Towards High-quality Neural Machine Translation for Polysynthetic Languages

Topic: Computational Linguistics

Brennan Dell

MA-level Qualifying Paper
Advisor: Lane Schwartz

Department of Linguistics
University of Illinois at Urbana-Champaign
United States
January, 2020

Abstract

The application of neural networks to the problem of machine translation (MT) has enabled the development of state-of-the-art models for many language pairs. However, neural machine translation (NMT) models have been shown to perform poorly when large amounts of training data are not available. Most of the world's languages (which include most indigenous languages), have little training data available, which can make the development of high-quality NMT systems for these languages difficult. This study focused on the problem of creating translation systems for English and the St. Lawrence Island Yupik (Yupik) language of western Alaska, a language with limited training data, and a complex polysynthetic morphology. This study evaluated two approaches to improve the quality of low resource NMT systems: using a multilingual model with large amounts of auxiliary data (Inuktitut), and optimizing the hyperparameters of NMT architectures specifically for low-resource settings. This study confirmed that each of these approaches is able to improve translation quality by about 3 BLEU. Nonetheless, the translation quality of the models did not exceed 7.48 BLEU, and were not able to significantly outperform statistical machine translation (SMT) baselines. Therefore, these approaches alone are not sufficient to produce high-quality translations for Yupik, and further research should be devoted to this problem. Research which leads to the development of a successful Yupik-English translation system may also apply to translation systems for other low-resource and polysynthetic languages. Therefore, such research is significant, as it would greatly expand the number of languages for which high-quality NMT is possible.

Contents

Reading Guide

This paper was written for an audience familiar with linguistics, but may not be familiar with concepts of machine learning, neural networks, or machine translation. Chapter 2 attempts to explain these topics briefly, but with enough detail to explain the factors which are important for this project.

Readers who wish to skip chapter 2 should be familiar with the following concepts:

- Train, validation, and test sets
- Overfitting
- Feed-forward Neural Networks
- Hyperparameters
- Dropout (in neural networks)
- Grid search
- Recurrent neural networks
- Byte-pair encodings
- BLEU scores and alternative translation metrics

Chapter 1 discusses the motivations for the development of this project, and the aspects of the St. Lawrence Island Yupik language, along with the challenges it poses for neural machine translation.

Chapter 2 discusses the fundamentals of machine learning, neural networks, neural machine translation, and automatic translation evaluation metrics to a reader who many no be familiar with those topics. Readers who are familiar with these topics may skip this chapter.

Chapter 3 describes the challenges of producing neural machine translation systems with limited training data available, and reviews the literature on proposed solutions to this problem. Chapter 3 also lists the experiments carried out in this study, and presents the results from these experiments.

Chapter 4 discusses the results, and attempts to explain them. This chapter concludes the study, addresses the hypotheses made in chapter 3, and describes directions for future research.

Chapter 1

Introduction

This chapter discusses the motivation for this project, and aspects of the St. Lawrence Island Yupik language. It also discusses at a high level the main findings of this project.

1.1 Motivations

There are two main goals to this project. The first is to develop a machine translation system for the languages of English and St. Lawrence Island Yupik (abbreviated as "Yupik" in this paper). Techniques which enable the development of a high-quality translation system for these two languages should in principle be useful to develop high-quality systems for other low-resource or polysynthetic languages. This research aims to determine what these techniques are, so they may be applied to expand the reach of machine translation to more of the world's languages.

The second goal is to evaluate existing methods developed to address the problem of low-resource machine translation, namely multilingual modelling, and hyperparameter optimization. These approaches were developed for high-resource languages, such as English, French, and German, all of which have much more training resources available compared to Yupik, and which are morphologically much more isolating than agglutinating. This work aims to determine if these methods will increase translation quality for a polysynthetic languages as well as the European languages for which they were developed.

Like many of the indigenous languages of North America, Yupik is an endangered language. It is spoken by a population of around 1,200, however very few of the Yupik people born after 1980 are able to speak the language (Schwartz, 2019). Yupik's status as an endangered language has resulted from the increasing presence of monolingual English speakers in St. Lawrence Island, education policies which focused on assimilation or even outright prohibition of native languages, and a lack of educational materials to develop school curricula in Yupik.

The language situation of Yupik is unfortunately quite common; UNESCO estimates that roughly 40% of the 6,700 languages spoken today are endangered . Languages which are disproportionately threatened are languages with less than 10,000 speakers, and indigenous languages. In recognition of this situation, UNESCO declared 2019 (the year of this writing) the International Year of Indigenous Languages (UNESCO, 2019).

