PART-OF-SPEECH TAGS AS INPUT FEATURES IN GERMAN-TO-ENGLISH NEURAL MACHINE TRANSLATION: A COMPARISON OF TAGSETS

Jacob Striebel

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Arts

Department of English Language and Literature

Central Michigan University
Mount Pleasant, Michigan
October 2020

ACKNOWLEDGMENTS

I wish to thank the members of my thesis committee: Dr. Richard W. Forest, Dr. William C. Spruiell, and Dr. Lisa Gandy. These faculty members provided valuable direction when this project began, reviewed the draft of the document, and made many contributions to the final product.

ABSTRACT

PART-OF-SPEECH TAGS AS INPUT FEATURES IN GERMAN-TO-ENGLISH NEURAL MACHINE TRANSLATION: A COMPARISON OF TAGSETS

by Jacob Striebel

Since machine learning approaches began two decades ago to displace primarily linguistics-based approaches to machine translation (MT), a perennial question has existed concerning the role linguistics might continue to play in MT. One response to this question is a research agenda which has investigated using linguistic annotations as supplementary input features in contemporary machine learning translation models. Prior research has shown that use of linguistic features in neural models, such as part-of-speech (POS) tags, during training and translating yields performance improvements. This thesis expands on prior work by comparing performance among multiple POS tagsets, in contrast to testing only inclusion versus exclusion of a single tagset. The finding – for the dataset, translation direction, POS tagsets, and non-trainable network parameters tested – is that performance, as measured by BLEU, continues to increase incrementally as the total number of POS categories in the tagset increases.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

CHAPTER 1

INTRODUCTION

In the past half-decade, neural machine translation (NMT) has achieved impressive results and is currently the state of the art for most language pairs (Barrault et al., 2019). The present position of NMT with respect to statistical machine translation (SMT) is analogous to the position held by SMT vis-à-vis rule-based approaches to machine translation two decades ago. This trend, in which the most successful machine translation systems have shifted from explicitly encoding linguistic information, to inferring linguistic information from bilingual corpora by increasingly sophisticated, and computationally expensive, mathematical procedures, has raised the question of what the relationship should be between the discipline of linguistics as traditionally practiced, and the field of machine translation as it currently stands. The question of this relationship is a central theme of the present work.

The research project which was conducted and which is documented in this thesis follows within a research agenda that has investigated the effects of using linguistic features as supplementary input in NMT models – specifically, the impact of using source-side part-of-speech annotation when training and translating. Research in this domain has until now compared the effects of a simple inclusion versus exclusion of part-of-speech information as input to NMT models (e.g., Sennrich and Haddow, 2016). The present research, in contrast, investigated the effects of different part-of-speech annotation schemes: i.e., the independent variable here is the number and type of part-of-speech tags used. Three models were trained and compared, using BLEU[1] as the primary evaluation metric, with a qualitative comparison of a

1 BLEU stands for “BiLingual Evaluation Understudy” and was first introduced by Papineni et al. (2002a).

small selection of sentences translated by each model also performed. Of the three models, one was a baseline trained without using any annotation, one was trained using the Stuttgart-Tübingen Tagset (STTS), and one was trained using an abbreviated version of STTS with half as many categories as full STTS. The language pair and translation direction used in the research was German to English. The bilingual training, validation, and test corpora used were taken from sets made available through the Workshops on Machine Translation (WMT).

At the outset of the project it was hypothesized that the full Stuttgart-Tübingen Tagset (STTS), with its more fine-grained word classes, might provide performance advantages by allowing a neural model to perform implicit disambiguations more effectively and learn syntactic patterns more effectively – but the possibility was also acknowledged that use of tagging schemes with too many word classes might hurt more than help, distracting the model from learning other patterns more relevant to translation. The experiment results, presented in Chapter 4 and discussed in Chapter 5, show that the model trained using full STTS performed most favorably. But before these results can be presented, Chapter 2 first provides an overview of the historical, theoretical, software, and corpora prerequisites which ground the present research, and Chapter 3 documents the experiment procedure developed for, and used in, the present research.

CHAPTER 2

PREREQUISITES

This chapter could alternatively be titled “Literature Review”; the current title was chosen instead for two reasons: first, to emphasize the unity of the material presented here as prerequisite to the experiment procedure described in the next chapter, which is the core of this thesis; and second, to indicate that the material reviewed here is not just prior published literature, but also includes items like the software packages and the corpora used in the research.

The present chapter is divided into four top-level sections, each corresponding to a variable in the experiment: two of the variables are important control variables, one is the independent variable, and one is the dependent variable. The first section (2.1) presents the experiment’s dependent variable, which is performance in terms of BLEU. The second section (2.2) presents a vital control variable, the neural MT software system used in the experiment. The third section (2.3) presents the experiment’s independent variable, tagset – the two tagsets which are compared in the experiment, to each other and to a baseline, are described here. The fourth section (2.4) presents another important control variable, the bilingual corpora which are used to train, validate, and test the translation models. Each of these sections includes a description of its variable in terms of the variable’s role in previous research, as well as of the variable’s role in the present research.

2.1 Evaluating Machine Translation Systems

When attempting to solve a difficult problem, a useful first step is to specify the conditions under which the problem can be considered solved, so that once a candidate solution has been developed, there is a straightforward way to evaluate the candidate for correctness.

Depending on the problem, solution conditions might be immediately obvious: for the difficult problem of landing a person on the Moon and returning them safely to the Earth, the two solution conditions are already explicit in the problem statement – thus there should be little trouble judging whether, or at what point, the problem has been solved. This example, however, turns out to be a problem which starkly contrasts with that of translation – at least in terms of solution conditions at the highest level. The contrast lies in the fact that the words “Moon” and “Earth” have very obvious, well-defined referents, whereas “the English target sentence *B* ‘equivalent’ to the German source sentence *A*” will not have a similarly pre-agreed-upon referent: this is actually the problem of translation – to produce an acceptable referent *B*.

Machine translation systems are evaluated based on some notion of average quality of the *B*’s produced, given a large set of *A*’s. In the following two sub-subsections, several options are given regarding how quality can be scored, including the option that is used in the present experiment.

2.1.1 Human Evaluation

Historically, the only translations were – and today the best translations usually still are, especially for a very nuanced text – produced by human translators. Furthermore, translations are produced for the use and convenience of humans. Since expectations about what translations need to be and do depend directly on humans needs, it seems to follow that using human evaluators is the best choice for evaluating translation quality, and indeed, human evaluation is widely considered the final arbiter of machine translation quality: for example, the “premier shared task” of the 2019 Conference on Machine Translation was to build and submit a machine translation system which would compete against other submissions for performance, where

performance was based mainly on human judgement of translation quality (Barrault et al., 2019). An operationalization of human judgement of translation quality has been common for evaluating MT systems at all of the Conferences on Machine Translation since the first in 2016 (Bojar et al., 2016, 2017; Barrault et al., 2018).

There are a variety of ways human evaluation can be conducted. One option is to choose a set of conceptual metrics, such as translation fluency in the target language, and translation faithfulness to its source, and then give human evaluators a Likert scale, or something along those lines, on which to rate candidate translations (e.g., evaluators might use a five-point scale to indicate their level of agreement to statements such as “The candidate faithfully translates the source sentence”). For metrics such as fluency in the target language, evaluators would not need to have knowledge of the source language. There is also a clever strategy by which faithfulness can be evaluated by human evaluators who do not have knowledge of the source language; this can be done as follows: one person who has knowledge of both the source and target language is given the source text (but not the translation). This person reads the source, then writes, in the target language, a reading comprehension quiz for the source text. The translation text can then be evaluated on faithfulness based on how well evaluators, who need only have knowledge of the target language, score on the quiz after reading the translation text. A review of some of the human evaluation techniques for machine translation that have been proposed over time can be found in Jurafsky and Martin (2009: pp. 894-895, 906-907).

A simple, perhaps not ideal, but pragmatic and concise solution for collecting human evaluations, assuming we have access to bilingual evaluators, is to simply give our evaluators a collection of source and target sentence pairs, and using a single metric, such as “adequacy,” ask

evaluators for each pair, e.g., “On a scale of 0 to 100, how much do you agree that the target sentence adequately expresses the meaning of the source sentence?” This is the approach that has been taken and defended in work such as Barrault et al. (2019).

The most significant drawback to human evaluation is cost, perhaps most notably cost in time. Especially when an MT system is in development, and frequent testing is desirable, it will likely not be practical to often conduct any very thorough human evaluation. In the second section of the “Results” chapter of this thesis (4.2), a selection of source sentences are compared to their translations made by the different models produced in the experiment. This is done in order to illustrate some of the relative strengths and weaknesses of the three models. However, although human evaluation is typically preferred if practicable, it is often not practicable – for this reason, there are currently in wide use several automatic evaluation schemes, which have been developed with the objective of correlating as highly as possible with human evaluation.

