Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Comparación entre modelos estadísticos y neuronales para la traducción automática.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: José Manuel Llorens Ripollés

Tutores: Francisco Casacuberta Nolla
Pawel Herman

2017-2018

Final Degree Project

Comparison between statistical and neuronal models for machine translation.

Author:
José Manuel Llorens Ripollés

Supervisors:
UPV: Francisco Casacuberta Nolla
KTH: Pawel Herman

Home School: **Escuela Técnica Superior de Ingeniería Informática**
Home University: **UPV - Universitat Politècnica de València**
Host School: **School of computer science and communication**
Host University: **KTH - Royal Institute of Technology**
Date: **May 2018**

Contents

Abstract

Machine translation is a thriving field that deals with multiple of the challenges that the modern world face. From accessing to knowledge in a foreign language, to being able to communicate with people that does not speakthelanguage, we can take great benefit from automatic translation made by software. The state-of-the-art models of machine translation during the last decades, based of inferred statistical knowledge over a set of parallel data, had been recently challenged by neural models based on large artificial neural networks.

This study aims to compare both methods of machine translation, the one based on statistical inference (SMT) and the one based on neural networks (NMT). The objective of the project is to compare the performance and the computational needs of both models depending on different factors like the size of the training data or the likeliness of language pair.

To make this comparison I have used publicly available parallel data and frameworks in order to implement the models. The evaluation of said models are done under the BLEU score, which computes the correspondence of the translation with the translation made by a human operation.

The results indicate that the SMT model outperform the NMT model given relatively small amount of data and a basic set of techniques. The results also shown that NMT have a substantially higher need of processing power, given that the training of large ANN is more demanding than the statistical inference.

Acknowledgements

I would like to thank Francisco Casacuberta Nolla for introducing me to the fascinating world that machine translation is, as well as for answering my multiple questions on the theory and the practice.

I would also like to thank Pawel Herman for taking me as a tutee and for its extremely valuable feedback through the whole project.

Chapter 1

Introduction

Machine translation (MT) is the use of software to produce fully automatic translation between languages. The ability to use software to translate between languages has been an early goal for computer scientists, present almost since the invention of the firsts electronic computers[1]. The different languages represent a barrier to knowledge, and to be able to break those barriers could be very beneficial in a large number of scenarios. At its early days MT was backed mostly by governments, as the ability to translate from languages of foreign nations that represent a thread to national security, economy and so on, could be a decisive competitive advantage and highly beneficial for a nation. However, we now live in a more-than-ever global world, a trend that seems likely to keep growing, and MT has also found a wide array of uses in the public sector.

Nowadays the most widespread use of MT is gisting, which can be defined as "looking for the main idea or most important point in a written or spoken text". One example of the huge use of MT for gisting is the Google Translate service, used by 500 million users, that translate more than 100 billion words each day. One of the key points of MT in gisting is that it does not need to provide a perfect translation, as long as the translation is good enough for the user to extract relevant information. However, MT has also found more applications like being integrated with a speech recognition system for uses such as telephone conversation translation[2][3], being integrated with hand held devices, providing tools like translation for medical aid in developing countries[4] or tourism. Other important consideration is that MT can be used only as a first layer, obtaining a rough translation that

can be fine-tuned by a human operator[5] in what is known as post-script translation.

MT has used different methods and approaches since its inception. However, in the last few decades, MT has evolved to reflect the idea that language is too rich and complex to be analysed and translated into a set of rules. Accordingly, new paradigms originated, instead of a rule-based approach this new paradigms aim for the machine to learn the rules of translation automatically from a large corpus of translated text, usually called corpora[6]. This are called Data-Driven Methods and nowadays they are the most studied methods for MT and the basis of this report.

The statistical models for MT (SMT) base their translation on the statistics estimated from the data and had been some of the models with higher adoption over the last decades[6]. However, with the rise of neural models in computer science we have seen a rapid progress in the Neural MT (NMT), with big companies like Google shifting their services from the previous statistical methods to NMT. This neural approach relies on large neural networks as a base for the translation[7]. The neural networks are trained with the parallel corpora and then assign a translation for a sentence based in the likelihood of the translation. By 2017 most of the best MT systems had used neural networks [8]. The aim of this project is to compare the traditional SMT with the new NMT models.

1.1 Problem Statement

In this project I will compare different models of MT, one based on statistical inference of phrase translations and other based on the use of ANNs. In particular, the aim of this study is to compare and examine key differences between SMT and NTM in various translation contexts. Specifically, this comparison address the following points:

- Performance difference depending on the amount of data.
- Performance difference depending on language pairs.
- Scaling properties

1.2 Scope and objectives

In this study I compare different paradigms of MT. The objective of the project is not to fine-tune the models but rather to provide a comparison framework which highlights the differences between the models. Specially, this study will focus on how do these models perform and scale given the amount of data used for training

The performance of both methods will be examined by the quality of their translation, which will be evaluated by its correspondence with the translation made by a human operator under the BLEU scoring metric[9].

The escalation properties of the SMT model will be examined by the CPU time required for the model to train with increasingly bigger training data sets. I will examine the NMT scaling properties by the GPU time required for its training with increasingly bigger training data sets. I will also provide CPU times for the training in NMT models with small datasets. The provided CPU times of the NMT can easily be used to estimate CPU times for bigger training sets, as the training time of an ANN is linear with the size of the training data, and therefore can be used to estimate values for the comparison with SMT models.

I will create two pair of models, one to translate from German to English, and one to translate from Russian to English. The objective of this is to examine if there are significant differences between the models performance and scaling properties given different languages pairs.

Due to computational resources limits, the NMT model will be restricted to conventional LSTM networks, the data will not be preprocessed with byte pair encoding and the model will not implement back-translation techniques. Due to the resources limits stated, this study is restricted to a 284247 parallel sentences dataset for the German to English model and a 235160 sentences dataset for the Russian to English model. Therefore, the comparison can only draw conclusions for this set of limitations and environment and they can not be extrapolated to bigger sizes of training data or different models.

1.3 Thesis outline

The rest of this report is organised as follows: In the second chapter the main background needed for the understanding of the report will be given. This background comprehends from natural language processing terminology and overview or artificial neural networks to more specific MT related background. The third section explains how the experiment pipeline was made and how the results were colected and evaluated. The fourth section presents the results of the experiments. The fifth section discuss the presented results and the limitations of the project. Finally, in the sixth section, a conclusion for the project is stated.

Chapter 2

Background

2.1 Words Sentences and Corpora

2.1.1 Words

Words are the basic atomic unit of meaning. However, although the concept of word seems simple in the Latin languages, as they are separated with spaces, the concept can become more difficult in other languages like Chinese were there are no spaces to separate each word. **Tokenization** is the process to break raw text into words and an important pre-processing for MT, as we need to feedthealgorithms with words, not with the raw text.

One interesting characteristic of words over a text is their distribution. In any text big enough we can observe that the frequency of any word is inversely proportional to its rank in the frequency table[10]. This empirical law implies that the product of the rank of each word an its frequency is roughly a constant. The most common words over a text are usually **function words** like *the, of, to, and*. This function words exist to explain or create grammatical or structural relationships into which the content words may fit. On the other hand we have **content words** that are words that have meaning like *Book, Sofa, Glass*. While the number of unique function words is usually upper-bounded, the number of unique content words can grow without limits as the text grows.

The relationship between the words of a language and how they are formed is studied under the morphology of the language. For example we can change the meaning of a word by slightly changing its

spelling as in 'houses' were we have added a 's' to change the meaning of 'house'. "Morphology creates various challenges for MT. While some languages express the relationships between words mostly with location of words, others use morphological variation of words. The appropriate transfer of relevant information is not straightforward"[6].

Finally we have the lexical semantics. When translating words we will face homonymy, meaning that some words are spelled the same way although they have completely unrelated meaning, and polysemy, words with different meaning. The task to determine the meaning of the word in a given context is called **word sense disambiguation**.

2.1.2 Sentences

Sentences are aggregations of words that, together, have a new meaning like expressing an idea or a past event. The sentences are structured and have a main verb and usually at least a subject as well as other components. One striking characteristic of language is that components of a sentence can be extended with other components in a recursive way, enabling arbitrarily deep structures. This recursion leads to some problems such as **structural ambiguity**: In the sentence "Jim eats steak with ketchup" we can't know if ketchup belongs to the verb (eating happens with the ketchup) or to the steak (the steak has ketchup on it). Structural ambiguity usually don't represent a problem for humans, as we solve the ambiguity with the semantic meaning (Eating with the ketchup makes no sense) but can present a challenge for MT systems. There different grammar formalism are studied under the Theories of Grammar.

2.1.3 Corpora

In both SMT and NMT we need a corpora of translated text to trainthemodel. One important concept is the **domain** of the corpora. MT is a difficult task, specially if the model to develop is of general propose. Instead of this general propose models it is usual to restrict the freedom of the translation text forthemodels to certain domains. On example would be to trainthemodel for scientific translation, were we can achieve a high performance. If after trainingthemodel for the scientific domain we try to use it to translate room-chat conversations we will

probably get bad results. MT models for specific domains, for example weather forecast, had been implemented with high performance [11].

Other characteristic of the text that have influence inthemodel would be the modality of the text and the topic. If we are trainingthemodel in written scientific papers, the text will be very different as if we were doing so with spoken transcripts, often ungrammatically and full of unfinished sentences. As for the topic we won't translate "book" as the same word if we are translating an airline website than if we are translating a library one.

Once we havetheparallel corpus there is one further step we need to take in order for the corpora to be useful: **sentence alignment**. The language is not always translated sentence by sentence, as some of them may be broken up, merged etc. Sentence alignment is an important task in SMT preparation and therefore had been widely researched and formalised.

2.2 Probability theory

Probability theory is the branch of mathematics used when the outcomes are not certain and many possibilities exist. Probability theory deals with uncertainty in a strict mathematical manner, expressing it through a set of axioms. In this section of the background I will give a simplified overview of the field centred in the knowledge useful for MT.

2.2.1 Common probability distribution

A probability distribution is a function than, in a simplified way, can provide us with the probability of the possibles outcomes for an experiment. In a more formal way a probability distribution can be stated as "A function of a discrete variable whose integral over any interval is the probability that the variate specified by it will lie within that interval."

Probability theory provide us with some common distribution that we can use. For example we have the **uniform distribution**, in which all the possible outcomes have the same probability.

One of the most important probability distributions is the **normal distribution**, also known as the bell curve or Gaussian distribution.

The Gaussian distribution follows the next formula:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where μ is the mean and σ is the variance. Notice that the Gaussian distribution is a probability density for a real-valued values of x. If we want to know the probability of the experiment outcome to be between an interval we need to integrate the area under the curve of that interval.

Usually we can't derive estimates about probability distributions by inspection alone. For example we won't be able to predict tomorrow weather by simple reasoning, but we need to gather facts and statistics and buildtheprobability distribution from them. In the same normal distribution we need to know which is the average and the standard deviation oftheexperiments. For example if we suspect that a coin may be crooked we can toss it 1 million times. If in that 1 million experiment 7 hundred thousand times we get heads we can compute the probability of heads as $700.000/1.000.000 = 0.7$.