Through IYIL, UNESCO recognizes the significance of global linguistic diversity, both for linguistic communities and academic linguistics. Many unique linguistic aspects, such as polysynthetic morphology, are common among endangered languages, and the loss of such languages would be harmful for linguistic inquiry. More importantly, the loss of a language is the loss of a unique cultural perspective, and the loss of an example of human genius and creativity (Grenoble and Whaley, 1996).

The development of natural language processing (NLP) technologies, while proving to be very useful, are strongly focused on high-resource languages; namely languages for which there is a large amount of digital language data. High-resource languages include many of the languages of Europe, as well as Chinese, Japanese, Korean. However, the datasets which modern NLP technologies have come to rely on are very small or non-existent for most of the world's languages. Because of this, a digital divide has developed between these high-resource languages and indigenous languages; providing advantages such as employment and social capital for individuals who are able to use a high-resource language (King, 2015).

NLP systems can prove particularly useful in the maintenance of a language through the development of pedagogical materials. As Iutzi-Mitchell and Graburn (1993) note, the development of a Kalaallisut[1]-only curriculum for school children grades four and under in Greenland has produced a large population of Kalaallisut-speaking children, counteracting the Danization policies from the 1950's. However, they also note that the development of this curriculum suffers from a lack of trained professionals to develop and teach the materials. These problems may be alleviated through the application of machine translation. For example, the application of post-editing, i.e. the process of having a human correct the output of an MT system, has been shown to significantly decrease the time of translation, and even improve translation accuracy (Green et al., 2013). If a high quality MT system is used, such a system would greatly reduce the time and effort needed to produce pedagogical materials. Therefore, a high-quality MT system for low-resource languages could potentially enable the development of a curriculum which can counteract the processes of language loss, such as that of Yupik.

Although the development of a high-quality translation system for Yupik and English could be beneficial to the natives of St. Lawrence Island, we acknowledge that technological solutions alone cannot sustain a language community. As Grenoble and Whaley (1996) note, the most important aspect in language revitalization is a revival movement led by the language community itself. Ad-

[1] Also known as West Greenlandic

ditionally, it is important that the Yupik language community develops an ideological conception of itself which can sustain its linguistic practices, and that harmful external ideological pressures from American society and institutions are recognized and countered (Dorian, 1998). However, technological support for such language communities has also been recognized as an important aspect of language revitalization (Dauenhauer and Dauenhauer, 1998). We therefore aim to provide technical support for the Yupik community through this project. Additionally, we hope that our work will encourage the wider NLP community to develop approaches which can handle a broader range of languages which may differ typologically, or in levels of available resources.

1.2 St. Lawrence Island Yupik

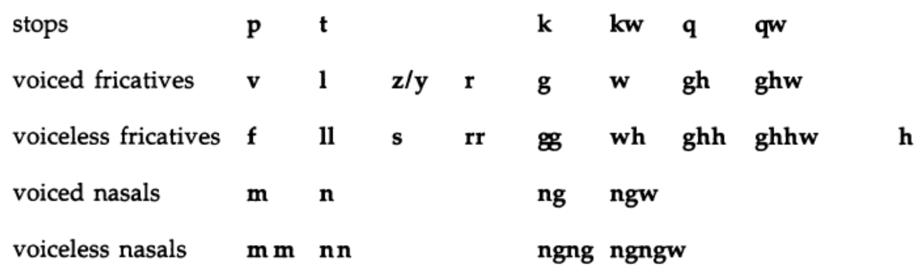
St. Lawrence Island Yupik ("Yupik" in this paper, ISO 639-3: ess), also known as Central Siberian Yupik, or CSY, is a member of the Yupik branch of the Eskimo-Aleut language family. It is spoken on St. Lawrence Island in the Bering Strait, and on the Chukotka Penninsula in eastern Siberia (Schwartz and Chen, 2017). St. Lawrence Island Yupik is closely related to the other Yupik languages the largest of which, Central Alaskan Yupik, is spoken on the western Alaskan mainland (Miyaoka, 2012). Yupik and the other Yupik languages are more distantly related (but nonetheless very similar to) the Inuit languages, which include Inupiaq on Alaska's North Slope, Inuktitut and its many dialects in northern Canada, and the Greenlandic languages (Fortescue et al., 1994). For the purposes of this paper, St. Lawrence Island Yupik will be referred to simply as Yupik.

Languages of the Eskimo-Aleut family are (in)famous for their complex morphology, where single words can represent the meaning of an entire sentence in other languages.