2.1.2 Automatic Evaluation

Although work has been done on predicting a translation’s quality in the absence of reference translations (e.g., by Specia et al., 2018), the automatic evaluation metrics currently in most prominent use rely on a notion of measuring the similarity between a translation output by an MT system, and one or more reference translations produced by professional human translators. How “similarity” is defined varies by particular metric, and this variety is what

differentiates metrics. A few examples of commonly used metrics are BLEU, NIST, and CHRF.[2] In the experiment described in this thesis, BLEU is used for automatic evaluation, so the focus of this subsection is to provide a description of how BLEU judges translation quality, including its strengths and weaknesses. The objective of an automatic MT evaluation procedure, such as BLEU, is to produce a translation quality score that correlates well with human judgments of translation quality.

BLEU takes single sentences as its basic unit of evaluation. Given a candidate-translation sentence produced by an MT system, and one or more reference-translation sentences produced by professional human translators, BLEU assigns the candidate translation a score between 0 and 1 (often scaled to between 0 and 100), inclusive, based on the number of unigrams, bigrams, trigrams, and quadrigrams the candidate has in common with the reference translations (BLEU is generalizable to use unigrams up to *n*-grams of any size *n* – stopping at quadrigrams isn't a hard requirement, but Papineni et al. (2002a) found that stopping at quadrigrams yielded the best performance in their experiments). A score of 0 means that of all the unigrams present in the candidate translation, not a single one of them appears in any of the reference translations. A score of 1 almost certainly means that the candidate translation matches one of the reference translations exactly. In addition to a BLEU score being based on the number of 1- through 4-grams a candidate translation has in common with a set of reference translations, there is one

2 BLEU, as stated in n. 1, stands for "BiLingual Language Understudy"; we will further note here that the name was chosen by Papineni et al. (2002a) because the metric is intended "as an automated understudy to skilled human judges which substitutes for them when there is need for quick or frequent evaluations." NIST gets its name because it was proposed by Doddington (2002) while at the National Institute of Standards and Technology. CHRF was proposed by Popović (2015) and stands for "CHaRacter *n*-gram F-score."

more factor used in the calculation, which is a brevity penalty which reduces the score of a candidate translation that is shorter than the reference translation it is closest to in length.

The below example is taken from Papineni et al. (2002a): Candidates 1 and 2 are translations of the same Chinese source sentence produced by two different MT systems, and References 1, 2, and 3 are human translations of this same source sentence.

Candidate 1: It is a guide to action which ensures that the military always obeys the commands of the party.

Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct.

Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.

Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference 3: It is the practical guide for the army always to heed the directions of the party.

It seems relatively clear that Candidate 1 expresses the meaning of the reference sentences more fluently and fully than Candidate 2, and this is reflected in the BLEU metric, which scores Candidate 1, when measured against these three references, significantly higher than Candidate 2. In order to calculate the score for each candidate, the first step is to identify all of the 1- through 4-grams that occur in the two candidate translations, identify their counts in each candidate, and identify their counts in each reference translation; this information, for unigrams, is provided in Table 2.1. Note that when parsing a sentence for its bigrams, the tokens <first_pos

- 1> and <final_pos + 1>, which are sometimes added to a sentence when counting bigrams, are not used by BLEU. Thus the first bigram in Candidate 1 is (“it”, “is”) rather than (<first_pos - 1>, “it”), and there are a total of 17 bigrams in Candidate 1 (its length in unigrams minus one). Likewise when counting trigrams, in terms of BLEU, the tokens <first_pos - 2>, <first_pos - 1>, <final_pos + 1>, <final_pos + 2> are not used. The total number of trigrams in Candidate 1 is 16 (its length in unigrams minus two). Start and end tokens are likewise not used for quadrigrams.

Table 2.1. Unigram Counts for BLEU Example

	Candidate 1	Candidate 2	Reference 1	Reference 2	Reference 3
it	1	1	1	1	1
is	1	1	1	1	1
a	1	0	1	0	0
guide	1	0	1	0	1
to	1	1	1	0	0
action	1	0	1	0	0
which	1	0	0	1	0
ensures	1	0	1	0	0
that	1	1	2	0	0
the	2	2	1	4	4
military	1	0	1	1	0
always	1	0	0	1	1
obeys	1	0	0	0	0
commands	1	0	1	0	0
of	1	0	0	1	1
party	1	1	1	1	1
to	0	0	0	0	1
insure	0	1	0	0	0
troops	0	1	0	0	0
forever	0	1	1	0	0
hearing	0	1	0	0	0

activity	0	1	0	0	0
guidebook	0	1	0	0	0
direct	0	1	0	0	0

Once all the *n*-gram counts have been obtained, the next step is to calculate what Papineni et al. (2002a) term the *modified unigram precision* of the candidates (and subsequently to calculate the modified bigram, trigram, and quadrigram precisions). *Ordinary* unigram precision is the number of unigrams in the candidate that appear somewhere in a reference translation divided by the total number of unigrams in the candidate. In contrast, when calculating the *modified* unigram precision of a candidate, the candidate's unigrams are first subtotaled by type (orthographically identical units are considered to share type, so for Candidate 1, for example, the unigram subtotal for type "the" is 2, as seen in Table 2.1). For each reference translation, its unigrams are also subtotaled by type. Once all of these subtotals are obtained, the candidate's unigram subtotals are each "clipped" to the size of the unigram subtotal of the reference translation whose subtotal for that type is largest (i.e., for each unigram type subtotal in a candidate, $Count_{clip} = \min(Count, Max\_Ref\_Count)$; for example, Candidate 1 contains 2 unigrams of type "the"; the unigram subtotal of References 2 and 3 for type "the" is 4, and this is the largest subtotal among the three references for "the"; thus $Count_{clip}$ for the unigram "the" in Candidate 1 = min(2, 4) = 2). The sum of the candidate's clipped subtotals is divided by the sum of its unclipped subtotals to yield modified unigram precision. The same procedure is followed for bigrams, trigrams, and quadrigrams. So the modified *n*-gram precision of a candidate is given by

$$\frac{\sum_{ngram \in candidate} Count_{clip}(ngram)}{\sum_{ngram' \in candidate} Count(ngram')} \quad .$$

Modified *n*-gram precisions are used by BLEU instead of ordinary *n*-gram precisions so that candidate translations that contain too many common words don't score too highly; for example, if Candidate 1 were only composed of the unigram "the" repeated 18 times, its ordinary unigram precision would be 18/18 because "the" does appear in a reference translation – but its modified unigram precision would only be 4/18 because the *Max_Ref_Count* for "the" in any of the three reference translations is 4. The problem illustrated here with pure precision is often ameliorated, in contexts other than BLEU, by balancing precision against recall, where recall would be a measure of the number *n*-grams in the references that appear in the candidate. Using pure recall was considered undesirable for BLEU because it would cause some sentences that contained too many words, for example, sentences that contained redundant information, to score highly.

In the below example taken from Papineni et al. (2002a), Candidate A has significantly better recall than Candidate B (i.e., 5/5 of the unigrams that appear in the references appear in Candidate A, and only 3/5 appear in Candidate B), but Candidate B is clearly the better translation.

Candidate A: I always invariably perpetually do.

Candidate B: I always do.

Reference A: I always do.

Reference B: I invariably do.

Reference C: I perpetually do.

The fact that BLEU is designed to work with multiple reference translations is what motivates its choice not to use pure precision and recall as the basis of its scoring. The "modified *n*-gram precision" that BLEU does use is an alternative to precision and recall that performs well when there are multiple references available to compare a candidate to.

Once all of the modified unigram, bigram, trigram, and quadrigram precisions have been calculated for a candidate translation sentence, the next step is to calculate the same values for the next candidate translation sentence in the test corpus, then for the next and so on until modified precisions have been calculated for every sentence in the test corpus. (BLEU can be calculated for a unit as short as a single sentence; however, BLEU typically does not begin to perform well (i.e., to correlate well with human judgments of translation quality) until the test corpus is at least a few hundred sentences long.) Once all of the individual modified precision values for the entire test corpus have been obtained, four values – a total modified unigram, bigram, trigram, and quadrigram precision – are calculated for the entire test corpus by taking a weighted average across all of the sentences, where the weight factor is the word length of each sentence.