One important notion when gathering probability distributions over statistics is that we need the data sample to be big enough to be representative of the true distribution. If we want to know if a coin is crooked but we only toss the coin once, either it results heads or tails, it won't give us enough information on the underlying distribution. This phenomena is known as the **law of large numbers** that states thattheprobability estimates gets increasingly better the more counts we collect.

2.2.2 Calculation of probability distribution

We can define a random variable X that may have several different values, with the probability function p providing us with how likely is each value of X. Any probability function needs to meet two different properties:

- $\forall x : 0 \leq p(x) \leq 1$
- $\sum_x p(x) = 1$

Usually we want to deal with multiple random variables at the same time. We will now study the relationship between those variables and

how to extract information and calculate probability distribution over them.

- Joint Probability: If we have to random variables A and B, we can define the joint probability distribution as p(A=a, B=b) usually abbreviated p(a,b). This joint probability distribution is the probability of getting both, the 'a' outcome for the A random variable and the 'b' outcome for the B random variable. If both variables are completely unrelated like the cast of a dice and the flip of a coin, we can easily deduce that p(a,b) = p(a) * p(b). However we will usually have variables that are somehow related to each other and in that case we will need other techniques to calculate the joint probability.

- Conditional Probability: The conditional probability is what gives us the information about how the variables relate with each others. Let's say that we want to know the $p(a)$ given b, this is denote as $p(a|b)$ and it is defined as:

  $$p(a|b) = \frac{p(a,b)}{p(b)}$$

  we can reformulate the conditional probability in what is known as **chain rule**:

  $$p(a,b) = p(a|b) * p(b)$$

- Bayes Rule: From the conditional probability and the chain rule we can formulate the **Bayesian rule**:

  $$p(a|b) = \frac{p(b|a) * p(a)}{p(b)}$$

  With the Bayesian rule we can express $p(a|b)$ in terms of its inverse $p(b|a)$ which is called the **posterior** and the elementary distributions $p(a)$ and $p(b)$(the **priors**)

- Interpolation: In practice we often find that we have different ways to estimatetheprobability distribution. We can use **interpolation** to mix them in a weighted manner. This a known strategy in the machine learning community known as ensemble learning.

2.2.3 Properties of Probability distribution

- Mean: the mean of a data sample is calculated as $\overline{x} = \frac{1}{n}\sum_i x_i$
- Variance: geometric mean of the difference between each event's value and the mean: $\sigma^2 = \frac{1}{n}\sum_i (x_i - \overline{x})^2$

2.3 Artificial Neural Networks

ANNs are computing systems that get some inspiration from the biological neural network. The use of this ANNs is to learn and apply learn knowledge on tasks such as classification or regression. The key point of the ANN is that they learn by considering examples and usually without a task-specific programming. ANNs have found a wide variety of uses such as image recognition, classification, regression and so on.

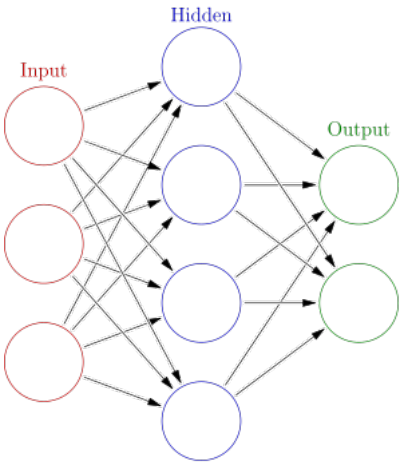

Figure 2.1: Example of a simple ANN.
Figure taken from the Wikipedia's article on Artificial Neural Network

ANN are based on units or nodes also called neurons. This neurons receive an input, change their internal stated with an activation function dependant on the input and produce an output that can be used by other units. The units get connected in a weighted way on a network, that can be also understood as a directed weighted graphs. The topology of the graph is called **architecture** of the network.

The inputs to a node are multiplied by their weights and added all together. Once we have this weighed added inputs, we apply the activation function of the neuron to obtain the output (or activation) of the neuron.

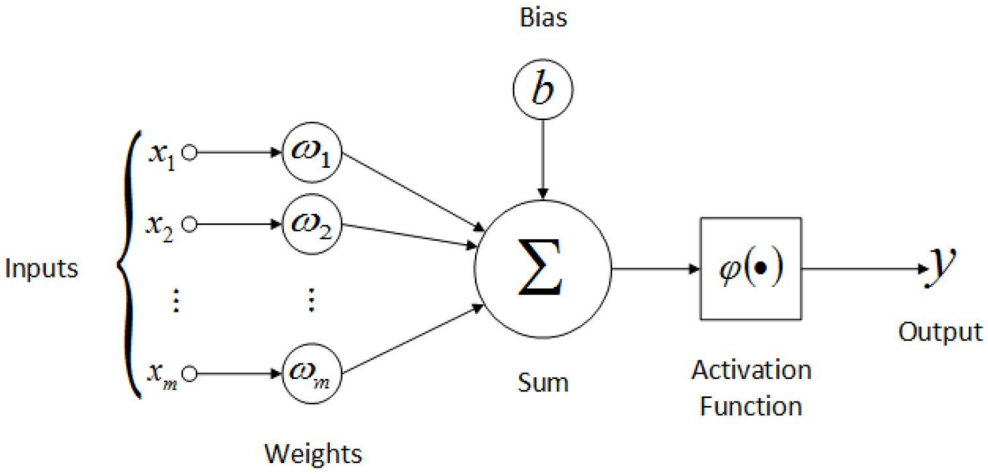

Figure 2.2: Example of a artificial neuron.
Figure taken from [12]

One important consideration when designingtheneurons is the activation function. There are a lot of mathematical functions that can be suited forthenetwork and some of the most representatives are the *threshold function*, *hyperbolic tangent* and one of the most frequently used the *Sigmoid function*

$$f(x) = sgn(\sum_{i=1}^{n} w_i x_i - \theta) \qquad f(x) = \frac{e^x}{e^x + 1} \qquad f(x) = \frac{e^x - e^{-x}}{e^x + e^{-1}}$$

(a) Threshold function (b) Sigmoid function (c) Hyperbolic tangent

The learning of the network occur by changing the weights between connections. There are different learning techniques for the weights but the Backpropagation(BP) is the mos usual one. BP is follows the next scheme:

1. We define a error function fortheneurons

2. We derive the error of the neuron with respect to its weights in order to know which is the direction in the hyperplane of the weights were the error increases the most.

3. We move in the contrary direction, minimising the error by a step controlled by the error and a learning rate we set arbitrarily

4. we repeat pass 2-3 a defined number of times or until we reach convergence or other stop condition.

At the beginning of a backpropagation step we can only compute the error in the output layer, as we can compare it with the desired output. Once we calculate the error in the output layer we back-propagate it to the prior layer, making them able to calculate their error and pass it back.

2.3.1 Feedforward Neural Network

A feedforward neural network (FFNN) is an ANN where the connections between unis do no form a cycle. This are the simplest ANN, were information move only forward. First of all we have the input layer, the set of nodes connected to the input, which transmit the info forwards. This starts a chain of forward transmissions through the n hidden layers, until the information reaches the output layer (set of nodes that we measure and which output is not connected to other neurons). We can observe a basic FFNN in figure 2.4 which represents a multilayer perceptron (MLP), the most common feedforward ANN.

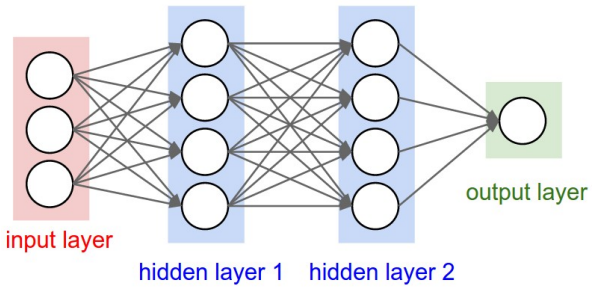

Figure 2.4: Example of a FFNN.
Source *https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/*

2.3.2 Recurrent Neural Networks

The Recurrent Neural Networks (RNN) can be seen as a generalisation of FFNN were we allow cycles in the architecture of the network. This characteristic allows RNN to exhibit dynamic temporal behaviour for long sequences and this makes RNN specially useful for some task such as speech recognition. An example of an unfolded RNN can be observed in figure 2.5

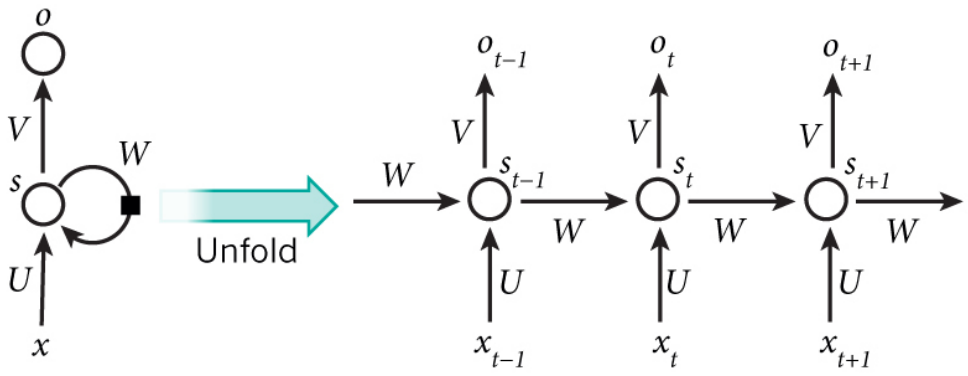

Figure 2.5: Example of a RNN unfolded.
Source: *http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/*

2.4 Long short-term memory

Long short term memory cells are building blocks for RNN[13]. When a RNN is composed of LSTM cells it is known as a LSTM network. The regular LSTM cell is composed of three gates:

- Input gate
- Output gate
- Forget gate

The three of this gates can be seen as conventional perceptrons units. The main capacity of a LSTM networks is its ability to learn long-term dependencies, which make them useful in different scenarios like machine translation[13]. An example of an LSTM cell can be observed at 2.6

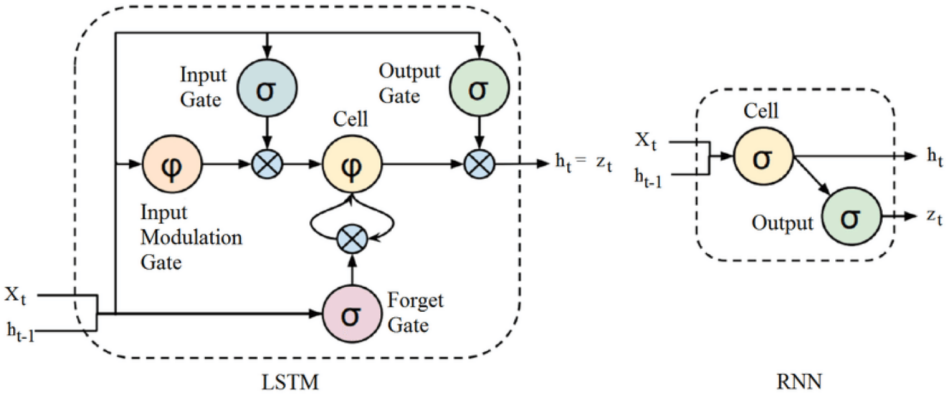

Figure 2.6: Example of a LSTM cell.
Image taken from [14]

2.5 Phrase-based SMT

2.5.1 Word alignment

In order to createthephrase based SMT model we have parallel text. However, this parallel text is not enough, as we need to align the words between sentences. Sentences are not aligned word by word, as different languages place words in a different way. In fact, when translating a sentence, words may not only change position but even disappear of get inserted.