1.2.1 Phonology and Orthography

The modern American orthography for Yupik was developed by linguist Michael Krauss of the Alaska Native Language Center, who collaborated with St. Lawrence Islanders and David Shinen from Wycliffe Bible Translators (Jacobson, 2001). The orthography used on the Russian side is based off of the Cyrillic alphabet.

All of the stop consonants in Yupik are voiceless, while the fricatives and nasal consonants may be voiced or voiceless. Voiceless consonants are commonly represented by doubling its voiced counterpart (e.g. "g" = [ɣ] and "gg" = [x]). Velar and uvular consonants may be labialized, which is indicated by a "w" following the labialized letter or letters. Therefore, a single phoneme in Yupik may be represented with one to five letters: the extreme is "ngngw", which represents the voiceless labialized velar nasal [ŋ̊^{w}]. We will call a sequence of letters a "grapheme" if it represents a single phoneme. In the case that two separate sounds occur in a Yupik word which could form a single grapheme, such as "n" [n] and "g" [ɣ], they are separated by an apostrophe. Therefore the



stops	**p**	**t**			**k**	**kw**	**q**	**qw**	
voiced fricatives	**v**	**l**	**z/y**	**r**	**g**	**w**	**gh**	**ghw**	
voiceless fricatives	**f**	**ll**	**s**	**rr**	**gg**	**wh**	**ghh**	**ghhw**	**h**
voiced nasals	**m**	**n**			**ng**	**ngw**			
voiceless nasals	**m m**	**n n**			**ngng**	**ngngw**			

Figure 1.1: The consonant graphemes of Yupik (Jacobson, 2001)

word "an'gighaquq" (he is going out for fresh air) is pronounced as [anɣiχaquq] rather than [aŋiχaquq].

There are four vowels in Yupik: "a" ([ɑ]), "i" ([i]), "u" ([u]) and "e" ([ə]). The vowels "a", "i", and "u" may be phonemically long in Yupik words, in which case they are doubled. Diphthongs are not permitted by the phonology of Yupik.

Vowel length in Yupik may be further modified by the language's prosody, causing phonemes which are underlyingly short to be pronounced phonetically long, or vice versa (Krauss, 1975).

Despite the phonetic transparency of the Yupik orthography, texts written in Yupik frequently contain spelling variations (Schwartz, 2019). NLP systems are commonly not equipped to handle spelling errors, therefore this variation in the orthography posed a challenge to developing an MT system for Yupik.

1.2.2 Morphology

Yupik is a polysynthetic language; morphemes in Yupik can be combined to form words which express the meaning of an entire phrase or sentence in English. With the exception of a single demonstrative prefix, Yupik adds its morphemes exclusively as suffixes.

Words are formed starting with a nominal or verbal root, called a "base". These bases may be followed by one or more derivational suffixes[2], which add meaning to the base, and may change the word's part of speech. A word in Yupik must end with an inflectional ending. Inflectional endings on nominal bases may be used to mark grammatical case or possession. Inflectional endings on verb bases indicate the person and number of the subject in an intransitive verb, or the person of both the subject and the object in an intransitive verb. An enclitic may optionally be added after the inflectional ending.

[2]called "postbases" by Jacobson (2001)

base (+ one or more postbases) + inflectional ending (+ enclitic)

Figure 1.2: The structure of a word in Yupik (components in parentheses are optional)

Concatenating morphemes in Yupik can trigger a series of morphophonological changes. These changes are illustrated in the figure below, demonstrating how a Yupik word may be produced. This word could be translated into English as “I am going to put crowberries in it”. It is built starting with the base “pagunghagh-” meaning crowberry.

Yupik Word	**English Translation**
pagunghagh-	crowberry
pagunghaligh-	to put crowberries in (something)
pagunghalighnaqe-	to be going to put crowberries in (something)
pagunghalighnaqa-	to be going to put crowberries in (something, transitive, indicative mood)
pagunghalighnaqaqa	I am going to put crowberries in it

Figure 1.3: The derivation of a word in Yupik (Chen and Schwartz, 2018)

Note that there is a single morpheme in Yupik which makes the verb transitive and in the indicative mood, and a single morpheme indicating the subject and the object.

1.2.3 Syntax

As mentioned in the morphology section, Yupik nouns are inflected for case. There are seven cases in Yupik, namely absolutive, relative (or ergative), ablative-modalis, vialis, equalis, terminalis, and localis. These inflectional endings are used to indicate the semantic roles of the nouns in the sentence. Yupik is an ergative-absolutive language, meaning that the absolutive case is used to mark both the subject of an intransitive verb, as well as the object of a transitive verb.

