problem of repetition. Over-sampling and filtering can be used on sampling as well as beam search-based methods to get a more diverse set of translations. Given the clear benefit of beam search producing correct translations more consistently and sampling producing more diverse and interesting outputs, another option might be to combine the two approaches in order to have both of these advantages. Stochastic beam search is a variant of beam search, that performs beam search over stochastically perturbed log-probabilities and can be seen as a connection of sampling and beam search. Therefore, it could be another interesting decoding algorithm for producing multiple alternative translations.

It should also be kept in mind, that the results of this thesis depend on the test data that has been used. For a more conclusive interpretation, further testing with better data is necessary. Data sets for evaluating machine translation should have a more extensive set of references, in order to accurately measure the performance of a model. Since there is not one single correct translation, the data used for judging translation quality should represent that fact.

On a related topic, further research should also focus on the automatic evaluation of machine translation. Different metrics that are intended to measure translation quality have been shown to vary considerably on the same set of candidates and references. The way the scores a calculated, means that they represent different parts of the similarity between candidate and reference sentence. While some focus more on over-all structure, others focus more on word matches. Capturing the complex aspects of language that differentiate good from bad translations in a numerical score is a difficult task. Research in this direction has to include a way to incorporate multiple different translations in a meaningful way and also try to make scores more consistent and comparable. This would allow further improvement in the quality of machine translation to be more measurable.

Machine translation has not yet reached a human level. Better translation systems can help people all over the world to communicate with each other. Although the quality of these systems has significantly increased with the rise of neural machine translation, the complexity of natural languages and the vast differences between them will continue to make machine translation a challenging task.

List of Figures

List of Tables

Bibliography

[Bahdanau et al. 2014] Bahdanau, Dzmitry; Cho, Kyunghyun; Bengio, Yoshua: Neural Machine Translation by Jointly Learning to Align and Translate. (2014). https://arxiv.org/abs/1409.0473

[Banerjee and Lavie 2005] Banerjee, Satanjeev; Lavie, Alon: METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In: *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, 2005

[Cho et al. 2014] Cho, Kyunghyun; Merriënboer, Bart van; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua: Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, 1724–1734

[Denkowski and Lavie 2014] Denkowski, Michael; Lavie, Alon: Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, 376–380

[Eberhard et al. 2021] Eberhard, David M.; Simons, Gary F.; Fenning, Charles D.: *Ethnologue: Languages of the World*. https://www.ethnologue.com/. edition: 24, 2021, Accessed: 06.07.2021

[Falcon 2019] Falcon, WA: *PyTorch Lightning*. https://github.com/PyTorchLightning/pytorch-lightning/. 2019, Accessed: 23.06.2021

[Fan et al. 2018] Fan, Angela; Lewis, Mike; Dauphin, Yann: Hierarchical Neural Story Generation. (2018). https://arxiv.org/abs/1805.04833

[Hochreiter 1991] Hochreiter, Josef: *Untersuchungen zu dynamischen neuronalen Netzen*. München, Germany, Technische Universität München, Diplomarbeit, 1991

[Hochreiter and Schmidhuber 1997] Hochreiter, Sepp; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Computation* 9 (1997), November, pages 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735. – DOI 10.1162/neco.1997.9.8.1735. – ISSN 0899–7667

[Holtzman et al. 2019] Holtzman, Ari; Buys, Jan; Du, Li; Forbes, Maxwell; Choi, Yejin: The Curious Case of Neural Text Degeneration. (2019). http://arxiv.org/abs/1904.09751

[Internet World Stats 2021] Internet World Stats: *INTERNET WORLD USERS BY LANGUAGE*. https://www.internetworldstats.com/stats7.htm. 2021, Accessed: 06.07.2021

[Kudo and Richardson 2018] Kudo, Taku; Richardson, John: SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, 2018, 66–71

[Lin 2004] Lin, Chin-Yew: ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*, Association for Computational Linguistics, July 2004, 74-81

[Lin and Och 2004] Lin, Chin-Yew; Och, Franz J.: Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2004, 605–612

[Luong et al. 2015] Luong, Thang; Pham, Hieu; Manning, Christopher D.: Effective Approaches to Attention-based Neural Machine Translation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, 1412–1421

[Mayhew et al. 2020] Mayhew, Stephen; Bicknell, Klinton; Brust, Chris; McDowell, Bill; Monroe, Will; Settles, Burr: Simultaneous Translation and Paraphrase for Language Education. In: *Proceedings of the Fourth Workshop on Neural Generation and Translation*, Association for Computational Linguistics, July 2020, 232–243

[McKinney 2010] McKinney, Wes: Data Structures for Statistical Computing in Python. In: Walt Stéfan van d. (ed.); Millman Jarrod (ed.): *Proceedings of the 9th Python in Science Conference*, 2010, pages 56–61