The word alignment is not given and, consequently,thedata is incomplete. Having incomplete data is a common situation in data science and there are a few algorithms designed to deal with it. Inthedata there are hidden relationships, the alignment between words, and consequently the word alignment can be seen as a **latent variable**. We will use the Expectation-Maximisation[15] algorithm in order to get the word alignment.

2.5.2 Phrase extraction

If we watch closely the figure 2.7, we can appreciate that "geth davon aus, dass" could be translated as "assumes that". This is what is known as phrase-translation, were the translation units no longer need to be words but can be a few (or more) consequent words. The phrase ap-

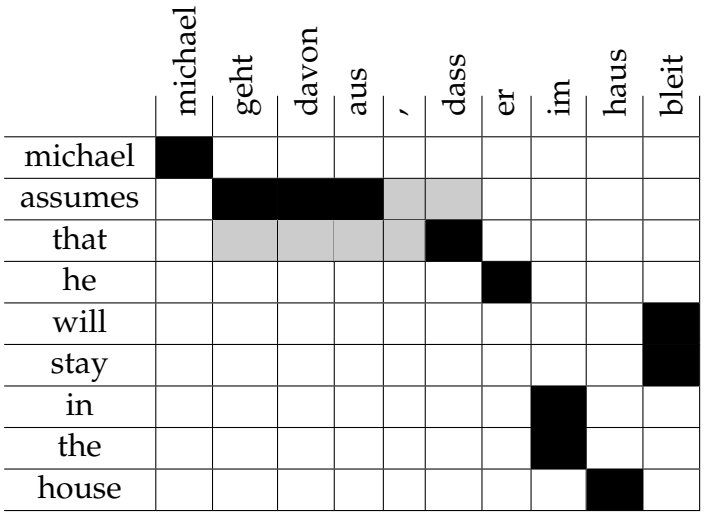

Figure 2.7: Example of word alignment.
Image taken from [6]

proach has many advantages like helping to resolve ambiguities or adding context to the translation.

Given a word alignment A we will define a phrase pair $(\overline{f}, \overline{e})$ as **consistent** with it if all three conditions are meet:

1. $\forall e_i \in \overline{e} : (e_i, f_j) \in A \rightarrow f_j \in \overline{f}$
2. $\forall f_i \in \overline{f} : (e_i, f_j) \in A \rightarrow f_e \in \overline{e}$
3. $\exists e_i \in \overline{e}, f_j \in \overline{f} : (e_i, f_j) \in A$

Notice that some words do not have alignments and consequently they can not violate the consistency of the phrase pair. Therefore {"asumes", "gerh davon aus"} and {"asumes","geth davon aus ,"} are both valid phrase pairs. Some examples of inconsistent phrase pairs would be:

- {"michael asumes","michael geht davon"}: the word "asumes" is aligned to "aus" but this is not in the German sentence, breaking the first rule of consistency.
- {"will","bleit"}: the word "bleit" is aligned with "stay" but this word is not in the English phrase, breaking the second rule of consistency.

Once we have defined the concept of consistency, given a word alignment the idea is to loop over all the possible source language phrases in a source sentence and match them with the minimal target phrase that make a consistent phrase pair. We will keep all the

matched phrases pairs over the whole text and then we will use them to build the **phrase translation table**. There are some extra considerations like, if the matched target phrase borders with an unaligned word, we will match the source with both, the original target phrase and with the original target sentence extended with the unaligned word[6].

2.5.3 Phrase translation probabilities

We have established how to extract phrases pairs from a sentence pair. We now need to establish the probability of translating the phrase $\overline{f}$ into the phrase $\overline{e}$, which is represented by $p(\overline{e}|\overline{f})$. This probability is inferred with the relative frequency:

$$p(\overline{e}|\overline{f}) = \frac{count(\overline{e}, \overline{f})}{\sum_{\overline{f_i}} count(\overline{e}, \overline{f_i})}$$

Where $count(\overline{e}, \overline{f})$ returns the number of sentences pairs where the phrase pair $(\overline{e}, \overline{f})$ is extracted.

2.5.4 Translation

Once we havethephrase translation table we can build a search algorithm to buildthepossible translations, also known as hypothesis, and then evaluate them to chose the most likely hyposthesis. Even with a small model, the search can be too big to explore in its totality and SMT usually resorts to heuristic algorithms like *beam search*. The *beam search* can be defined as "An optimisation of the best first graph search algorithm were only a predetermined number of paths are kept as candidates"[1]. The current state-of-the-art in beam search is to generate the target sentence word by word, from left to right while keeping a fixed amount of candidates at each time step[16]

[1]http://foldoc.org/beam+search

2.6 Neural Machine Translation

2.6.1 Word representation

One important question in NMT is how to represent words. We need to feed or ANN with a vector of real numbers, but words are discrete items over a large vocabulary. One used approach to transform the words to the desired vector is to use the *bag-of-words* model, which assigns one dimension to the vector for per word in the vocabulary. In a *bag-of-words* the representation of a word will be a vector with all 0 except the dimension of the word itself. Once we have the words represented as a vector with or *bag-of-words* model, we will introduce a extra layer between the *bag-of-words* vectors and the ANN. This extra layer is know as **word embedding** and its objective is to transform the *bag-of-words* vector to another vector, usually with lower dimensionality, where words that occur in similar context are presented near each other in the hyper-space represented by the vector. There are different techniques to obtain the embedding, but in this project we will use the *word2vec*[17] technique developed by a Google team led by Tomas Mikolov.

2.6.2 Neural Language Models

Neural Language Models are models that, given the *n* previous words in a sentence, predict the probabilities of the words to be the next word in the sentence. We can express formally the probability given by the models as $p(w_i|w_{i-1}, w_{i-2}, w_{i-3}, w_{i-4})$, where w_i refers to the *i-th* word in a sentence. An example of this kind of Languages Models can be seen in figure 2.8

Instead of using a fixed number of words as input, we can make use of RNN. With the use of RNN we remove the restriction on the fixed size of the context window, conditioning on context sequences of any length. An exmple of this kind of recurrent Language Models can be seen in figure 2.9

2.6.3 Encoder-Decoder approach

The most common NMT models are a extension of the Neural Language Models with some refinements like the attention mechanism

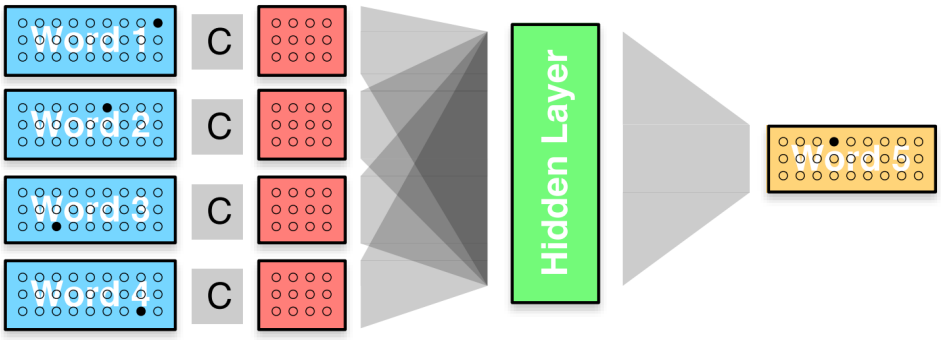

Figure 2.8: Example of a feed-forward neural language model. Image taken from [7]

[18]. We will base out NMT model in the RNN language model. We will trainthemodel with the Encoder-Decoder Approach, see figure 2.10 for an illustration. What we will do is to concatenate the source language sentence and the target language sentence, adding begin and end of sentence tokens. We then train the model with the BPTT algorithm with some additions[7]. When we want to decode (translate) we simply feed the model with the source sentence until the model predicts an end of sequence.

2.6.4 The beam search

Our model predicts words one by one. To choose a wordthemodel computes the probabilities of the words to be the next output word. One we have the probabilities the model chooses the word with a highest probability. The chosen word influences the probabilities of the next words. This process where we choose only the most likely words can lead to some problems like the so-called garden-path-problem, where we follow a sequences of words and realise to late that it is an incorrect one. What we will do to improvethetranslation is to, instead of always choosing one word, doing a pruned beam search, were in each step we will chose up to n words and then keep developing its translation, being able to provide different possible translations (also known as hypothesis) each of them with a translation probability. At the end of the beam search we can choose the sentence with best probability as the best translation. We can observe an visual representation of this beam search at figure 2.11

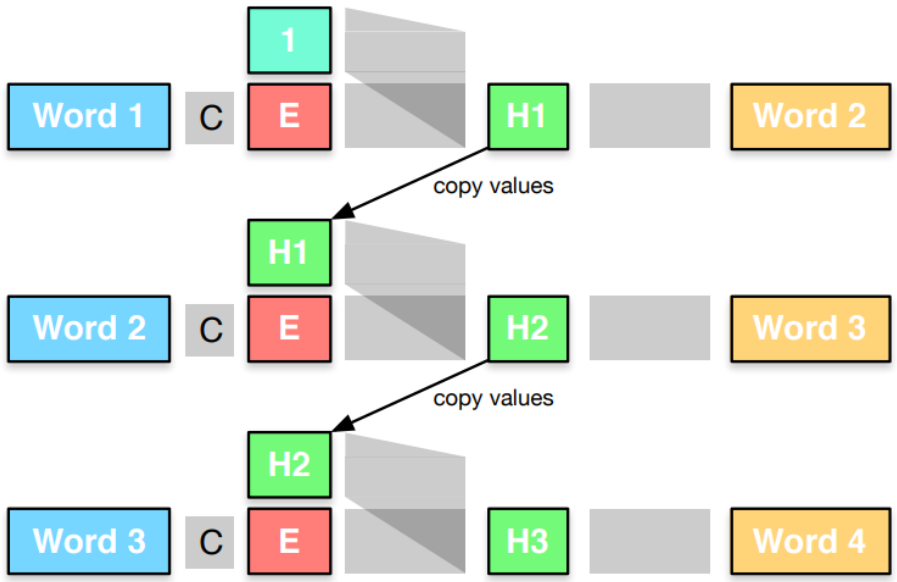

Figure 2.9: Example of a recurrent neural language model.
Image taken from [7]

2.7 Related work

Machine translation has been an active research field since the fifties and, therefore, a lot of literature had been produced over the years.

Back in 1949, Warren Weaver, one of the pioneers in MT, stated the analogy between translating languages and decoding the Enigma Machine[19]. Lots of researchers started to work on MT, and major funding went into the field. A report[20] to asses the progress and the limits on computational linguistics, specially on machine translation was commissioned to ALPAC by the government of the EEUU in 1966. The report had very sceptical conclusions and resulted on a reduction in funding by the EEUU government on MT and an a heavy abandonment of the research in the field, specially in the EEUU. In the 1980s an IBM team lead by Makato Nagao created the first translation model that was based on large numbers of translation examples[21].