Verbs in Yupik may be polypersonal, indicating both the subject and the object in its inflection. Inflectional endings in Yupik usually render pronouns unnecessary except for emphasis (Jacobson, 1977).

The high amount of information conveyed by these inflectional endings allow Yupik (and the other languages of the Eskimo-Aleut family) to exhibit a very free word order. However, the most common word order of Yupik is SOV (Fortescue, 1993).

1.2.4 Language Status

Yupik spoken by around 1,200 individuals in the United States (Vakhtin, 1998), however many of the Yupik people born after the 1980’s are not able to speak

Yupik (Schwartz, 2019). The endangerment of Yupik is due to in large part to the earlier assimilationist education policies in Alaska, which only allowed English as the legal language of instruction until 1970. Miyaoka describes the effects of these policies:

> *"The older Native people still retain vivid but gloomy memories of those old days when they had to go without lunch, had their mouths taped shut, or received even more severe corporal punishments, just because they uttered a single native word in school. Some parents, perhaps fearing reprisals from teachers or feeling that knowledge of Yupik would somehow interfere with their children's ability to absorb English, which they saw as essential to the future success of the younger generation, ceased to speak their own language even at home. Until the late 1960s it was actually illegal to teach in any language other than English."* (Miyaoka, 2012, p.7).

These experiences of Alaskan Natives as described by Miyaoka demonstrate the damage that such policies cause to native communities, and the languages which these communities speak.

After the assimilationist policies were discontinued in 1970, bilingual education programs were developed in Alaskan schools. However, these bilingual policies were intended less to revitalize of the native Alaskan languages, and instead aimed to transition native children to an English language curriculum more easily. This policy had the somewhat paradoxical effect of further endangering the native Alaskan languages, as it facilitated the process of the language shift to English, instead of reinforcing the status of the native languages (Miyaoka, 2012, p.8).

These reasons are the likely explanations of the lack of knowledge of Yupik among the younger generation of St. Lawrence Islanders. Today, education programs suffer from the lack of resources for native students to learn their language, and for English speaking instructors to native languages (Schwartz, 2019). The application a post-editing system as described by Green et al. (2013) with a high-quality MT system for Yupik and English may reduce the labor needed to produce these materials. If such curricula are developed, they may even counteract the effects of a colonial language, as was the case in Greenland (Iutzi-Mitchell and Graburn, 1993).

Given these potential benefits of an MT system, the goal of this project was to produce a system with the highest possible quality.

1.2.5 The Challenges of Yupik for NLP

Many of the aspects mentioned about Yupik pose challenges for NLP systems, including machine translation systems. These aspects will be enumerated in this section.

The first difficulty of Yupik stems from its highly productive morphology. Many NLP applications use a closed vocabulary, which is not ideal for a language which can productively produce a large number of words via derivational suffixes. Although derivational suffixes are common in many languages including English, derivational suffixes are used much more frequently in Yupik, and therefore must be accounted for in an NLP application. Additionally, an NLP system for Yupik must also be able to handle morphophonological alternations, instead of simply splitting words into sequences of characters.

The second difficulty comes from irregularities in the spelling. There is a high degree of variation in spelling between Yupik individuals (Schwartz, 2019). NLP systems perform poorly in such situations, as they often consider two words to be identical only if their characters match exactly. Even a small difference in spelling will cause words to be unrecognizable for most NLP systems.

The third difficulty is the presence of ergativity in the language. As the subject of an intransitive verb and the object of a transitive verb are marked identically in Yupik, it may be a problem for an NLP system to correctly identify the semantic role of the entities referenced in a sentence. Most NLP systems focus on the languages of Europe and East Asia, which are mainly nominative-accusative languages. Therefore, ergativity is a feature which is often not handled by NLP systems. It may therefore be difficult for a machine translation system to properly translate Yupik text in such a way that preserves the semantic roles of the nouns in the sentences.

The fourth difficulty is the free word order of Yupik. NLP systems which make use of machine learning have difficulty generating sentences when there is not a clear pattern in the sentence structure. Therefore, if there are frequent variations in the word order, an NLP system such as a translation system may not be able to learn a proper word order, or the default SOV word order of Yupik.