An equivalent, slightly more time efficient and significantly more space efficient alternative to first computing the modified 1- through 4-gram precisions for each sentence individually and then computing the four weighted averages is, while iterating through the sentences of the test corpus once, to maintain eight running sums, the first four being sums of the sums of the clipped subtotals of unigrams, bigrams, trigrams, and quadrigrams of each type for each candidate sentence, and the final four being sums of the sums of the *un*clipped subtotals of unigrams, bigrams, trigrams, and quadrigrams of each type for each candidate sentence; once

these eight sums have been calculated, the four modified *n*-gram precisions of the test corpus can be calculated by four divisions: by dividing the sum of the sums of the clipped subtotals of unigrams of each type for each sentence by the sum of the sums of the *un*clipped subtotals of unigrams of each type for each sentence, and by doing the same for bigrams, trigrams, and quadrigrams: i.e., p_n, the modified *n*-gram precisions of an entire test corpus, for $n = 1, 2, 3, 4$, is

$$\frac{\sum_{C \in \{Candidates\}} \sum_{ngram \in C} Count_{clip}(ngram)}{\sum_{C' \in \{Candidates\}} \sum_{ngram' \in C'} Count(ngram')} .$$

The geometric mean of p_1, p_2, p_3, and p_4 is then taken. A geometric mean is used to take into account the exponential decay that occurs from p_1 to p_2 to p_3 to p_4 (an arithmetic mean would be dominated by p_1, which by itself lacks the information about a candidate's fluency expressed in p_2, p_3, and p_4). Once the geometric mean is taken, all that remains is to calculate a brevity penalty and take its product with the geometric mean to arrive at the final BLEU score. The modified *n*-gram precisions already penalize translation candidates that are too long, but they do not punish translations that are too short. Therefore a brevity penalty is used, calculated from *r* and *c*, where *c* is the total length of the candidate translation corpus, and *r* is the test corpus' effective reference length. The test corpus' effective reference length is calculated by summing the lengths of the reference translations that most closely match the lengths of the candidate translations. If $c >$ r then the brevity penalty is 1, but if $c \leq r$ then the brevity penalty is a decaying exponential of r/c: i.e., $e^{(1-r/c)}$. Thus a BLEU score can be calculated according to the formula

$$BP * \exp\left(\sum_{n=1}^{4} \frac{1}{4} \log p_n\right)$$

where *BP* is the brevity penalty, and $\exp(x) = e^x$.

In Papineni et al. (2002a)'s presentation of the BLEU metric, in addition to defining the procedure for calculating BLEU, the results of a study are also presented which evaluated the performance of BLEU for judging Chinese-to-English translation quality. The study used 20 human (non-professional-translator) judges to provide human evaluations of translation quality for five sets of candidate English translations of Chinese sentences; three of the sets were produced by commercial MT systems, and two of the sets were produced by non-professional human translators. Two additional sets of professional human translations were used for references. Papineni et al. (2002a) found a statistically significant correlation between BLEU's judgments of translation quality and the human judgments of translation quality for all of the five sets of candidate translations. Papineni et al. (2002b) also conducted a similar study, but one that expanded its language pairs to evaluate, in addition to Chinese-to-English, also Arabic-, Spanish-, and French-to-English, finding similar correlation between human judgments and BLEU's judgments as Papineni et al. (2002a).

BLEU is, however, not perfect. One drawback that has already been mentioned is that a BLEU score is only meaningful in relation to a relatively large test corpus. Papineni et al. (2002a) state this as "*quantity leads to quality*," the takeaway being that one often can't very meaningfully talk about the BLEU score of a single sentence, or a single paragraph, or maybe even a single document, depending on the document's size. Also, BLEU is known to perform poorly (i.e., not correlate well with human judgments) when used to compare systems with divergent architectures, such as a purely statistical *n*-gram-based system and circa 2009 Systran, which was a hybrid rule-based–statistical system (Jurafsky and Martin, 2009: p. 897). Some

researchers have argued that BLEU is better described as a document similarity metric rather than as a translation quality metric (Zhang et al., 2004).

But in spite of BLEU's drawbacks, it remains in wide use and is considered a good-enough solution in many cases, particularly for evaluating incremental changes to a system that maintains the same general architecture, which is how BLEU is used in the present research.

2.2 Building Machine Translation Systems

In the history of machine translation, there have been two paradigms that have been dominant for building MT systems (with some overlap between the two possible). A human language can be thought of as a system of patterns in which certain patterns and interrelations of patterns map to certain meanings. In one approach to machine translation, the MT system builder attempts to explicitly specify a mapping from the patterns of one language to the patterns of a second language that share the same meaning. This is the classic approach to MT and is termed "rule-based" machine translation; this approach is described in subsection 2.2.1. The second approach shares with the first the objective of establishing a mapping from the patterns of one language to the those of a second with the same meaning, but the strategy used for establishing the mapping is different; in the second approach, the MT system builder works to specify a general procedure by which the mapping can be inferred from data (i.e., a large bilingual corpus), instead of attempting to specify the mapping directly. This second approach has received the most attention in recent MT research and is the one taken in the experiment described in the next chapter. This data-driven approach to MT system building is described in subsections 2.2.2 and 2.2.3. Subsection 2.2.4 presents the particular MT software framework used in the present research.

2.2.1 Rule-Based MT

The classic rule-based approach to MT was the first approach to be applied to the problem of machine translation. An MT system is considered rule-based if linguistic information is incorporated into the system explicitly in the form of linguistic transformation rules. There are a variety of strategies that have been proposed over time for structuring and executing translation according to linguistically motivated transformation procedures. One scheme for categorizing the various types of procedures, which is widely cited in the literature, was proposed by Vauquois (1968; cited by, e.g., Jurafsky and Martin, 2009: p. 867). In Vauquois' taxonomy, there are three basic categories of MT system: direct, transfer, and interlingua.

In systems in the direct category, source sentences are translated word by word into the target language using a bilingual dictionary. Dictionary entries might simply map a source-language word to a single target-language word – or a dictionary entry might be more complex, for example, providing multiple translations for a single word, which are chosen among depending on the word in the source sentence that precedes or follows the word being translated. The only processing other than use of the bilingual dictionary that occurs in the direct approach is morphological analysis to facilitate dictionary lookup, and morphological generation once target lemmas have been chosen – and also, importantly, simple word reorderings might be performed, such as moving adjectives to before nouns if translating from Spanish to English, or moving participles to the final position in an independent clause if translating from English to German. However, a rule like this second, which is grammatically necessary, would tend to be somewhat unwieldy in a direct system, because direct systems operate only on words, not on any higher-level grammatical structures, such as clauses, so this second rule would have to be written

something like: participles are moved from their original position to the final position before the next punctuation mark. There are many cases, however, when this particular articulation of the rule wouldn't hold – one simple example being if there is an embedded relative clause, set off by commas, after the participle but still within the independent clause. In order for the rule to work correctly in a wide range of cases, it would need to be made more complex. This is an example of possibly the most significant disadvantage of direct MT systems: in order for them to be very successful, many of their rules need to be quite complex, accounting for the wide range of legitimate possibilities in the structure of source sentences, and of the target sentences being generated; however, because direct MT has no unifying higher-level grammatical view of sentences, it can become difficult to judge and debug the interactions of rules that are of necessity complicated, but essentially ad hoc.

There is a solution to this problem presented by the transfer category of MT systems, which is to introduce a higher-level grammatical representation. In the transfer approach, a source sentence is first parsed into a form that shows the syntactic relationships between its words, as in a constituency-grammar tree. This source-language parse is then transformed to a target-language parse, according to a set of transformation rules. In the direct approach, the MT system's transformation rules operate on individual words, whereas in the transfer approach, transformation rules operate on syntactic units (although the transfer approach, just like the direct approach, uses a bilingual dictionary for translating individual words). Transfer systems improve upon direct systems in the way they manage structural differences between languages, such as word order, but there are still a variety of problems that they do not solve very well, for example, a purely syntactic approach is generally inadequate for translating idioms. And, of course, the

transfer approach requires the development of a good method for parsing a source language's syntax, and for flattening target-language syntactic parses.

The final category in the Vauquois hierarchy is interlingua. Interlingua is possibly the most intuitive approach to rule-based MT: since humans are the best translators, this approach says, if we want a computer to translate well, we should program it to emulate the procedure followed by a human translator, which is to read a sentence in the source language, understand its meaning, and then produce a sentence in the target language that expresses that meaning. In the interlingua approach, translation rules do not describe how to translate from a source language to target language, rather they describe how to translate from a source language to a semantic intermediate representation, and from the intermediate representation to a target language. The intermediate representation is the "interlingua." The most attractive aspect of the interlingua approach is that it simplifies the process of building highly multilingual MT systems. Suppose there are 8 languages that need to be translated among freely. Using the direct or the transfer approach, 56 rule-sets are needed; however, using an intermediate representation, only 16 are needed – one to translate from each language to the interlingua, and one to translate from the interlingua into each language. As the number of languages that need to be translated between grows, the savings of using an interlingua approach over a transfer approach, in number of needed rule-sets, grow at a rate with an upper bound of n^2. However, this isn't the entire story. As the number of languages that an interlingua needs to support grows, the amount and type of semantic information that the interlingua needs to encode likewise grows, which in turn requires increases in the size and complexity of the rule-sets and dictionaries that are used for translating into the interlingua. In practice, interlingua approaches are only feasible for relatively small

language-subsets; however, incorporating aspects of semantic analysis that are used in interlingua approaches into a transfer approach has sometimes produced encouraging results. For example, the old rule-based commercial Systran system used, in addition to syntactic transfer rules, also semantic analyses such as word-sense disambiguation and translation of idioms (Senellart et al., 2001).