The research on Machine translation continued at IBM, contributing with its research to a resurgence in interest in machine translation. One of the publications was [22], were a statistical model of transla-

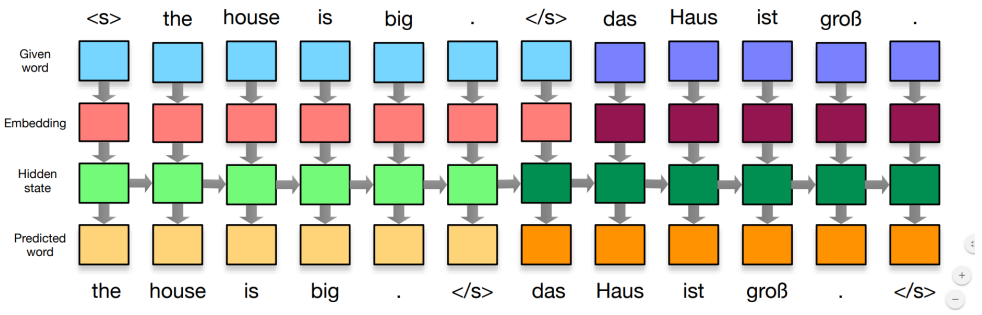

Figure 2.10: Example of a Sequence-to-sequence encoder-decoder model.
Image taken from [7]

tion from French to English was shown. Most of the same authors published 3 years later a paper were the mathematical insights of 5 different models of statistical translation were described[23]. Nowadays Phillip Kohen has gathered most of the knowledge on SMT on his book [6].

Although the adoption of ANN for MT is a recent development, it has been in the sight of researchers for as back as the 1990s. In 1993 a team comprised of M. Asunción Castaño, Francisco Casacuberta and Enrique Vidal proposed models surprisingly similar to the ones used nowadays[24]. At this early stage, the computational needs exceeded the resources and NMT went into a hibernation stage until in 2007, when Holger Schwenk made significant improvements[25]. The problem with the computational needs was still present, and therefore the NMT ideas proposed by Schwenk were slowly adopted. In the meantime, ANN made its way to the then state-of-the-art SMT models by providing additional scores, extending translation tables, reordering and so on [26][27][28][29].

In 2013 and 2014 there were some efforts to abandon statistical approaches completely[30][31][32], but it was not until 2015 when, with the addition of the attention mechanism, NMT finally produced competitive results[18][33]. As with the SMT, Philipp Kohen have produced a very comprehensive paper that gather the main information needed for NMT[7]

Each year a number of competitions are held were MT systems are compared[6]. One of the most important one is *The Conference on Machine translation(WMT)*, which is held since 2016, and its predecessor

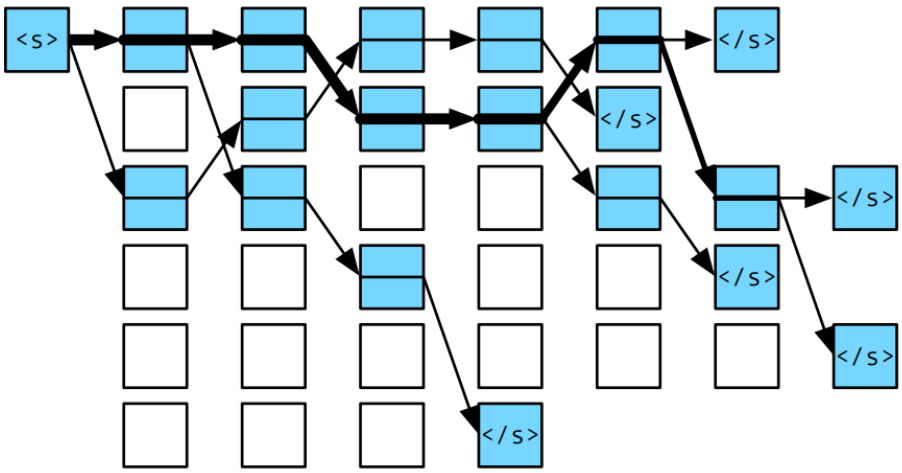

Figure 2.11: Example of a search graph from a beam search.
Image taken from [7]

The Workshop on machine translation (WMT), held from 2005 to 2015. In the conference, as well as in the workshop, the models are evaluated and their results are freely available. We can compare the results on the different models submitted by the different organisation. In 2015 all but one of the submitted models in the competition were statistical models, with the neural one under-performing in comparison[34]. In the 2016 conference, a neural model achieved the best performance in most of the languages[35]. In the 2017 almost all of the submitted models were based on NMT[8]

Chapter 3

Method

In the previous section I have explained the models being compared. In this one I will explain the methodologies used in the experiment pipeline. For the SMT model I have chosen the Moses[36] framework as it provides with a complete set of tools for model creation. For the NMT I have chosen the OpenNMT[37] framework for the same reason I have chosen Moses.

For both models I have chosen a corpora with a news domain. This corpora can be downloaded at the WMT18 webpage [1] and contains parallel text comprised of translated news. The election of this dataset is due to its wide use in different scenarios like the WMT, were it is one of the training sets for the News translated task. I will use the German-English and the Russian-English corpora, the former comprised of 284246 sentences and the latter comprised of 235159 sentences.

3.1 Statistical model

3.1.1 Preprocessing of the data

First of all, we have to pre-processthedata from raw data to truecased and tokenized data, both steps playing a key role inthetraining pipeline. In order to tokenize and truecase the text, I have used the scripts provided by Moses with this end. Notice that an additional script is run in order to delete the sentences that excess certain amount of tokens, as they can cause problems for the training algorithm and make us get

[1] http://data.statmt.org/wmt18/translation-task/training-parallel-nc-v13.tgz

worse results. The upper bound of tokens for the sentences I have chosen is 80, as suggested in the Moses documentation. Intheexperiments I have used the next scripts provided by Moses and available in its webpage [2]

1. tokenizer/tokenizer.perl
2. recaser/truecase.perl
3. training/clean-corpus-n.per

andthedata gets processed as follows:

1. Before preprocessing
    - Guy Verhofstadt: Brexit will be delayed unless Britain makes further concessions to EU.
    - Getting into a huge online fight with JK Rowling about owls, Brexit and postage stamps.
2. After tokenization
    - Guy Verhofstadt : Brexit will be delayed unless Britain makes further concessions to EU .
    - Getting into a huge online fight with JK Rowling about owls , Brexit and postage stamps .
3. After truecase
    - guy Verhofstadt : Brexit will be delayed unless Britain makes further concessions to EU .
    - getting into a huge online fight with JK Rowling about owls , Brexit and postage stamps .

Note that an extra steep will take place were sentences with a exceeding number of tokens will be eliminated from the dataset.

[2] http://www.statmt.org/moses/

3.1.2 Training

Once we havethedata ready we can start trainingthemodel. As stated in the previous section I need to establish a word alignment for parallel data. Moses will do it with GIZA++[38] which is a freely implementation of the IBM models for word alignment. The establishment of the word alignment is the most computing-consuming step in the training. In the training GIZZA++ will be run twice, in bidirectional runs, and the final word alignment will be based on both runs intersection. Once GIZA++ has established the word alignments Moses will infer the Lexical Translation Table and will extract the phrases and save them in a file. Finally the phrase translation probability table will be created with the procedure explained in 2.5.3

3.1.3 Language model

The Language Model (LM) are statistical models learned from monolingual text that compute how probable a sentence is in a given language. The LM are used to ensure a fluent output bythemodel by mixing the probability of a translation given a source sentence with the probability of the translation in the target language. Moses relies on external tools to create the Language model.

3.1.4 Tuning

When we are decoding (translating) with Moses. the framework assigns a score for the translation hypotheses using a linear model. In the tuning section the framework finds the optimal weights for this linear model. For this end I have used a separate test set in order to score the model and keep the optimal weights.

3.1.5 The Decoder

Once I have the model trained, the decoder Moses will use a beam search to generate a set with translation candidates and, if we don't specify the opposite, will chose the most likely of the candidates as the translation sentence.

3.1.6 Environment

For the SMT model creating with Moses I have used a laptop with the following specifications:

- Processor: Intel Core i7-7500U (2 cores, 4 threads, 2.70GH base frequency, 3.5GH base frequency).
- Memory: 8GB DDR4
- Storage: 256GB SSD

3.2 Neural Model

For the NMT models I will use the tools given by the OpenNMT which can be used to pre-process the data, train the network, translate and evaluate. OpenNMT is a generic deep learning framework that can be found in it webpage [3] and is mainly specialised in sequence to sequence models. For this project I have used the LUA implementation of OpenNMT.

3.2.1 Preprocessing

As with the SMT wee need to preprocessthedata in order to prepare it for the training. IntheNMT this process includes several steps like tokenization, in order to generate an adequate dataset to use in the training step. One of the output of the pre-processing with OpenNMT is the obtaining of a dictionary file for each language, which assigns an internal number for each different token. One of the main hyper parameter that I decided to tune is the vocabulary size, which is the number of tokens that will get an index assigned in the dictionary file. Internally the system never uses the words themselves, but uses these indices and therefore the system will not be able to recognise a token if this do not have an index assigned. While both vocabulary sizes are important, the target vocabulary size can be decisive in the quality ofthetranslation. Ifthetarget language real vocabulary is big but we restrict it too much in the pre-processing time, the model will produce a lot of "unknown" words in the translation. When we restrict the vocabulary size of any language the preprocessing keeps the more

[3] http://opennmt.net/

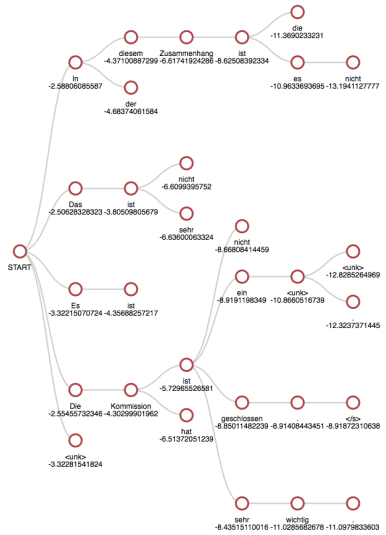

Figure 3.1: Beam search visualisation.
Source: http://opennmt.net/OpenNMT/translation/beam_search/

frequent words. The vocabulary size is an hyperparameter to tune for the network and therefore there is not any hard rule to choose the size. However we can use the Zipf rule [10] to help us chose the vocabulary size.

3.2.2 Training

The training of the model is handled automatically by the OpenNMT[37] framework. In this project the default options were used, were I used a simple LSTM RNN as a encoder and a LSTM RNN with an attention layer as a decoder. The documentation of the training script used can be found at its documentation page[4], while its implementation can be found at its github page [5].