Finally, a major difficulty of Yupik is the lack of data available to use as training data for NLP systems. Machine translation systems based on machine learning systems usually require a bilingual corpus as training data. For Yupik, the only available source of this data is the Wycliffe New Testament. This corpus contains roughly 8,000 sentences, but NMT systems usually require hundreds of thousands or even millions of sentences to produce decent translations (Koehn and Knowles, 2017). Producing larger amounts of bilingual data is difficult as well, as the population of Yupik speakers is very small, and there is difficult to find translators to produce bilingual corpora. Therefore, an NMT system for Yupik must be adapted to the paucity of the data.

The difficulties mentioned thus far demonstrate that applying NLP systems adapted for high-resource languages will not be ideal for developing a translation system for Yupik. This is because of Yupik's orthography, morphology and syntax, as well as the small amount of text suitable for training data. This study will evaluate approaches to NMT which have been developed to solve these problems.

1.3 Overview of Experiments and Results

In this project, we attempted to produce a neural machine translation system which could produce translations from Yupik to English and English to Yupik with high quality. Neural machine translation (NMT) systems are the current state of the art in machine translation, however they have been found to perform poorly in comparison to statistical machine translation systems (SMT, the previous dominant paradigm in machine translation) when training data is scarce (Koehn and Knowles, 2017). This project investigates two methods of improving the quality of an NMT system in low-resource settings. We implemented these two methods to produce Yupik translation systems, and compared them to NMT systems which did not use these optimizations, and to an SMT systems.

The first approach to low-resource NMT is to use a multilingual system, that is, to train a single NMT model with multiple language pairs. Multilingual NMT systems have been found to outperform NMT models trained only on bilingual data (Johnson et al., 2017; Lakew et al., 2018). This effect is particularly strong when the languages used by the multilingual model are related (Imankulova et al., 2019; Zoph et al., 2016; Gu et al., 2018). Therefore, we produced a multilingual NMT model, using a large multilingual corpus of Inuktitut-English data from the Nunavut parliamentary records (Micher, 2018), along with a corpus of Yupik-English data from the Yupik translation of the New Testament (Yupik Translation Committee, Wycliffe Bible Translators, Inc., 2012). We aimed to use the large amounts of auxiliary data to counteract the limited amounts of training data we had for Yupik.

The second approach we used to improve our NMT system's quality is to adjust the structure and components of the neural network in a manner that is appropriate to a low-resource translation setting. This process is known as hyperparameter optimization, a key component of neural network research. Sennrich and Zhang (2019) found that applying the appropriate hyperparameter optimizations to NMT systems allow it to outperform SMT systems in low-resource settings, contradicting the findings of Koehn and Knowles (2017). In our project, we applied the techniques and hyperparameters developed by Sennrich and Zhang to an NMT system for Yupik, to determine if their findings could be reproduced on a polysynthetic language.

We found that both approaches toward low-resource NMT optimization were able to produce translation models which outperformed NMT systems trained solely on Yupik-English data and with the default hyperparameters. Each approach saw improvements by about 3 BLEU for both the Yupik$\rightarrow$English and English$\rightarrow$Yupik translation directions. However, the best performing models were not able to achieve a BLEU score greater than 7.5 BLEU, therefore the goal of creating a high-quality Yupik translation system was not satisfied. Additionally, the models we trained were not able to significantly outperform the SMT baselines, as Sennrich and Zhang were able to do.

Therefore, we found that the NMT optimizations as we applied them are not able to produce an NMT system which is of high quality, or which can outperform an SMT model. Future research should aim to solve these problems, and there remain many techniques for achieving this. In particular, augmenting the training data using monolingual data in a process called backtranslation (Sennrich et al., 2016a) could be useful, because there are much larger monolingual resource in Yupik than there are bilingual resources. Additionally, a speech-to-text translation system may also be useful, as was found by Bansal et al. (2019), who made an Mboshi-French translation system using limited amounts of speech data. Finally, more emphasis in future research could be dedicated to SMT systems, which this project suggests are better at producing Yupik translations than NMT approaches. Future research will investigate these approaches, with the aim of producing translation systems for more of the worlds languages, and providing the benefits of NLP to more of the world's peoples.

Chapter 2

Foundational Background

The purpose of this chapter is to describe the core concepts of this research project to an individual who is familiar with linguistics, but may not be familiar with topics in natural language processing. This chapter will discuss fundamental concepts of machine learning, neural networks, machine translation, and evaluation metrics. If the reader is already familiar with the topics below, they may skip this chapter.