[Mikolov et al. 2013] Mikolov, Tomas; Chen, Kai; Corrado, Greg; Dean, Jeffrey: Efficient estimation of word representations in vector space. In: *1st International Conference on Learning Representations, ICLR 2013* (2013), January. https://arxiv.org/abs/1301.3781

[Pan and Yang 2010] Pan, Sinno J.; Yang, Qiang: A Survey on Transfer Learning. In: *IEEE Transactions on Knowledge and Data Engineering* 22 (2010), no. 10, pages 1345–1359. http://dx.doi.org/10.1109/TKDE.2009.191. – DOI 10.1109/TKDE.2009.191

[Papineni et al. 2002] Papineni, Kishore; Roukos, Salim; Ward, Todd; Zhu, Wei-Jing: BLEU: A Method for Automatic Evaluation of Machine Translation. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. USA: Association for Computational Linguistics, 2002, 311–318

[Paszke et al. 2019] Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Kopf, Andreas; Yang, Edward; DeVito, Zachary; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie; Chintala, Soumith: PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. In: Wallach, H. (ed.); Larochelle, H. (ed.); Beygelzimer, A. (ed.); Alché-Buc, F. d'(ed.); Fox, E. (ed.); Garnett, R. (ed.): *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, 8024–8035

[Pennington et al. 2014] Pennington, Jeffrey; Socher, Richard; Manning, Christopher D.: GloVe: Global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, 1532–1543

[Post 2018] Post, Matt: A Call for Clarity in Reporting BLEU Scores. In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, October 2018, 186–191

[Raffel et al. 2020] Raffel, Colin; Shazeer, Noam; Roberts, Adam; Lee, Katherine; Narang, Sharan; Matena, Michael; Zhou, Yanqi; Li, Wei; Liu, Peter J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. In: *Journal of Machine Learning Research* 21 (2020), no. 140, 1–67. http://jmlr.org/papers/v21/20-074.html

[Ruder 2019] Ruder, Sebastian: *Neural Transfer Learning for Natural Language Processing*. Galway, Ireland, National University of Ireland, PhD thesis, February 2019

[Ruder 2020] Ruder, Sebastian: *Why You Should Do NLP Beyond English*. http://ruder.io/nlp-beyond-english. 2020, Accessed: 06.07.2021

[Shazeer and Stern 2018] Shazeer, Noam; Stern, Mitchell: Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. (2018), April. http://arxiv.org/abs/1804.04235

[Vaswani et al. 2017] VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, undefinedukasz; POLOSUKHIN, Illia: Attention is All You Need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* Long Beach, California, USA: Curran Associates Inc., 2017. – ISBN 9781510860964, pages 6000–6010

[Vijayakumar et al. 2018] VIJAYAKUMAR, Ashwin K.; COGSWELL, Michael; SELVARAJU, Ramprasath R.; SUN, Qing; LEE, Stefan; CRANDALL, David; BATRA, Dhruv: Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. (2018). https://arxiv.org/abs/1610.02424

[Wikipedia contributors 2021] WIKIPEDIA CONTRIBUTORS: *List of Wikipedias — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/wiki/List_of_Wikipedias. 2021, Accessed: 06.07.2021

[Wolf et al. 2020] WOLF, Thomas; DEBUT, Lysandre; SANH, Victor; CHAUMOND, Julien; DELANGUE, Clement; MOI, Anthony; CISTAC, Pierric; RAULT, Tim; LOUF, Rémi; FUNTOWICZ, Morgan; DAVISON, Joe; SHLEIFER, Sam; PLATEN, Patrick von; MA, Clara; JERNITE, Yacine; PLU, Julien; XU, Canwen; SCAO, Teven L.; GUGGER, Sylvain; DRAME, Mariama; LHOEST, Quentin; RUSH, Alexander M.: Transformers: State-of-the-Art Natural Language Processing. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Online: Association for Computational Linguistics, October 2020, 38–45

[Wu et al. 2016] WU, Yonghui; SCHUSTER, Mike; CHEN, Zhifeng; LE, Quoc V.; NOROUZI, Mohammad; MACHEREY, Wolfgang; KRIKUN, Maxim; CAO, Yuan; GAO, Qin; MACHEREY, Klaus; KLINGNER, Jeff; SHAH, Apurva; JOHNSON, Melvin; LIU, Xiaobing; KAISER Łukasz; GOUWS, Stephan; KATO, Yoshikiyo; KUDO, Taku; KAZAWA, Hideto; STEVENS, Keith; KURIAN, George; PATIL, Nishant; WANG, Wei; YOUNG, Cliff; SMITH, Jason; RIESA, Jason; RUDNICK, Alex; VINYALS, Oriol; CORRADO, Greg; HUGHES, Macduff; DEAN, Jeffrey: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. (2016). https://arxiv.org/abs/1609.08144

[Young 2015] YOUNG, Holly: *The digital language divide.* http://labs.theguardian.com/digital-language-divide/. 2015, Accessed: 06.07.2021

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Ort, Datum	Melina Zanon