3.2.3 Translation

The translation is done with a beam search as explained in 2.6.4. The beam search with OpenNMT provides some options like the ability to tune the beam size which can be used to trade-off translation time and search accuracy. We can also use the beam tree to chose the n-best

[4]http://opennmt.net/OpenNMT/options/train/
[5]https://github.com/OpenNMT/OpenNMT/blob/master/train.lua

translations for a query. We can see a visual representation of a search in the figure **??**

3.2.4 Hyperparameter selection

The NMT model is based on a LSTM network and therefore I need to define its architecture. There is not hard directives when it comes to architecture selection, and usually a mix of knowledge and trial and error is needed. First I built a baseline system with two stacked LSTM networks with 500 hidden units each and 50.000 vocabulary size. I have chosen this architecture as a baseline model based on the OpenNMT documentation. I then have trained two more models, one with a larger architecture (2 stacked LSMT networks with 1024 hidden units), and one with an unlimited vocabulary size.

I have chosen this 3 architectures to see how the number of hidden nodes and the vocabulary size impact the performance. Due to time limitations I have compared the performance of the 3 architectures with models trained with 50.000 sentences (48.784 after preprocessing), as I considered the they can give insightful information for the bigger models. This is not an in-depth hyperparameter tuning and the aim was to help me chose the architecture.

The results of this comparison can be seen in section 4, where can be seen that the baseline architecture outperformed the other two and, therefore, was the one chosen for the NMT models of this project.

3.2.5 Environment

I have trained the NMT models under the *Google Cloud Computing* services with the following specifications:

- Processor: 2x vCPU
- Memory: 16GB
- Graphics card: 1x NVIDIA Tesla K80
- Storage: 500GB HDD

3.3 Evaluation

Evaluation is a difficult task in MT and therefore different methods have been developed, either human-based or software-based[39]. For this project the evaluation has been done with the BLEU metric score[9]. BLEU is an automatic method for MT evaluation that highly correlates with human evaluation. BLEU is quick, cheap and language-independent, which makes it a good choice for this project. BLEU base its evaluation in a modification of the precision measure[40]. To evaluate a translation we need a reference text to compare the translation with. The reference can include several different translations for a source sentence, but I will focus on references with only one possible translation. The conventional precision calculates how many words of the MT translated sentence (From now on candidate) appear in the reference sentence. The repetition of common words of phrases is a common problem that can lead to over-inflated precision scores as can be seen in the next translation:

- **Candidate:** The the the the the the the
- **Reference:** The water bottle was in the table

Out candidate would obtain a 7/7 precision score but obviously it is not a good translation. BLUE introduces a modification of the precision score to deal with this over-inflation. With the BLEU score we first count the maximum number of times each word in the MT translation occurs in the reference. Afterwards we clip the total count of each candidate word by the reference count, we add these clipped counts up and we divide by the length of the candidate. Following the last example we would obtain a precision of 2/7 as the word "the" is clipped to 2, the maximum times the word "the" appears in the reference.

We compute the modified precision with uni-grams, which means with words by their own, with bi-grams, which means pairs of consecutive words and so on until four-grams. The modified precision for a n-gram decays exponentially with the number of grams, as it is much more difficult to find sequences of 4 consecutive words of the candidate in the references than of 3 consecutive words, and so on. Therefore we will weight the modified precision obtained by each n-gram, giving more importance to the longer n-grams. BLEU also uses some techniques to penalise candidates too short, as this kind of sentences

can obtain better precision scores although being worst translations, as its modified precision is divided by a lower number.

The BLEU metric score is implemented in both frameworks in use in this project. The BLEU implementation that the Moses framework uses can be found in its github repository[6]. The BLEU implementation used on OpenNMT follows the sintaxy[7] used in the Moses implementation and can be found in its github repository[8].

[6]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

[7]http://opennmt.net/OpenNMT/tools/scorer/

[8]https://github.com/OpenNMT/OpenNMT/blob/master/benchmark/3rdParty/multi-bleu.perl

Chapter 4

Results

In order to facilitate the comparison, all the results in this section are presented as plots. The source data of all the plots is available in form of tables in Appendix A.

4.1 SMT

4.1.1 Quality scores

First of all I addressed the quality of the SMT model. This model is build based on the phrase translation probabilities inferred over a large parallel corpora. I built the model with the Moses framework, evaluating the outcome with the BLEU metric score. I gradually increase the number of sentences in the training set to address how the size of the training set affects the performance of the model and in its computational needs. I have created multiples models for translation from German to English and from Russian to English with different sizes of the training dataset.How the models performed can be seen on the figure 4.1.

In both models I found that the learning curve of the model was similar to a exponential rise to a limit. At the beginning, the German to English model learns extremely quick, going from 0 BLEU score to more than 13 with just an approximately 10.000 sentence training. The learning of this model quickly loses momentum and slows down. We can observe that, with approximately the first 10.000 sentences, the model improves by more than 13 BLEU points but, from 219391 to 283453 sentences the model dos not even improve half point. The

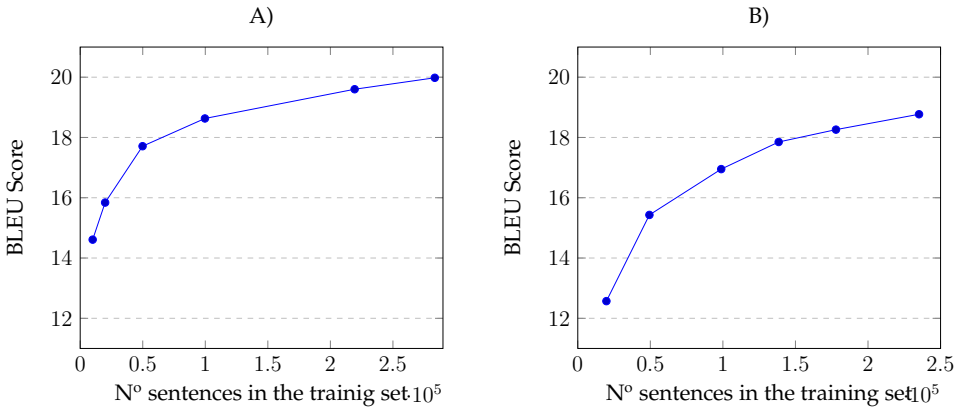

Figure 4.1: BLEU score of multiple SMT models for different languages and training dataset sizes (Number of sentences): A) German to English models, B) Russian to English models

results of the Russian to English model are similar to the German to English model. One possible explanation for this learning curve is that there are phrases pairs that are very common, as they are used to give cohesion, explain or create grammatical structures. This very common phrase pair will occur multiple times even in smalls corpora and, consequently, the model will be able to learn them even from a small dataset. To learn rare phrase pairs the model requires of a larger dataset to be able to find examples of those phrase pairs in them. This explain why the learning slows down but does not completely stops. Once have added enough sentences to the dataset for the model to have learn most of the common phrase-pairs, it will keep learning as I increase the dataset because the model will probably find more, rarer phrase pairs.

4.1.2 CPU time required by the SMT model

As explained in section 3.1.6, I have trained the SMT model with a Intel Core i7-7500U. The training conditions have been identical for every model trained and the data have been collected by the log files generated by Moses.

As we can observe, the CPU-time required for the models to train is linear with the size of the training dataset. The training time increases at an approximate $3.50E-07$ minutes per sentence. One interesting observation is that both models scale equally independent of the lan-

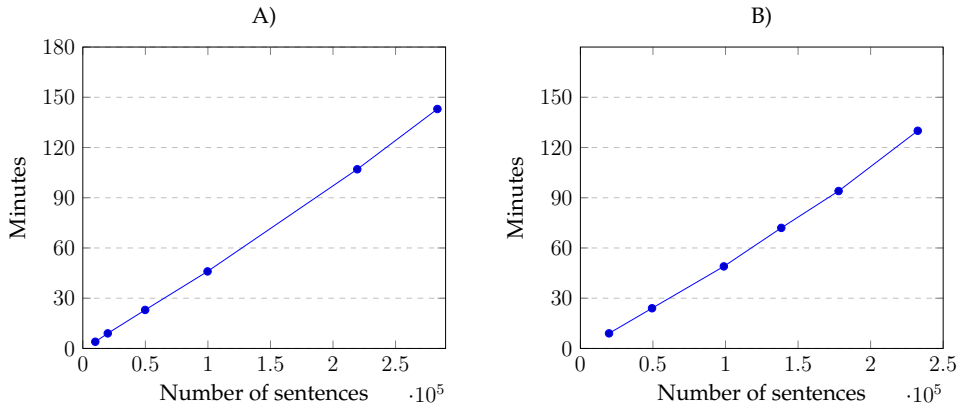

Figure 4.2: Training time of SMT models for different languages and training dataset sizes (Number of sentences): A) German to English models, B) Rusian to English models.

guage.

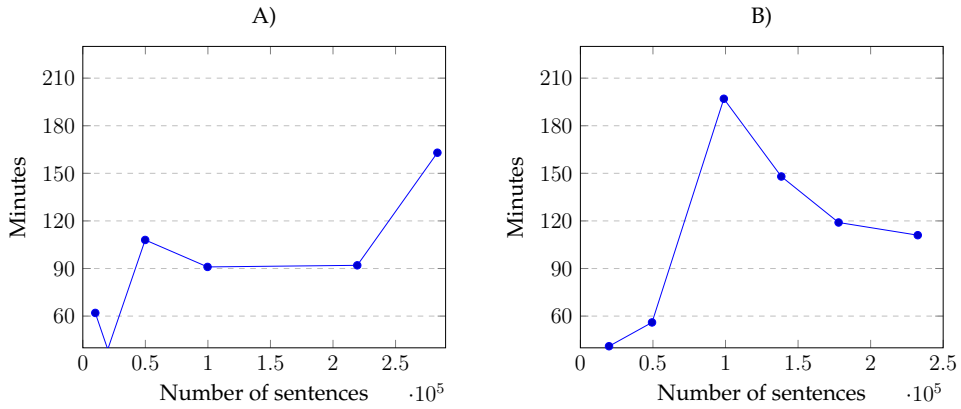

Figure 4.3: Tuning time of the SMT models with different languages pairs and training dataset sizes(Number of sentences): A) German to English models, B) Rusian to English models.

As explained in section3.1.4, in the SMT models one extra steep is needed to improve the translation quality, known as tuning. As we can see in figure 4.3, the tuning time seems more erratic than the training time.

The total time needed for the creation of the SMT depending on the number of sentences used to create them can be seen in figure 4.4. As we can see, although the tuning time makes it irregular, it tend to increase with the number of sentences in the training set.

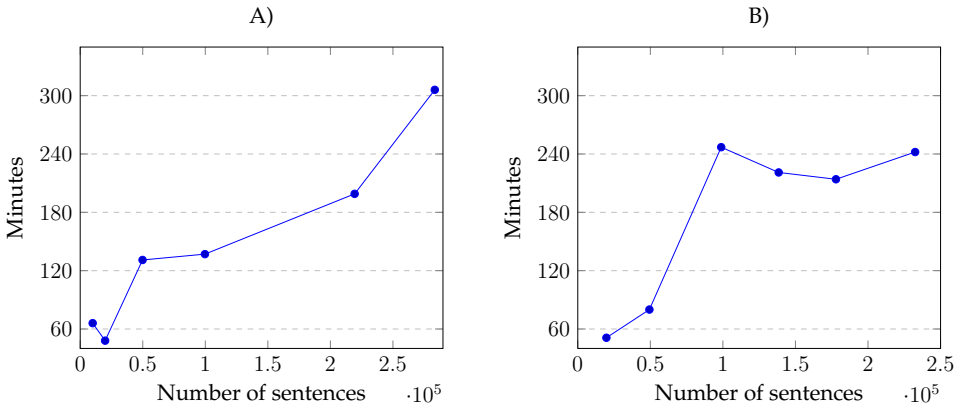

Figure 4.4: Total time of the SMT models with different languages pairs and training dataset sizes(Number of sentences): A) German to English models, B) Rusian to English models.