2.2.2 Statistical MT

As early as 1949, it was suggested that statistical methods and ideas from information theory might be applied to the problem of machine translation (Brown et al., 1990: p. 79); it took half a century, but this has now become the current direction in MT research. This approach to machine translation that in the ’90s began to challenge, and eventually displace, rule-based MT is termed statistical machine translation (SMT); its chief impetus was a 1993 paper authored by Brown et al., researchers at IBM, titled “The Mathematics of Statistical Machine Translation.” In the paper, a procedure is presented for generating word(s)-to-word(s) alignments between parallel sentences in bilingual corpora by purely statistical means.[3] (The key to Brown et al.’s procedure is a clever application of the Expectation Maximization algorithm published by Dempster et al. in 1977.) SMT translates a source sentence to a target sentence by finding the target sentence which maximizes a probability function that balances the target’s fluency as a sentence in the target language and the target’s faithfulness to the source sentence. Fluency in the target language is represented using a monolingual *n*-gram model, and the model representing faithfulness between a target sentence and source sentence is assembled using the word

3 As a prerequisite to the word-alignment research of Brown et al. (1993), several papers had already been published describing procedures for aligning sentences in bilingual corpora. This research is reviewed in subsection 2.4.1.

alignments in a bilingual corpus generated using Brown et al.'s procedure. The search procedure that is used to find the suitable target sentence is called the *decoding algorithm*, one version of which was specified in an IBM patent, filed in 1991 and approved in 1995 (Brown et al., 1995). The decoding algorithm uses a beam search, a heuristic greedy search algorithm: a heuristic solution is necessary to cope with a search space which must in effect include all possible sentences in the target language. Within SMT, considerable research has been done on phrase-based models as an improvement on word-based models (e.g., Koehn et al., 2003). SMT was essentially the exclusive MT paradigm of study from 2000 until the mid-2010s; however, between about 2014 and 2016 artificial-neural-network approaches (NMT) began to perform equally well, and then better, than SMT.

2.2.3 Neural MT

Neural machine translation (NMT) can be classified, technically, as a type of statistical machine translation. In NMT, just as in SMT, the free parameters of statistical models are estimated from data – the difference is only in the architecture of the models. One of the first papers to report a successful application of a neural-network architecture to machine translation (Kalchbrenner and Blunsom, 2013) describes its method as a "statistical [approach]." Moreover, Cho et al. (2014b) describes NMT as "a new approach to statistical machine translation." However, the difference in model architecture between SMT and NMT is not insignificant: while SMT uses corpus data to estimate the free parameters of a target-to-source faithfulness probability model, which then serves as a key subcomponent in the translation system, the architecture of artificial-neural-network models is more general in its pattern-inferential ability,

allowing NMT systems to be trained end-to-end, rather than being assembled from independently trained subcomponents.

The architecture of contemporary NMT systems tend be what is termed “encoder–decoder” – this architecture appears to have been first proposed by Cho et al. (2014a). An “encoder” is a neural network that takes a source-language sentence as input, and produces a vector encoding of that sentence. A “decoder” is a neural network that takes an encoder-produced vector-encoding as input and outputs a sentence in the target language. The encoder and decoder networks are trained jointly to maximize the probability of producing the target sentences, given the source sentences that are contained in a bilingual training corpus. Within this architecture, there are a variety of design options: the format of the intermediate vector-encoding is one item that needs to be decided on; some of the first proposals (e.g., Sutskever et al., 2014) used fixed length vectors, but Bahdanau et al. (2015) found that a variable-length encoding scheme improved performance, most markedly for longer sentences. Also the configuration of the encoder and decoder networks can be varied; recurrent neural networks and convolutional neural networks have both been used successfully – but Vaswani et al. (2017) suggested that use of an attention mechanism can make recurrences and convolutions unnecessary, decreasing training cost in GPU time and improving performance.

2.2.4 The MT System in the Present Research

The present study used Nematus, a free and open-source neural machine translation program, which is an effort of the University of Edinburgh’s Natural Language Processing Group. Among Nematus’s advantages is support for translation from multiple input features, which is essential to the research conducted under this thesis. Nematus is described in multiple

published works, some of which are listed in the documentation file in the root directory of the project's GitHub repository.[4] The particular network parameters used in the present research are described in subsection 3.3.2.

2.3 Part-of-Speech Tagging

The practice of defining a set of word classes for a language, with the intention that every word in the language corresponds to a word class, is more than two thousand years old. Word classes, when known, can be used to explicate the structural and meaning relationships between the words in a sentence. Words are placed in the same class if, when they appear, they tend to play similar roles. One of the most straightforward and important tests for word class membership (i.e., determining if two words do "play similar roles") is the interchangeability test – if a word can frequently be substituted for another word in a sentence, with the sentence remaining a valid sentence in the language, then the two words likely share the same word class (although different linguists have different ideas about what the exact rules of valid inference should be when reasoning about interchangeability).

Word classes, which are also called parts of speech, have a wide range of applications in linguistics research and in natural language processing. Knowing a word's part of speech in a sentence is often valuable, because a part of speech frequently encodes much more information than what is present in the word's naked orthographic form. One example given by Church (1988) of information that can sometimes be deduced from part of speech is pronunciation: e.g., the vowel in the word "wind" depends on whether the word is a noun or a verb. Assigning the

4 https://github.com/EdinburghNLP/nematus

words of a sentence part-of-speech labels, or *tags*, can be an important step in disambiguating a sentence for a variety of computational-linguistics applications.

The present section has four subsections, which introduce part-of-speech categories as they relate to the present research. The first subsection (2.3.1) covers the two German part-of-speech categorization schemes, *tagsets*, which are used by the experiment. The second subsection (2.3.2) discusses approaches to automatically assigning parts of speech to words in the context of a sentence. The third subsection (2.3.3) explains how tagging is performed in the tagset-comparison experiment which is the subject of this thesis. And the final subsection (2.3.4) explains the motivation behind this experiment.

2.3.1 Tagsets

A wide range of tagging schemes have been used for different languages over time; nevertheless, the basis of all of these schemes tends to remain the set of eight categories proposed for Greek by Dionysios Thrax of Alexandria (Thrax, 1874) in the second century BC. Thrax's categories are noun, verb, participle, article, pronoun, preposition, adverb, and conjunction. The two tagsets, which are described in the next two paragraphs, mostly maintain the classical eight categories as their base, providing some tweaks and additions to these categories, but, more often than not, simply partitioning the original eight categories into sets of subcategories (allowing for more fine-grained analyses), without modifying the boundaries between the top-level originals.

The first tagset used in the experiment is the Stuttgart-Tübingen Tagset (STTS), which uses 54 categories and is designed for use with contemporary German (Schiller et al., 1995,

1999).[5] The 54 categories of STTS are subclasses of 11 top-level categories: noun, verb, article, adjective, pronoun, cardinal, adverb, conjunction, adposition, interjection, and particle (there are also a few miscellaneous categories for things like punctuation).[6] The top-level category for adjectives, for example, is subdivided into separate categories for inflected, attributive adjectives and for uninflected, predicate adjectives.

The second tagset used is an abbreviated version of STTS employing 27 categories. Each category in full STTS is a subclass of one category in the abbreviated tagset; some categories in the abbreviated tagset have multiple subclasses, while others have only one (i.e., are equivalent to a single category in full STTS). For example, the categories attribute adjective and predicate adjective in STTS are represented by only one category, adjective, in the abbreviated tagset; however, the category adverb in STTS maps to exactly one category, adverb, in the abbreviated tagset. Table 2.2 lists all categories used in each tagset and indicates the mapping of STTS categories into their superclasses in the abbreviated tagset.

Table 2.2. STTS Categories and Abbreviated STTS Categories[7]

STTS		Abbreviated STTS	
ADJA	attributive adjective	ADJ	adjective
ADJD	predicative or adverbial adjective		
ADV	adverb	ADV	adverb
APPR	preposition; circumposition (left part)	AP	adposition
APPRART	preposition with fused article		

5 Zinsmeister et al. (2014) note that "The Stuttgart-Tübingen TagSet (STTS) is a de-facto standard for the part-of-speech tagging of German texts."

6 One location where documentation on STTS can be found is https://www.ims.uni-stuttgart.de/en/research/resources/lexica/germantagsets/.

7 The English description provided in the table for each category in STTS is taken from the document "Deutsches Wortart-Tagset STTS/German Part-of-Speech Tagset STTS," available at https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/STTS-Tagset.pdf.

APPO	postposition	"	"
APZR	circumposition (right part)		
ART	definite or indefinite article	ART	definite or indefinite article
CARD	cardinal number	CARD	cardinal number
FM	foreign language material	FM	foreign language material
ITJ	interjection	ITJ	interjection
KOUI	subordinating conjunction followed by “zu” and infinitive	KOUI	subordinating conjunction followed by “zu” and infinitive
KOUS	subordinating conjunction followed by clause	KOUS	subordinating conjunction followed by clause
KON	coordinating conjunction	KON	coordinating conjunction
KOKOM	comparative conjunction	KOKOM	comparative conjunction
NN	simple noun	N	noun
NE	proper name		
PPOSS	substitutive possessive pronoun	PPOS	possessive pronoun
PPOSAT	attributive possessive pronoun		
PRELS	substitutive relative pronoun	PREL	relative pronoun
PRELAT	attributive relative pronoun		
PWS	substitutive interrogative pronoun	PW	interrogative pronoun
PWAT	attributive interrogative pronoun		
PWAV	adverbial interrogative or relative pronoun		
PDS	substitutive demonstrative pronoun	PO	other pronoun
PDAT	attributive demonstrative pronoun		
PIS	substitutive indefinite pronoun		
PIAT	attributive indefinite pronoun without determiner		
PIDAT	attributive indefinite pronoun with determiner		
PPER	irreflexive personal pronoun		
PRF	reflexives personal pronoun		
PAV	pronominal adverb		

PTKZU	“zu” particle before infinitive	PTKZU	“zu” particle before infinitive
PTKNEG	negation particle	PTKNEG	negation particle
PTKVZ	separated verb prefix	PTKVZ	separated verb prefix
PTKANT	answering particle	PTKANT	answering particle
PTKA	particle before adjective or adverb	PTKA	particle before adjektive or adverb
TRUNC	first element of a (truncated) compound	TRUNC	first element of a (truncated) compound
VVFIN	finite full verb	V	verb
VVIMP	imperative (full verb)		
VVINF	infinitive (full verb)		
VVIZU	infinitive with incorporated “zu” particle (full verb)		
VVPP	past participle (full verb)		
VAFIN	finite verb (haben, sein, werden)		
VAIMP	imperative (haben, sein, werden)		
VAINF	infinitive (haben, sein, werden)		
VAPP	past participle (haben, sein, werden)		
VMFIN	finite modal verb		
VMINF	infinitive (modal verb)		
VMPP	past participle (modal verb)		
XY	non-word containing special symbols	XY	non-word containing special symbols
$,	comma	$,	comma
$.	sentence-final punctuation	$.	sentence-final punctuation
$(	other punctuation	$(	other punctuation

2.3.2 Automatic Tagging

Like machine translation, the history of automatic tagging – that is, the development of software to assign parts of speech to words within sentences automatically – goes back to the middle of the last century. The first part-of-speech assignment algorithm is possibly the parser in

Zellig Harris's Transformations and Discourse Analysis Project, which was implemented between 1958 and 1959 (Jurafsky and Martin, 2009: p. 169). Harris's system used a dictionary that contained parts of speech for words, but the system was also able to perform part-of-speech disambiguation using 14 hand-written rules, which took into account an estimate of the relative frequencies of particular tags for particular words (Jurafsky and Martin, 2009: p. 169). Subsequent automatic tagging systems likewise relied on dictionaries combined with rule-sets for disambiguation. Then, a little before statistical methods were applied to MT, statistical methods began to be applied to the problem of automatic tagging. One influential stochastic procedure was published by Church (1988), which used a hidden Markov model approach. Also influential research, published not long after, was Brill (1992, 1995), which introduced a hybrid machine-learning–rule-based tagger. The tagger architectures proposed by Church and Brill continue to achieve very good results, often still performing as well, or nearly as well, as contemporary neural-network model approaches, such as Nguyen and Verspoor (2018).

2.3.3 The Tagger in the Present Research

The corpora used in the present research – the contents of which are described in subsection 2.4.2 – were conveniently obtained pretagged in CoNLL format, which includes two columns for POS tags: one of the columns contained STTS tags, while the other contained tags from the abbreviation of STTS described in subsection 2.3.1. The tagger with which the corpora were pretagged is ParZu, which is free and open-source software provided by the Computational Linguistics Group at the University of Zurich, and which achieves tagging accuracy competitive with state of the art. Thus, executing tagging of the training, validation, and test corpora was not a step required in the present study; however, in the next two paragraphs two taggers are

described which were originally evaluated for the job of tagging the corpora, and which could be used to conduct the experiment described in this thesis, generalized for arbitrary (not-pretagged) corpora.

The tagger evaluated for use with STTS was TreeTagger described in Schmid (1994, 1995), which is available free, for non-commercial use, at Schmid's webpage at the University of Munich. TreeTagger is a hidden Markov model tagger which uses a decision tree to help estimate parameters, in a similar vein as the Brill Tagger (Brill, 1992, 1995). TreeTagger achieved 97.5% accuracy on a corpus of articles from the newspaper *Stuttgarter Zeitung*.

An additional tagset that was originally considered for use in the experiment was the Universal Dependencies Project's POS tagset (UPOS), which employs 17 categories. The tagger evaluated for use with UPOS was the jPTDP tagger-parser described in Nguyen et al. (2017) and Nguyen and Verspoor (2018), which is available on GitHub under the GNU General Public License. jPTDP uses a neural-network model architecture, and achieved 94.07% accuracy tagging the UD-German-GSD corpus, which is composed of 292,000 sentences in the genres news, reviews, and Wikipedia. The abbreviated STTS tagset described in subsection 2.3.1 was chosen for use in the experiment over UPOS because no significant advantage was hypothesized to using one of these tagsets over the other, and using the abbreviated STTS tagset reduced the complexity of conducting the experiment, since the corpora are available pretagged with this tagset.

2.3.4 Motivating Part-of-Speech Tags as Input Features in NMT

Subsection 2.2.4 – which presented the the neural machine translation software used in the present research, Nematus – included a statement that draws attention to Nematus's support

for training and translating using multiple input features, on the source side. In the present section, 2.3, the additional input features that are used by the present research are introduced – part-of-speech tags. In the current subsection, a justification for this choice of additional input features is given.

Although SMT and NMT have shifted much of the focus in MT research from linguistics to mathematics, there are a variety of areas (e.g., preprocessing, annotating, parsing corpora) where more traditional linguistics remains visible in the discipline: in particular, one research agenda has investigated the effects of using different linguistic features (such as part-of-speech tags) as additional input features in statistical and neural translation models. This research has frequently concluded that linguistic input features provide measurable improvement in statistical and neural translation models, particularly for languages that display traits such as morphological richness and homonymy. For example, Sennrich and Haddow (2016) found that using lemma annotations, part-of-speech tags, and syntactic dependency labels as input features to an NMT model significantly improved the model's performance over baseline for translation between English and German, and for translation from English into Romanian. The present research continues in this vein for POS annotation, now making a comparison beyond a simple inclusion versus exclusion of explicit POS information, comparing the performance of multiple candidate POS annotation schemes.

2.4 Corpora

In the context of linguistics research, a corpus is a collection of texts that share something in common – the texts might be on the same topic or be thought of as being in the same genre, or the commonality might be something else; the main thing the texts have in common could just be

that they're all written (or spoken) in the same language. The purpose of a corpus is to provide data from which conclusions can be drawn about the thing the corpus' texts have in common. For example, someone might gather a corpus of inscriptions, which have in common that they are all written in the same ancient Near-Eastern undeciphered script, with the objective of drawing conclusions about the language encoded by the script, such as its language family or its grammar. The procedure by which a linguist draws conclusions from corpora might vary by linguist and school, but there are some practices that have become increasingly common across the board, such as working with corpora in computer-readable form, and using computer tools to conveniently perform many types of analysis that would have been very labor intensive, or even intractable, with paper and pencil. The rise in the availability of computer-readable texts is one factor that lead to the application of statistical methods to problems in linguistics. The availability of digital multilingual corpora – such as transcriptions of legislative proceedings in Canada, the EU, the UN, and Hong Kong – was, and is, obviously critical in the origination and ongoing development of SMT and NMT. In this section, the corpora prerequisites for the experiment described in the next chapter are reviewed.

2.4.1 Segmentation and Alignment

When computer-readable corpora are originally published, they are usually formatted as simple plain text. The first step in nearly any analysis then is to perform word segmentation – that is, to determine word boundaries within the corpus. This is a difficult problem for some languages, such as Chinese, that don't encode word boundaries orthographically; however, for English it is not such a difficult problem, since word boundaries are usually signaled by white space. Nevertheless, there are still some cases in English when this doesn't hold: when

segmenting English, decisions need to be made such as whether and when hyphenated forms should be treated as one or multiple words, and how to treat contractions and possessives – e.g., is the possessive *'s* an inflectional ending or a separate word? Similar decisions need to also be made when segmenting German. Segmentation of English and German is often accomplished, without too much difficulty, in a rule-based fashion.

Once word, and sentence, boundaries have been established in a corpus, if the corpus is bilingual, a useful second step is to determine an alignment between the sentences that translate each other (with the eventual goal being to line up words and phrases that translate each other), setting aside the material that doesn't align well to anything. In a bilingual corpus that had been prepared with the most painstaking care, the number of sentences in each language might originally be the same, making this step unnecessary, but in practice, large bilingual corpora contain noise – they don't have the same number of sentences in both languages. There are several statistical procedures available for aligning sentences. For example, Kay and Röscheisen (1993) describes a procedure that uses lexical information, determining a sentence alignment by finding a partial word alignment, based on the distribution of words in the parallel texts. Other procedures, which use no lexical information, include Brown et al. (1991), which aligns sentences based only on their lengths in words, and Gale and Church (1993), which aligns sentences based on lengths in characters.