4.2 NMT

4.2.1 BLEU score by number of epochs

To create the NMT models I had based the translation on large ANN. The models are implemented with the OpenNMT framework, which provides with the set of tools needed to build them. As with the SMT models, all the models are evaluated using the BLEU metric score.

When NMT models are trained, an iterative process takes place where the ANN is trained. This iterative process can be done indefinitely, and istheresponsibility to choose a stop condition. An epoch is a complete loop run where we train the ANN with all the training set. In order to estimate how many epochs should I trainthemodels with, I have computed the BLUE score over each epoch for both languages trained with approximately 50.000 parallel sentences and using the baseline architecture (2 layers, 500 hidden nodes, 50.000 vocabulary size). As we can see in figure 4.5, the learning happens mostly in the first 20 epoch.

I have repeated the anterior experiment with the maximum available data for each language: 284247 paralel sentences (283453 after preprocessing) for the German to English model and 235160 (232480 after preprocessing) for the Russian to English one. As we can see in figure 4.6, the learning here is quicker at the firsts epochs. In this models the learning happens mostly in the first 15 epoch, a common number of

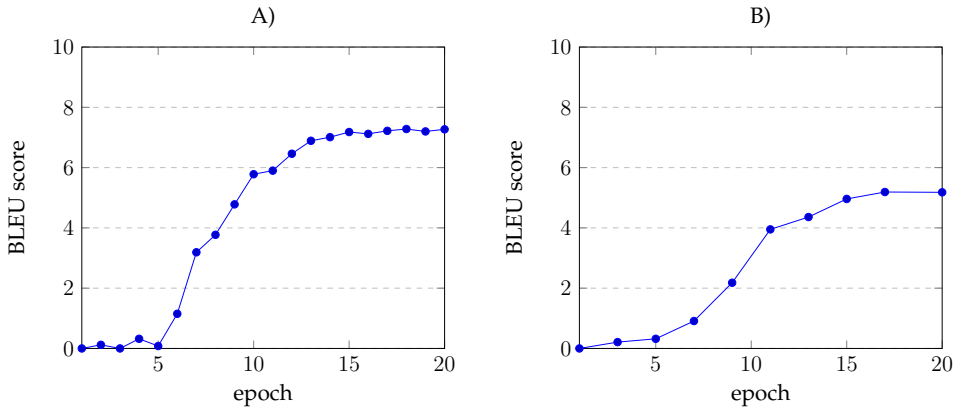

Figure 4.5: BLEU Score over the number of epoch for two baseline models (2 layers, 500 hidden nodes, 50.000 vocabulary size): A) German to English model trained with 48784 parallel sentences, B) Russian to English model trained with 46509 parallel sentences.

needed epoch in NMT[7].

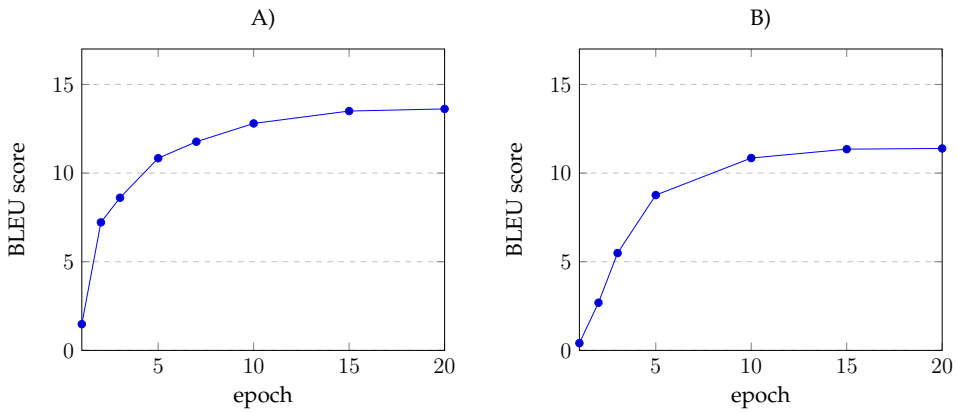

Figure 4.6: BLEU Score over the number of epoch for two baseline models (2 layers, 500 hidden nodes, 50.000 vocabulary size). A) German to English model trained with 277.749 parallel sentences, B) Russian to English model trained with 219.954 parallel sentences.

I have used this information as a empirical stop criteria in the training of the models.

4.2.2 Hyperparameter selection

As explained in section 3.2.4, I need to define the architecture ofthenetwork as well as to tune in some parameters. I have decided to train a baseline model and then compare its performance with two different models, one with a bigger ANN and one with a bigger vocabulary size. The baseline model architecture is that of a two layers LSTM network, with 500 hidden units and a 50.000 vocabulary size. All the models of this subsection are models of translation for German to English, and they are trained over 48.784 sentences of parallel data. As well as the performance, I also wanted to know if a different architecture or vocabulary size had effects on the learning curve. In the figure 4.7 we can observe the performance and learning rate ofthebaseline system.

I have compared the baseline model with a model with a bigger architecture, a two layered LSTM network with 1024 hidden units. As we can see in figure 4.8, this model performed worse than the baseline model, meaning that a smaller network is able to perform better on unseen data that the larger ANN, when trained on a small dataset (approximately 50.000 sentences). The baseline model scored 7.35 BLEU points at its peak, while the model with a larger ANN scored 5.06 at its peak. Both models have a similar learning curve, heavily decreasing its learning at around the fifteenth epoch.

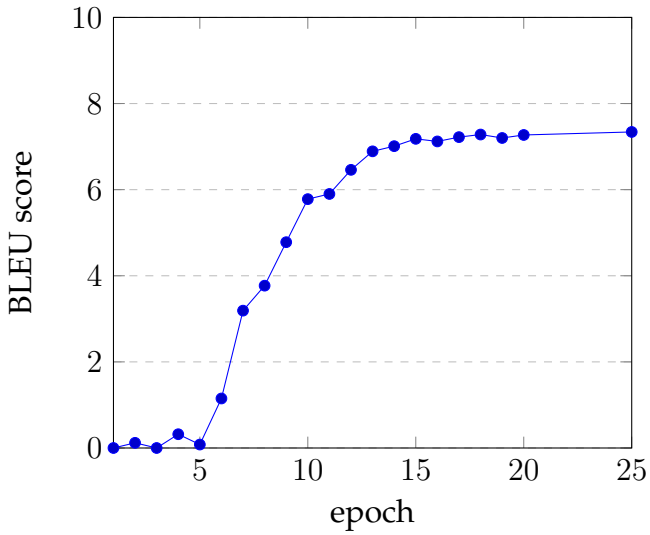

Figure 4.7: Performance of the baseline model over the number of epoch.

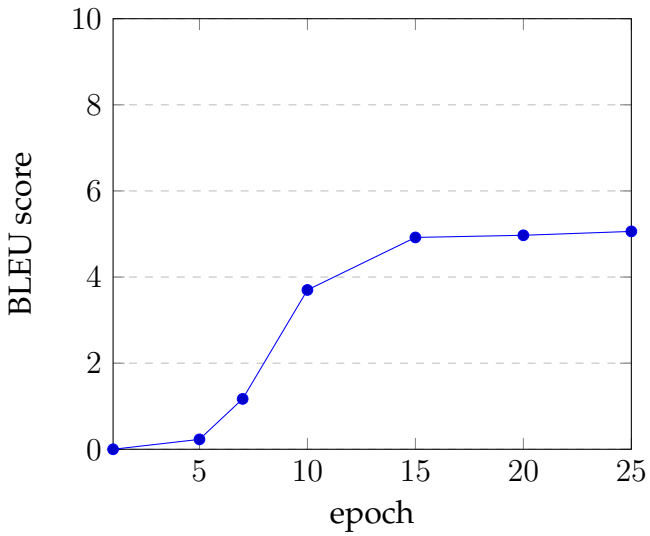

Figure 4.8: Performance of the modified model with a bigger number of hidden units (1024) over the number of epoch.

I have also compared the baseline model with a model with unlimited vocabulary size. The performance of this model can be seen at figure 4.9. Restricting the vocabulary size yielded better results than indexing every single word. The learning curve of both models is similar, heavily decreasing the learning after approximately the fifteenth epoch.

4.2.3 BLEU score over increasing training set

Once I had chosen the hyperparameters fortheNMT models, a two-layered LSMT network with 500 hidden units and a 50.000 vocabulary size, I measured its performance depending on the training dataset size. I have trained models for both language pairs with increasingly larger training sets, and have measured its performance at the twentieth epoch. The results can be seen at figure 4.10, where we can observe a similar learning curve for both models. We can also observe than the Russian to English model underperformed in comparison with the German to English model.

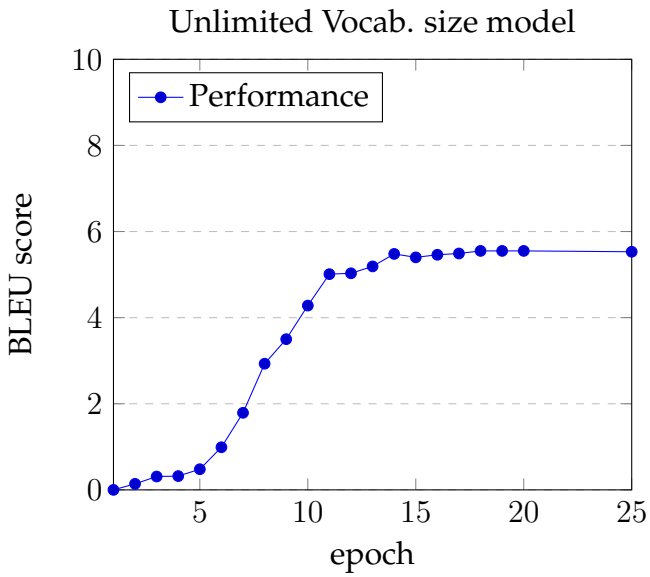

Figure 4.9: Performance of the modified model with an unlimited vocabulary size over the number of epoch.

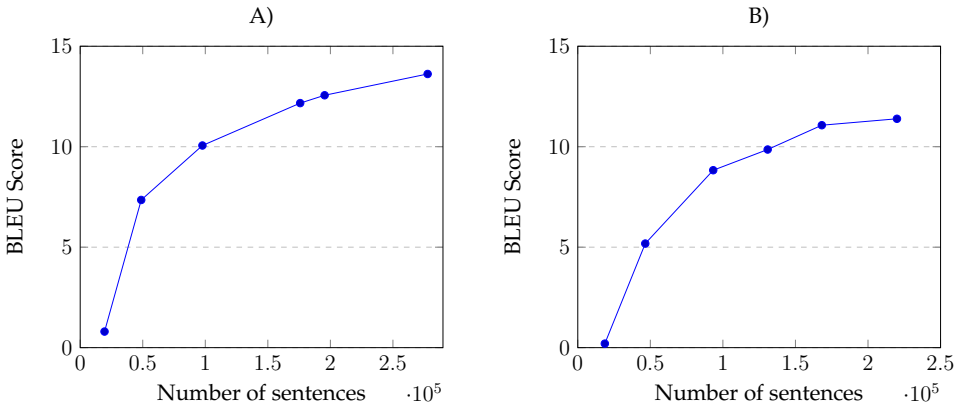

Figure 4.10: BLEU scores of NMT models with different dataset size (number of sentences) and languages pairs: A) German to English models, B) Russian to English models.