2.4.2 The Corpora in the Present Research

Four bilingual corpora are used in the present research: one for training, one for validation, and two for testing. The corpora are mostly made up of news text and transcriptions of legislative proceedings from the European Parliament. The corpora were each originally

assembled and distributed in conjunction with one of the Workshops, or Conferences, on Machine Translation. The training corpus consists of about 4.2 million German-English sentence pairs, which were made available for the First Conference on Machine Translation, in 2016. The validation corpus is made up of about 3,000 German-English parallel sentences and was made available for the 2013 Workshop on Machine Translation. The first test set is made up of about 2,200 German-English parallel sentences and was made available for the 2015 Workshop on Machine Translation, and the second test set contains about 3,000 sentences in each language and is from the 2016 Conference on Machine Translation.[8]

8 The corpora are obtained from http://data.statmt.org/rsennrich/wmt16_factors/.

CHAPTER 3

PROCEDURE

The previous chapter provided a review of the research, software, and corpora prerequisites that are needed to perform the experiment which is the subject of this thesis. The current chapter gives a thorough description of the experimental procedure developed for, and followed in, the present research. The objective of this chapter is to provide everything that will be needed to reproduce the results of the experiment,[9] which are preliminarily reported in Chapter 4, and examined in further detail in Chapter 5.

The present chapter is divided into four sections. The first section (3.1) gives a description of the computer-system environment that was used to conduct the experiment, and an explanation is given of how to set up this environment. The second section (3.2) describes how to obtain and preprocess the the corpora for training, and for performing evaluation once training is complete. The third section (3.3) gives the procedure for executing model training. The fourth and final section (3.4) describes the procedure for evaluating the models once trained.

3.1 Setting up the Environment

Contemporary deep learning models, such as the recurrent neural networks used in the present research, are computationally very expensive to train. Luckily for machine learning researchers, however, the type of computations required for model training can be largely represented as matrix and vector arithmetic operations, which can be executed, massively in

9 Note that when training a neural network, the procedure for initializing the network's weights and biases contains a random component, and, since the many-parameter function, which is the neural network, is not guaranteed to converge to a global minimum of its cost function during training, a reproduction of the experiment procedure described in this chapter may not yield exactly the same results as those reported here, although it may be expected that a reproduction will produce results extremely close to the report given here. This point is further addressed in section 5.2.

parallel, on graphics processing units (GPUs). A GPU is processor, originally developed for computer graphics applications, such as rendering photorealistic video games, which is optimized to execute linear vector-arithmetic programs, whereas a CPU is optimized to execute programs that have many conditional branches – branching programs being more common, in general. It has been commented that the video game industry subsidized the present boom in deep learning by developing and making widely available GPUs as relatively inexpensive commodity hardware (Chollet, 2018). This is all to say that an environment which is set up to conduct the present experiment must contain a GPU (if only a CPU were used, training each model might take over a month, in contrast to the four days per model in the environment which is described here).

First of all, practically speaking, the GPU chosen must be produced by Nvidia and must be CUDA-enabled. CUDA is the interface used by most machine learning libraries, including TensorFlow – and Nematus uses TensorFlow as its own interface, thus Nematus depends on CUDA. Among CUDA-enabled Nvidia GPUs, the particular specification item which proved most relevant to the present research was on-board memory. A Nvidia RTX 2070 GPU with 6 GB memory was the board chosen as part of the original test environment. Each translation model, summing the size for all parameters, is no larger than 1.5 GB. However, the manner that the software (Nematus, TensorFlow, CUDA) structures the model and other program data in memory during training expands far beyond the size of the model when it's simply sitting on disk. In fact, when using the RTX 2070, training always failed during program initialization with a memory allocation error: the 6 GB of on-board memory were exhausted, and the program still needed more. Subsequently, a Nvidia K80 GPU with 12 GB memory was tested. Using the K80,

training could only complete one epoch until memory allocation failure, and consequent program termination, occurred. Finally, a Nvidia T4 GPU with 16 GB memory was tested; the T4 was the only GPU tested that was able to successfully train the three models, each in turn, to convergence. Therefore, it is recommended that the GPU chosen includes at least 16 GB of integrated memory (the hard line to train the models described in this thesis would appear to be somewhere between 12 and 16 GB).

One decision, which will be made alongside the choice of GPU, is the choice of base computer system into which the chosen GPU will be integrated. There are a few options available here which can be divided into two main categories: purchasing a physical machine and GPU, or leasing a machine and GPU through a cloud service. The first category of option is more cost effective for applications where the system will be in consistent use for a longer period of time – at minimum, several months. The second category of option is more cost effective when the demand for a system is expected to be either short term, or largely intermittent. In the present experiment, a total of only about two weeks of GPU-time were needed to train the three models, so unless a computer installed with a high-memory Nvidia GPU is already on hand, it likely makes most sense to lease, which is what was done for the present study.

There are a relatively wide range of vendors that offer GPU-equipped virtual machines (VMs) for lease in the cloud. A few options were investigated. Amazon Web Services was one option that was considered; however, the AWS billing structure for GPU-enabled VM leases proved inamicable to the research project's relatively limited computing-resource requirements. Another option that was investigated, and tested out, was Microsoft Azure. At the time the study was carried out, a VM with a Nvidia K80 GPU installed could be leased competitively on Azure.

One of these VM instances (which in the Azure nomenclature are labeled “NC6”) was briefly leased to do training, but it was learned, in the span of about a day, that the K80’s 12 GB of integrated memory is insufficient to train models with the parameters chosen. Although the Azure GPU-equipped VM type that used the K80 was a good fit for the project – that is, assuming its memory had been sufficient – and was competitively priced, it is surprising that Azure offers no middle option between this GPU VM and their highest price points (in order to use a GPU with more than 12 GB of memory through Azure, one would have to lease an “ND6s” model, which uses the Nvidia P40 GPU with 24 GB memory, and which, even given its specifications, is not very competitively priced).[10]

In the end, all three translation models were trained on a VM accessed through the Google Cloud Platform (GCP). GCP, in contrast to Azure, does not prepackage GPUs into VM models; rather, a GPU can be configured as an add-on to a variety of VM types. The VM used was the “n1-standard-4” model, and the GPU added on was a Nvidia T4;[11] a 250 GB disk was used, which proved sufficient to hold all of the intermediate save-points for the three translation models during training.

Once hardware has been chosen (i.e., VM model and GPU model), the next choice is operating system. The only restriction here is that the OS be Unix/Linux based and that Nvidia provides a GPU driver and CUDA library for it. The set of supported operating systems can be checked on the web page from which the GPU driver and CUDA library install scripts may be

10 There actually may be an intermediate option through Azure – for example, there is the “NP10s” model which sits at an intermediate price point, but, strangely, the specification of this model, as well as that of a few other Azure VM models, is not given in the regular documentation.

11 Note that before a GPU can be attached to a GCP VM, one must submit a request for one’s GPU quota to be increased from its original 0.

downloaded.[12] Once a VM instance has been created with one of the supported operating systems, the driver and CUDA library should be installed according to the instructions provided on the installer download page. One convenience of GCP is that when creating a VM, one of the operating systems that can be chosen is “Deep Learning on Linux,” which is a recent Debian Linux image with the Nvidia GPU drivers and CUDA preinstalled.

At this point, the next requirement is installing the Python programming language and the TensorFlow GPU library. Common practice for doing this is to use a Python environment management tool, such as Anaconda.[13] The version of Nematus used in the present research was validated on Python 3.5.2, so the version closest to this, which is available, is what should be installed. If using Anaconda, this can be accomplished by running

```
$conda create -n nematus_tf python=3.5
```

.

TensorFlow can then be installed with

```
$conda activate nematus_tf
```

```
$python -m pip install tensorflow-gpu
```

.

The final two environment requirements are to download the Nematus-NMT program itself, along with the scripts to do corpora preprocessing, translation-model training, and model evaluation once training is complete. These are all packaged in a GitHub repository,[14] which can be cloned or downloaded.

12 https://developer.nvidia.com/cuda-downloads
13 Anaconda can be downloaded from this page: https://www.anaconda.com/products/individual
14 https://github.com/striebel/eng799_2020_thesis

3.2 Obtaining and Preprocessing the Corpora

A description of the corpora used in the present research is given in subsection 2.4.2. In the current section, an explanation of how to obtain and preprocess the corpora is given. (If any fine-grained details are omitted in this section, or anywhere subsequently in this chapter, the scripts referenced provide the most complete and unambiguous documentation of what is being done.)[15]

The corpora are hosted at statmt.org. If the "eng799_2020_thesis" repository has been downloaded, a script is provided, "wmt16-scripts/factored_sample/download_corpus_files.sh," which can be run to automatically download the needed corpora files. In the same directory as this script, once all corpora files have downloaded, a second script, "decompress_corpus_files.sh," can be run to decompress all the just-downloaded files.