4.2.4 GPU times with increasing training set

I have timed the training ofthedifferent models to asses its scaling with the training data size. We can observe the results in the figure 4.11, where we clearly see that the training increases linearly with the size of the training set. One interesting observation is that the training of

the Russian to English models is slower than the German to English models.

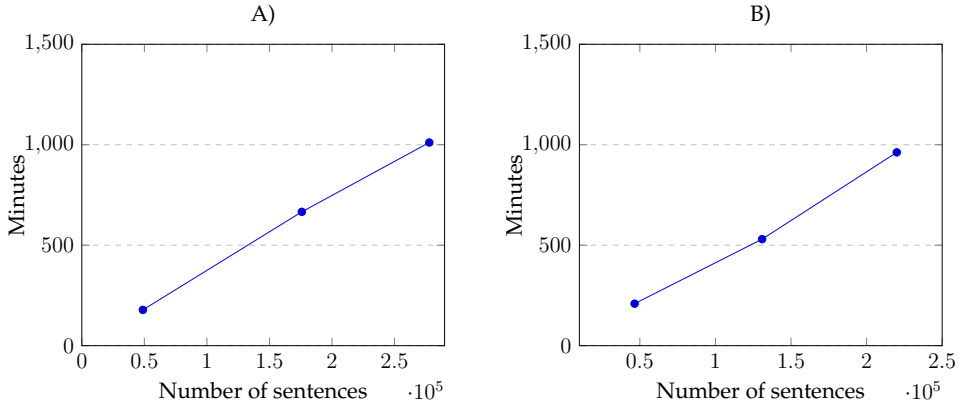

Figure 4.11: GPU training time comparison for NMT models with different training set size (number of sentences) and languages pairs: A) German to English models, B) Russian to English models

4.3 Comparison of the performance

In the figure 4.12 we can seethefinal comparative results on performance. The SMT models outperformed the NMT ones for all the sizes of the training dataset, scoring approximately 5 BLEU points more in the German to English trained with all the data available for the language pair, and around 7 points more in the Russian to English trained with all the available data for the language pair.

4.3.1 Comparison of the training time.

As the training time for the NMT models far exceeded the time limitations for the project, I have made a estimation given the seconds per epoch in the same environment that the SMT models were trained on with all the avaliable data for both laguages pairs. Once I have colected the time per epoch I have multiplies it by 20 to obtain an estimate of the CPU time to train the models. We can estimate with great confidence the time for other datasets sizes as the training time is linear with it.

As we can see in the figure 4.13, the time required for the NMT is orders of magnitude greater than the time needed by the SMT. The

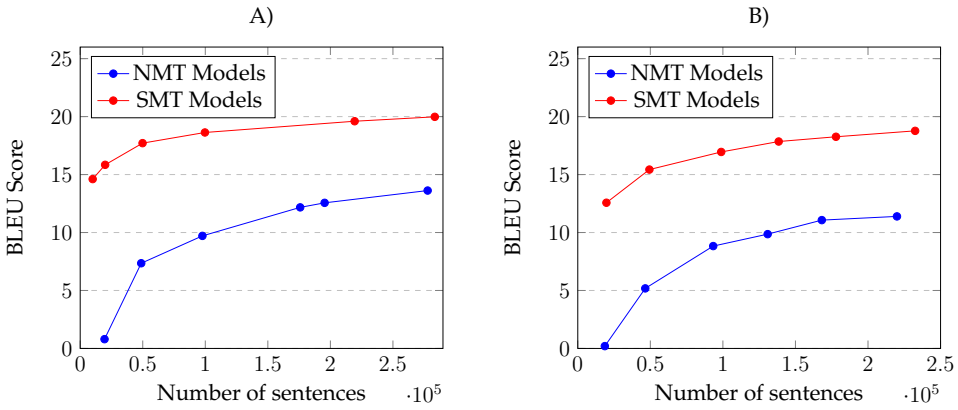

Figure 4.12: Comparison between SMT and NMT models with different training dataset sizes (number of sentences) and languages pairs: A) German to English models, B) Russian to English models.

NMT approach requires far more compatitonal resources than the SMT approach.

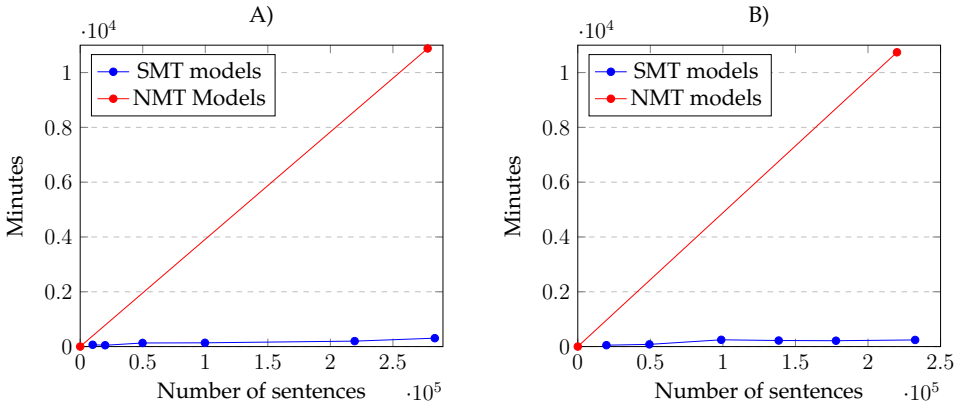

Figure 4.13: Comparison between CPU-time required by SMT and NMT models for different training dataset sizes (number of sentences) and languages pairs: A) German to English models, B) Russian to English model.

Chapter 5

Discussion

In this project I have compared two different paradigms of MT, one based on statistical inference of phrase pairs translation probabilities and other based on the use of ANN. I have used relatively small datasets for the training of models. I have created models for two languages pairs, one being German to English and the other Russian to English. The SMT outperformed the NMT with all the sizes of training for both language pairs. The most difference in performance has been found with extremely small data sets like the ones with approximately 20.000 sentences, where the SMT performed around 14 BLEU points better than the NMT for the German to English model and more than 13 for the Russian to English. This results follow the lines by Kohen[7], that NMT need of some more extra steps that the ones implemented in this report to improve the results. With a provided dataset of 284247 parallel sentences from German to English (283453 after preprocessing for the SMT model and 277749 after preprocessing for the NMT model), the SMT model achieved a 139% performance in comparison with the NMT model. With a provided 235160 parallel sentences from Russian to English (232480 after preprocessing for the SMT model and 219954 after preprocessing for the NMT model), the SMT model achieved a 167% performance in comparison with the NMT model. In both paradigms the models from German to English outperformed the models from Russian to English.

It is interesting to note that this results are not in line with the actual state-of-the-art, where NMT is outperforming SMT models[8]. The explanation of this is that this stat-of-the-art NMT models use some extra techniques like BPE and backtranslation and that they are trained on

extremely large datasets in the order of tens of millions of parallel sentences[8].

One implication of the results is that, in case that we only have a relatively small training set, SMT models can be a better option than NMT. This is specially important for languages with a small number of speakers or endangered languages, were the gathering of parallel data can be a difficult task.

The NMT proved to be much more expensive to train in terms of computational resources. While CPU-trained SMT times are within a sensible range, the times for CPU-trained NMT models make them unpractical to train on CPUs, making necessary the use of GPUs to get the time down to a sensible amount. The need of specific hardware is specially important if we intend to research and need to create multiple models. The NMT models scaling also proved more sensitive than the SMT, with the NMT models from Russian to English being noticeably slower to train than the ones From German to English. This is due to the lower amount of source tokens per second processed in the Russian to English model, with the German to English model processing up to 133% the numbers of source tokens per second that the Russian to English models do.

One implication of this results is that, in case of not having acces to high-end hardware suitable for ANN training, the SMT can be a better option, as their requeriments are lower and can be asumend even for low/mid-end hardware.

5.1 Limitations

This project was mostly limited by the relatively small training sets for the models. This is due to the high amount of resources needed to train the models with larger datasets. This project is also limited to a specific domain for both, training and testing sets, which are from a news domain and therefore I can not conclude that the results would be the same for a general propose domain. The project has also been limited to a unique technique for evaluation, the BLEU metric score.

Other limitation I have found in this project is that I have restricted the models to a set of techniques due to time limitations, but that both paradigms can be heavily tuned and multiple different techniques could be implemented with potentially different results.

This project is lacking of statistical rigour, as I only disposed of a single testing set, and therefore the results are inconclusive. I did some basic bootstrapping techniques to calculate the confidence intervals at 95%, which were around 0.1 BLEU points for all the models. However, more rigorous and proper statistical techniques with more testing sets should be used in order to make the project conclusive.

5.2 Ethics and sustainability

MT can raise some ethical dilemmas as when it is used as a tool for people monitoring. Some governments are counted within the largest funders of research in MT[6], given that MT facilitates the automatically monitorization of foreign people that is not speaking or texting in the government language. This can be a ethical dilemma as the MT facilitates that monitoring, which can be understood as a intrusion in the privacy.

The use of MT can be beneficial for the planet as it makes easier the communication between people. One example would be the reduction of paper used for information papers when, instead of printing them in different languages, they can be printed in just one of them and then the user can use MT to translate it to his language.

5.3 Future work

As stated in previous sections, this project was heavily restricted due to time and resources limitations, specially in the set of techniques used to build models. I would like to asses how the inclusion of BPE and backtranslation affected the performance of the NMT models. It would be interesting to try to get close to the scores the models are getting in the MT competitions like [8], for which the models whould be trained with bigger datasets.

Other interesting research would be how the election of the news domain have influenced the results. To try to address this questions, I would try to find a general propose domain dataset of approximately the same number of sentences thanthenews domain one, and then evaluate the models trained with the general domain with the new domain test set. I would also train models with the news-domain dataset and then I would test them with the news domain test set.

One interesting finding was that a smaller number of hidden units yielded better results than a larger one. It would be interesting to watch how this architecture holds up against the one with a large number of hidden units if I increase the training set. Also I would be interested in comparing the architecture with architectures with a different number of layers.

Chapter 6

Conclusion

The result of this study showed that this project's SMT models performed better than the NMT when implemented with a basic set of techniques and a relatively small training set. In both paradigms, the German to English models performed better than the Russian to English ones. I also showed that the computational resources for the creation of NMT models far exceeds the resources needed by SMT. All this observations are just trends and they are not conclusive as this project is lacking of statistical testing due to lacking of data for proper statistical techniques.

Appendices

Appendix A

Results in table form

German to English SMT models		
nº sentences before processing	nº sentences post processing	BLEU score
284247	283453	19.98
220000	219391	19.60
100000	219391	18.63
50000	49852	17.71
20000	19938	15.84
10000	9972	14.61

Figure A.1: BLEU Scores of the SMT German to English models with the provided number of sentences and the sentences post preprocessing.