Once these files have been obtained and decompressed, the next, and only remaining, preprocessing step is to execute the "preprocess.sh" script from within the "wmt16-scripts/factored_sample" directory. This script may take a few hours to complete and performs all the formatting of the corpora that is necessary to run the experiment. The "preprocess.sh" script performs the following actions: tokenization and truecasing (done by the mosesdecoder library); further segmentation using byte-pair encoding (BPE; for open vocabulary translation – done by the subword-nmt library); extraction of the STTS tags and the abbreviated STTS tags from CoNLL format (which is the original format of the corpora when downloaded), and creating new POS-annotated corpora files in the format accepted by Nematus.

15 Importantly, please note that most of the scripts referenced in this chapter are not written by me from scratch, but are adaptations of scripts made available by other authors; whenever a file in the "eng799_2020_thesis" repository is not originally mine, its source is cited in its commit history.

3.3 Training the Models

The objective of the present research, as has been stated, is to make a comparison among three translation models, where each is trained from the same corpus but where for training the first model, the corpus is annotated with STTS tags, for training the second model, the corpus is annotated with abbreviated STTS tags, and for training the final, control model, no annotation is used.

The present section discusses the particulars of model training. In its first subsection (3.3.1), the information necessary to execute training of each model is given. In the second subsection (3.3.2), an explanation of the architectural and training parameters of the three neural models are given.

3.3.1 Executing Training

A separate script is provided in the "wmt16-scripts/factored_sample" directory to train each model: "train_udts.sh," "train_stts.sh," "train_ctrl.sh." If using the hardware discussed in section 3.1, each of these scripts should take about four days to complete. It is convenient to execute these scripts disattached from any terminal or session – i.e., so that one can secure shell into the leased VM, execute the script, then disconnect, with the training process continuing to run on the VM even though the user session has ended. One way to do this is to execute the training scripts under the "nohup" command, which is what was done.

3.3.2 Non-trainable Network Parameters

In this subsection the non-trainable network parameters are given which specify the architecture of and parameterize the training of the three models.[16] The parameters given here are also documented in the three training scripts referred to in the preceding subsection.

The basic architecture is an attentional encoder-decoder. The encoder and decoder networks are both recurrent neural networks. The total size of the embedding layer for each of the models is 500. In the control model, 493 of these nodes are assigned to words and 7 are assigned to BPE labels. In the STTS model, 467 of the nodes are assigned to words, 7 are assigned to BPE labels, and 26 are assigned to POS tags. In the abbreviated STTS model, 467 nodes are assigned to words, 7 to BPE labels, and 26 to POS tags. The hidden layers are of size 1024, a maximum sentence length of 50 is used, training minibatches of size 80 are used, the learning rate is set at 0.0001, and Adam is used as the optimizer.

3.4 Evaluating the Models

As discussed in subsection 2.1.2, once the three models have been successfully trained, the evaluation metric which is appealed to for a comparison of quality among the models is BLEU. The test corpora against which BLEU is calculated are “newstest2015” and “newstest2016.” The contents of these corpora are described in subsection 2.4.2. The present section describes usage of the script which was written for calculating BLEU against these corpora.

The script used for evaluation is “evaluate.sh” and is in the “wmt16-scripts/factored_sample” directory. evaluate.sh requires two command-line arguments, the first

16 The parameters are based on those used by Sennrich and Haddow (2016).

being the model to evaluate (“ctrl” or “udts” or “stts”), and the second being the corpus to evaluate against (“2015” for newstest2015, or “2016” for newstest2016). The evaluation script is run six times, and the results are reported in Table 4.1.

The steps performed by evaluate.sh are the following. First, the selected preprocessed test corpus is translated by the model. Once all the sentences in the test corpus have been translated, BPE segment merging, detruecasing, and detokenization are performed. Finally, BLEU is calculated against the selected corpus using a script provided by the mosesdecoder library.

CHAPTER 4

RESULTS

Having followed the experiment procedure described in the preceding chapter, the present chapter reports the results obtained. The first section (4.1) reports the BLEU scores achieved by each of the models when evaluated against each of the test corpora at the end of training. The second section (4.2) gives a comparison of two sentences as translated by each of the three models, in order to provide additional perspicuity which is potentially not present in the BLEU scores when considered alone.

4.1 BLEU Scores

Table 4.1. BLEU Score Earned by Each Model against Each Test Corpus upon Completion of Training

	Newstest2015	Newstest2016
Control	18.84	19.97
Abbreviated STTS	18.88	20.22
STTS	18.91	20.28

More detailed discussion of the implications of the experiment results is provided in the next chapter; however, as a preliminary observation, it can be seen that, as measured by BLEU, the two models trained from a corpus with POS annotations are of marginally higher quality than the baseline, and, of these two best-performing models, the model trained using the larger number of POS categories performed best, although again by a slim margin.

4.2 Comparison of Selected Translations

In this section, two source sentences selected from the newstest2016 corpus are presented, each alongside four translations, one translation produced by each of the three models trained in the experiment, and one human reference translation taken from the English half of the bilingual corpus.

Source:	Nach dem Umbau werden die Räume moderner aussehen.
Control:	The restructuring of the rooms will be more modern.
Abbr. STTS:	After the reconstruction, the rooms will be more modern.
STTS:	After restructuring, the rooms will be modern.
Ref.:	After the conversion the rooms will look more modern.

One syntactic difference between German and English is that German is a V2 language – the finite verb comes in the second position – while English is SVO – the finite verb generally succeeds the subject and precedes the object. In the baseline, control translation, the subject is correctly moved to before the verb; however, as if the model were trying to enforce a V2 ordering in English, the subject is incorporated into a larger noun phrase, the other source for which is a prepositional phrase, whose head should be the finite verb, not the subject. The two translations that were made with access to POS annotations, on the other hand, are more accurate and natural, with the subject being moved before the finite verb, but coming after the leading prepositional phrase, from which it is correctly distinct.

Source:	Danach war der Mann, der sich nach Angaben seines Anwalts mittlerweile wieder auf freiem Fuß befindet, in eine größere Zelle verlegt worden.

Control: After that, the man who, according to his lawyer's information, has now once again been at free, was moved into a larger cell.

Abbr. STTS: After that, the man who is now at liberty, according to his lawyer, has been moved to a larger cell.

STTS: After that, the man, who, according to his lawyer, has now returned to free foot, was moved into a larger cell.

Ref.: Afterwards the man, who according to a statement by his lawyer has since been released, was transferred to a larger cell.

One area were it might be speculated that a model which is trained with access to part-of-speech information could outperform a model trained without such access is the area of relative clauses. In German, relative pronouns are largely homonymic with definite articles, as well as with demonstrative pronouns, so it could be hypothesized that POS information here might be helpful for disambiguation. In many cases, however, such as in the example given above, it is interesting to observe that the control model performs about just as well as the other translation models in rendering the long relative clause in the source into English. It can also be noted that, in this example, the model trained using the full STTS tagset had difficulty translating the idiomatic “auf freiem Fuß” (on the loose), while the control model does a slightly better job on this point.

CHAPTER 5

DISCUSSION

In the preceding chapter, the experiment results are presented with minimal commentary; in the present chapter, a discussion is provided of some of the potential implications of these results. In this chapter's first section (5.1), the main conclusions that may be drawn from the results are presented. In the second section (5.2), a few limitations of the experiment are discussed which have the potential to moderate confidence in the conclusions presented in section 5.1. In the final section (5.3), a recommendation is made for future work.

5.1 Significance

The primary finding of the research, which is shown in Table 4.1, is that the model trained using annotations from an abbreviated STTS performs better against the test corpora, in terms of BLEU, than the baseline, and the model trained using regular STTS annotations likewise performs better than the model trained using the abbreviated set of POS categories. This suggests that for neural machine translation, POS annotations do not represent a redundancy that is learned from raw text; rather, the incorporation of POS tags when training a neural translation model (for German-to-English) leads to increased performance – this is a confirmation of prior research – and furthermore the experiment results suggest, as a new finding, that use of more finely delineated POS categories leads to greater increases in performance.

Table 5.1. Difference between BLEU Scores Earned by Each Model against Each Test Corpus upon Completion of Training

		Control	Abbr. STTS	STTS
Newstest2015	Control	0	−0.0021	−0.0037
	Abbr. STTS	+0.0021	0	−0.0016
	STTS	+0.0037	+0.0016	0
Newstest2016	Control	0	−0.013	−0.015
	Abbr. STTS	+0.012	0	−0.0030
	STTS	+0.015	+0.0030	0

Another implication of the results data that can be commented on concerns the *amount* of useful information that is encoded by the POS annotations, which cannot be learned by the model from raw text. The results reported in Table 4.1, which are further analyzed in Table 5.1, suggest that although there is some information that is encoded by POS labels that cannot be easily learned from raw text, this information may account for only somewhere between 0.37% and 1.5% of the total.

5.2 Limitations

In this section, a few limitations of the research, which need to be kept in mind when considering the experiment's results, are presented. One potential limitation of the research, which was first pointed out in subsection 2.1.2, concerns evaluation methods. Ideally, a comprehensive human evaluation would have been used to compare performance among the three models; however, due to limited resources, BLEU was used instead as the primary metric for performance comparison. Some of the intrinsic limitations of BLEU are discussed near the end of subsection 2.1.2, but one additional item that should have attention drawn to it is that the validation and test corpora each included only one reference translation for each source sentence.

Another limitation of the research concerns the non-trainable network parameters that were used for the models. Due to limited resources, only one set of non-trainable parameters could be tested. Ideally, multiple variations would have been compared. For example, the effect of devoting more, or fewer, of the embedding layer nodes to the POS annotations might have been tested. Thus, there may exist configurations of non-trainable network parameters that provide larger performance gains for POS annotation than those reported here, or there might be a configuration that would allow the control model to perform just as well as the models trained with annotations.

One final potential limitation of the research, to which attention is also drawn in n. 9, concerns the reproducibility of the experiment results given that the free parameters of a neural network are initialized, before training begins, to random values, and that consequently if two networks with equivalent non-trainable parameters are trained on the same data, the free parameters of each network will almost certainly not converge to identical values, thus leading to slightly different performance when comparing the two networks. Because the differences in performance among the three neural-network models trained in the present study are relatively small, we should ask whether the differences are significant. There may be no completely convincing way to check this other than by running additional iterations of training from scratch, and checking whether the new results remain close to what is reported in Table 4.1. Nevertheless, in an important deep learning review article, LeCun et al. (2015) report that for large neural networks, such as those used in the present study, given a particular configuration of non-trainable network parameters, and a particular training dataset, convergence of the network to

models of differing quality as a result of different random initializations of the network's free parameters, in practice, tends not to be a problem.

5.3 Recommendations

The objective of the research conducted under this thesis, as has been reiterated throughout, was to determine the relative performance of two part-of-speech tagsets when used as supplementary input in neural MT. The findings of this research have been reported and discussed above. However, as was alluded in the first paragraph of the introduction to this thesis, the present work, in addition to what it offers in particular, also belongs to a wider discussion concerning the use of linguistically motivated approaches in natural language processing applications, such as MT, in general. In terms of this general question, the present research confirms a common observation: on the one hand, supplementing machine learning approaches with linguistics often leads to measurable improvements, but on the other hand, these improvements are often small.

For the POS tagsets tested in the present research, for newstest2015, tagset size is nearly proportional to performance improvement over baseline, but for newstest2016, the improvement added by full STTS over the abbreviated STTS tagset is not as large relative to the performance improvement of the abbreviated STTS tagset over baseline. Potential future work is to test a larger tagset, and use more, or larger, test corpora. This will clear up the question of how quickly the performance increases of adding more tag categories begin to plateau.

REFERENCES

[Bahdanau et al., 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *Proceedings of the International Conference on Learning Representation (ICLR)*.

[Barrault et al., 2018] Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliot, and Stella Frank. 2018. Findings of the Third Shared Task on Multimodal Machine Translation. *Proceedings of the Third Conference on Machine Translation*, pp. 304-323.

[Barrault et al., 2019] Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 Conference on Machine Translation (WMT19). *Proceedings of the Fourth Conference on Machine Translation*, vol. 2, pp. 1-61.

[Bojar et al., 2016] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. *Proceedings of the First Conference on Machine Translation*, vol 2, pp. 131-198.

[Bojar et al., 2017] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). *Proceedings of the Second Conference on Machine Translation*, pp. 169-214.

[Brill, 1992] Eric Brill. 1992. A Simple Rule-Based Part of Speech Tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, pp. 152-155.

[Brill, 1995] Eric Brill. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, vol. 21, no. 4, pp. 543-565.

[Brown et al., 1990] Peter F. Brown, John Cocke, Stephan A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, vol. 16, no. 2, pp. 79-85.

[Brown et al., 1991] Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. Aligning Sentences in Parallel Corpora. 1991. *29th Annual Meeting of the Association for Computational Linguistics*, pp. 169-176.

[Brown et al., 1993] Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, vol. 19, no. 2, pp. 263-311.

[Brown et al., 1995] Peter F. Brown, John Cocke, Stephan A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, Jennifer C. Lai, and Robert L. Mercer. 1995. Method and System for Natural Language Translation. U.S. Patent No. 5,477,451.

[Cho et al., 2014a] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Behdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*.

[Cho et al., 2014b] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103-111.

[Chollet, 2018] François Chollet. 2018. *Deep Learning with Python*. Manning.

[Church, 1988] Kenneth Ward Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Proceedings of the Conference on Applied Natural Language Processing (ANLP)*, pp. 136-143.

[Thrax, 1874] Dionysios Thrax, translated by Thomas Davidson. 1874. The Grammar of Dionysios Thrax. *The Journal of Speculative Philosophy*. Accessed at https://en.wikisource.org/wiki/The_grammar_of_Dionysios_Thrax on 19 Jul. 2020.

[Doddington, 2002] George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. *Proceedings of the Second International Conference on Human Language Technology Research*, pp. 138-145.

[Gale and Church, 1993] William A. Gale and Kenneth W. Church. 1993. A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics*, vol. 19, no. 1, pp. 75-102.

[Jurafsky and Martin, 2009] Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed., Pearson.

[Kalchbrenner and Blunsom, 2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700-1709.

[Kay and Röscheisen, 1993] Martin Kay and Martin Röscheisen. 1993. Text-Translation Alignment. *Computational Linguistics*, vol. 19, no. 1, pp. 121-142.

[Koehn et al., 2003] Phillip Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL '03)*, vol. 1, pp. 48-54.

[LeCun et al., 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature*, vol. 521, pp. 436-444.

[Nguyen et al. 2017] Dat Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing. *Proceedings of the 2017 Conference on Computational Natural Language Learning (CoNLL)*, pp. 134-142.

[Nguyen and Verspoor, 2018] Dat Quoc Nguyen and Karin Verspoor. 2018. An Improved Neural Network Model for Joint POS Tagging and Dependency Parsing. *Proceedings of the 2018 Conference on Computational Natural Language Learning (CoNLL)*, pp. 81-91.

[Papineni et al., 2002a] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 311-318.

[Papineni et al., 2002b] Kishore Papineni, Salim Roukos, Todd Ward, John Henderson, and Florence Reeder. 2002. Corpus-based Comprehensive and Diagnostic MT Evaluation: Initial Arabic, Chinese, French, and Spanish Results. *Proceedings of Human Language Technology 2002*, pp. 132-137.

[Popović, 2015] Maja Popović. 2015. CHRF: Character *N*-Gram F-Score for Automatic MT Evaluation. *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pp. 392-395.

[Schiller et al., 1995] A. Schiller, S. Teufel, and C. Thielen. 1995. Guidelines für das Tagging deutscher Textcorpora. Technical Report, University of Stuttgart and University of Tübingen.

[Schiller et al., 1999] Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. Guidelines fur das Tagging deutscher Textcorpora mit STTS. Technical Report, University of Stuttgart and University of Tübingen.

[Schmid, 1994] Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*.

[Schmid, 1995] Helmut Schmid. 1995. Improvements in Part-of-Speech Tagging with an Application to German. *Proceedings of the Association for Computational Linguistics (ACL) SIGDAT-Workshop*.

[Senellart et al., 2001] Jean Senellart, Péter Dienes, and Tamás Váradi. 2001. New Generation Systran Translation System. *Machine Translation Summit VIII*.

[Sennrich and Haddow, 2016] Rico Sennrich and Barry Haddow. Linguistic Input Features Improve Neural Machine Translation. 2016. *Proceedings of the First Conference on Machine Translation*, vol. 1.

[Specia et al., 2018] Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón F. Astudillo, and André Martins. 2018. Findings of the WMT 2018 Shared Task on Quality Estimation. *Proceedings of the Third Conference on Machine Translation*, pp. 689-709.

[Sutskever et al., 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. 2014. *Advances in Neural Information Processing Systems (NIPS)*.

[Vaswani et al., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmer, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems (NIPS)*.

[Vauquois, 1968] Bernard Vauquois. 1968. A Survey of Formal Grammars and Algorithms for Recognition and Transformation in Mechanical Translation. *IFIP Congress*, vol. 2, pp. 1114-1122.

[Zhang et al., 2004] Ying Zhang, Stephen Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST Scores: How Much Improvement do We Need to Have a Better System?. *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*.

[Zinsmeister et al., 2014] Heike Zinmeister, Ulrich Heid, and Kathrin Beck. 2014. Adapting a Part-of-Speech Tagset to Non-standard Text: The Case of STTS. Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), pp. 4097-4104.