Russian to English SMT models		
nº sentences before processing	nº sentences post processing	BLEU score
235160	232480	18.77
180000	177923	18.26
140000	138446	17.85
100000	98846	16.95
50000	49364	15.43
20000	19729	12.57

Figure A.2: BLEU Scores of the SMT Russian to English models with the provided number of sentences and the sentences post preprocessing.

German to English SMT models		
nº sentences before processing	nº sentences post processing	Training time
284247	283453	143
220000	219391	107
100000	99713	46
50000	49852	23
20000	19938	9
10000	9972	4

Figure A.3: Training time of the SMT German to English models with the provided number of sentences and the sentences post preprocessing.

German to English SMT models		
nº sentences before processing	nº sentences post processing	Tuning time
284247	283453	163
220000	219391	92
100000	99713	91
50000	49852	108
20000	19938	39
10000	9972	62

Figure A.4: Tuning time of the SMT German to English models with the provided number of sentences and the sentences post preprocessing.

German to English SMT models		
nº sentences before processing	nº sentences post processing	Total time
284247	283453	306
220000	219391	199
100000	99713	137
50000	49852	131
20000	19938	48
10000	9972	66

Figure A.5: Total time of the SMT German to English models with the provided number of sentences and the sentences post preprocessing.

Russian to English SMT models		
nº sentences before processing	nº sentences post processing	Training time
235160	232480	130
180000	177923	94
140000	138446	72
100000	98846	49
50000	49364	24
20000	19729	9

Figure A.6: Training time of the SMT Russian to English models with the provided number of sentences and the sentences post preprocessing.

Russian to English SMT models		
nº sentences before processing	nº sentences post processing	Tuning time
235160	232480	111
180000	177923	119
140000	138446	148
100000	98846	197
50000	49364	56
20000	19729	41

Figure A.7: Tuning time of the SMT Russian to English models with the provided number of sentences and the sentences post preprocessing.

Russian to English SMT models		
nº sentences before processing	nº sentences post processing	Total time
235160	232480	242
180000	177923	214
140000	138446	221
100000	98846	247
50000	49364	80
2000	19729	51

Figure A.8: Total time of the SMT Russian to English models with the provided number of sentences and the sentences post preprocessing.

	BLEU Score		
Epoch	Baseline	1024 Hidden nodes	Unlimited Vocab
1	0	0	0
2	0.12		0.14
3	0		0.31
4	0.32		0.32
5	0.08	0.23	0.48
6	1.15		0.99
7	3.19	1.17	1.79
8	3.77		2.93
9	4.78		3.5
10	5.78	3.7	4.28
11	5.9		5.01
12	6.46		5.03
13	6.89		5.19
14	7.01		5.48
15	7.18	4.92	5.4
16	7.12		5.46
17	7.22		5,49
18	7.28		5.55
19	7.2		5.47
20	7.27	4.97	5.55

Figure A.9: Performance ofthehyperparameter selection models with different number of epoch of traning

German to English NMT models		
nº sentences before processing	nº sentences post processing	BLEU score
284247	277749	13.62
200000	195392	12.5
180000	175859	12.17
100000	97702	10.06
50000	48784	7.27
20000	19452	0.8

Figure A.10: BLEU score of NMT models from German to English with different training dataset sizes.

German to English NMT models		
nº sentences before processing	nº sentences post processing	GPU time
284247	277749	1011
180000	175859	666
50000	48784	178

Figure A.11: GPU time required for the full training of NMT models from German to English with different training dataset sizes.

Russian to English NMT models		
nº sentences before processing	nº sentences post processing	BLEU scores
235160	219954	11.39
180000	168174	11.07
140000	130855	9.86
100000	93336	8.83
50000	46509	5.18
20000	18645	0.2

Figure A.12: BLEU score of NMT models from Russian to English with different training dataset sizes.

German to English NMT models		
nº sentences before processing	nº sentences post processing	Total time
235160	219954	962
180000	168174	750
140000	130855	530
50000	46509	209

Figure A.13: GPU time required for the full training of NMT models from Russian to English with different training dataset sizes.

Model	German to English	Russian to English
Pre-processing Sentences	284247	235160
Post-processing sentences	277749	219954
CPU - Epoch time	543.97	536,98
Expected training time	10879,44	10739,6

Figure A.14: Expected CPU time required for the full training of NMT models with both language pairs with all the available corpora.

Bibliography

[1] A. Reynolds Jr, "The conference on mechanical translation.", *Mechanical Translation*, vol. 1, no. 3, pp. 47–55, 1954.

[2] M. Dureja and S. Gautam, "Article: Speech-to-speech translation: A review", *International Journal of Computer Applications*, vol. 129, no. 13, pp. 28–30, Nov. 2015, Published by Foundation of Computer Science (FCS), NY, USA.

[3] A. W Black, R. D. Brown, R. Frederking, R. Singh, J. Moody, and E. Steinbrecher, "Tongues: Rapid development of a speech-to-speech translation system", Apr. 2002.

[4] U.-V. Albrecht, M. Behrends, R. Schmeer, H. K. Matthies, and U. von Jan, "Usage of multilingual mobile translation applications in clinical settings", *JMIR mHealth and uHealth*, vol. 1, no. 1, 2013.

[5] M. Carl, S. Gutermuth, and S. Hansen-Schirra, "Post-editing machine translation", *Psycholinguistic and cognitive inquiries into translation and interpreting*, vol. 115, p. 145, 2015.

[6] P. Koehn, *Continuous Univariate Distributions Vol. 2*. Cambridge University Press, 2009.

[7] ——, "Neural machine translation", *CoRR*, vol. abs/1709.07809, 2017. arXiv: `1709.07809`. [Online]. Available: `http://arxiv.org/abs/1709.07809`.

[8] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, S. Huang, M. Huck, P. Koehn, Q. Liu, V. Logacheva, C. Monz, M. Negri, M. Post, R. Rubino, L. Specia, and M. Turchi, "Findings of the 2017 conference on machine translation (wmt17)", in *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 169–214. [Online].

Available: http://www.aclweb.org/anthology/W17-4717.

[9] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation", in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://doi.org/10.3115/1073083.1073135.

[10] G. K. Zipf, *Human behavior and the principle of least effort: An introduction to human ecology*. Houghton-Mifflin., 1949.

[11] B. Thouin, "The meteo system", *Practical experience of machine translation*, pp. 39–44, 1982.

[12] R. M. S. d. Oliveira, R. C. F. AraÃ, F.-c. J. B. Barros, A. P. Segundo, R. F. Zampolo, W. Fonseca, V. Dmitriev, and F. S. Brasil, "A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges", en, *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, vol. 16, pp. 628–645, Sep. 2017, ISSN: 2179-1074. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2179-10742017000300628&nrm=iso.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[14] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description", *CoRR*, vol. abs/1411.4389, 2014. arXiv: 1411.4389. [Online]. Available: http://arxiv.org/abs/1411.4389.

[15] T. K. Moon, "The expectation-maximization algorithm", *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, Nov. 1996, ISSN: 1053-5888. DOI: 10.1109/79.543975.

[16] M. Freitag and Y. Al-Onaizan, "Beam search strategies for neural machine translation", *arXiv preprint arXiv:1702.01806*, 2017.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space", *CoRR*, vol. abs/1301.3781, 2013. arXiv: 1301.3781. [Online]. Available: http://arxiv.org/abs/1301.3781.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *CoRR*, vol. abs/1409.0473, 2014. arXiv: 1409.0473. [Online]. Available: http://arxiv.org/abs/1409.0473.

[19] W. Weaver, "Translation", *Machine translation of languages*, vol. 14, pp. 15–23, 1955.

[20] J. R. Pierce and J. B. Carroll, *Language and Machines: Computers in Translation and Linguistics*. Washington, DC, USA: National Academy of Sciences/National Research Council, 1966.

[21] M. Nagao, "A framework of a mechanical translation between japanese and english by analogy principle", *Artificial and human intelligence*, pp. 351–354, 1984.

[22] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation", *Comput. Linguist.*, vol. 16, no. 2, pp. 79–85, Jun. 1990, ISSN: 0891-2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=92858.92860.

[23] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation", *Comput. Linguist.*, vol. 19, no. 2, pp. 263–311, Jun. 1993, ISSN: 0891-2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=972470.972474.

[24] M. A. Castaño, F. Casacuberta, and E. Vidal, "Machine translation using neural networks and finite-state models", *Theoretical and Methodological Issues in Machine Translation (TMI)*, pp. 160–167, 1997.

[25] H. Schwenk, "Continuous space language models", *Computer Speech & Language*, vol. 21, no. 3, pp. 492–518, 2007.

[26] ——, "Continuous space translation models for phrase-based statistical machine translation", *Proceedings of COLING 2012: Posters*, pp. 1071–1080, 2012.

[27] S. Lu, Z. Chen, and B. Xu, "Learning new semi-supervised deep auto-encoder features for statistical machine translation", in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 122–132.

[28] S. Kanouchi, K. Sudoh, and M. Komachi, "Neural reordering model considering phrase translation and word alignment for phrase-based translation", in *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, 2016, pp. 94–103.

[29] P. Li, Y. Liu, M. Sun, T. Izuha, and D. Zhang, "A neural reordering model for phrase-based translation", in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 1897–1907.

[30] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models", in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1700–1709.

[31] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[32] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", *CoRR*, vol. abs/1409.1259, 2014. arXiv: 1409.1259. [Online]. Available: http://arxiv.org/abs/1409.1259.

[33] S. Jean, O. Firat, K. Cho, R. Memisevic, and Y. Bengio, "Montreal neural machine translation systems for wmt'15", in *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015, pp. 134–140.

[34] O. Bojar, R. Chatterjee, C. Federmann, B. Haddow, M. Huck, C. Hokamp, P. Koehn, V. Logacheva, C. Monz, M. Negri, M. Post, C. Scarton, L. Specia, and M. Turchi, "Findings of the 2015 workshop on statistical machine translation", in *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1–46. [Online]. Available: http://aclweb.org/anthology/W15-3001.

[35] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Neveol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri, "Findings of the 2016 conference on machine translation", in *Proceedings of the First Conference on Machine Translation*, Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 131–198. [Online]. Available: http://www.aclweb.org/anthology/W/W16/W16-2301.

[36] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation", in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ser. ACL '07, Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 177–180. [Online]. Available: http://dl.acm.org/citation.cfm?id=1557769.1557821.

[37] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation", *CoRR*, vol. abs/1701.02810, 2017. arXiv: 1701.02810. [Online]. Available: http://arxiv.org/abs/1701.02810.

[38] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models", *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[39] A. L. Han and D. F. Wong, "Machine translation evaluation: A survey", *CoRR*, vol. abs/1605.04515, 2016. arXiv: 1605.04515. [Online]. Available: http://arxiv.org/abs/1605.04515.

[40] I. D. Melamed, R. Green, and J. P. Turian, "Precision and recall of machine translation", in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*, ser. NAACL-Short '03, Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 61–63. DOI: 10.3115/1073483.1073504. [Online]. Available: https://doi.org/10.3115/1073483.1073504.