- Train, validation, and test sets
- Overfitting
- Feed-forward Neural Networks
- Hyperparameters
- Dropout (in neural networks)
- Grid search
- Recurrent neural networks
- Byte-pair encodings
- BLEU scores and alternative translation metrics

2.1 Fundamentals of Machine Learning

This section discusses the fundamental concepts of machine learning which NMT relies on.

2.1.1 Machine Learning

NLP applications can be broadly divided into two approaches: rule-based approaches, which apply knowledge from linguistic inquiry to improve an application, and machine learning, which learns linguistic patterns from the data with little to no human input.

Rule Based Approaches

A rule-based NLP application is a program is written to encode linguistic knowledge. Such an example would be the work of Chen and Schwartz (2018), who converted the analysis of Yupik morphophonology developed by Jacobson (2001) into a finite-state transducer using the *foma* framework (Hulden, 2009) which can segment Yupik words into its constituent morphemes. The CMUDict (Weide, 2014) could also be considered an example of a very simple rule based system to convert English words into phoneme sequences. CMUDict achieves phonetic transcription by pairing 134,000+ words with a phonetic transcription in English. The early approaches to machine translation were also rule-based systems. The first publicly demonstrated MT system translated Russian sentences into English by essentially replacing each Russian word with its translation in English, and applying six syntactic rules to convert the output into a valid sentence of English (Dostert, 1955; Schwartz, 2018).

Rule-based approaches are beneficial, because they can typically be converted into a program in a fairly straightforward manner. Additionally, rule-based systems will process all linguistic data consistently according to the rules they were designed with, which are typically more linguistically accurate. However, rule-based systems are difficult to develop; they often require a thorough manual linguistic analysis. Occasionally, the rules developed by a human linguistic may be incorrect. It could be that the linguist made a mistake, or that the linguist simply did not consider enough linguistic data to make a set of rules which can characterize the entire language.

Machine Learning Approaches

The second general approach to natural language processing is to make use of machine learning. In general, all machine learning programs take a dataset as input which consists of a set of problems and their solutions. In the context of machine translation, this dataset could consist of a set of sentences in a source language, each of which is paired with its translation in a target language. Machine learning systems are developed to look for correspondences between the problems and the solutions, with the hope that the correspondences it "learns" will allow it to solve new examples of problems which it has not encountered.

There are two general ways in which machine learning systems "learn" patterns in its training data. The first is to use statistical methods to calculate or estimate which combinations of inputs most likely correspond with certain outputs. The second is to use a neural network, a mathematical function which "learns" patterns by gradually adjusting its parameters to fit training data. The functionality of neural networks will be described later.

A major challenge which machine learning systems face is that their quality depends heavily on the quality of the training data. For a well-functioning system, the training dataset must be large in size, such that most or all of the phenomena of the particular task is represented. For instance in the context of machine translation, the training data must be large enough such that most of the language's vocabulary occurs in it. We cannot expect that a machine translation will be able to learn a pattern if the pattern is very rare, or even does not occur at all in the training data.

Along with size, the domain of training data is also very important. The term "domain" in machine learning refers to the type of data which the training data consists of. In NLP, the domain of a training set could be parliamentary proceedings, TV subtitles, literature, news, academic publications, etc. Each of these domains may use different vocabularies and linguistic registers. Therefore, we can expect a machine learning system trained solely on a single domain to perform poorly on out-of-domain data. It is therefore crucial in a machine learning task to understand the limitations of the training data, and when possible, adapt a machine learning system to handle data from a variety of domains.

Most of the languages in the world are low-resource languages, which means that these languages have little, or even no data available to use as training data for NLP systems. When the training data for a language is limited, the domains of the data are often limited as well. In the case of Yupik, the only example of a bitext, that is, a text with a sentence-for-sentence translation into English, is the New Testament produced by Wycliffe Bible Translators. Translation systems trained using only the Bible as training data will likely be biased towards language common in biblical stories and religious terminology, and may not extend well to other domains. The aim of this project is to apply the state-of-the-art methods to handle the challenges to machine learning from small and domain-specific training datasets.

Once a machine learning system has been trained, it is important to evaluate its quality. We can measure the quality of a system by presenting it with a series of inputs, and calculating the percentages for which a system can produce the correct output. If this accuracy is measured on the data which the system was trained on, it is likely that the accuracy will be artificially high. Performing this kind of evaluation could not determine if the system correctly found underlying patterns in the data, or if it simply memorized examples from the training set. Therefore, a machine learning system is evaluated on a dataset called a test set: a set of data which is held out from the training process and is only used to evaluate the final performance of a machine learning system. Typically, the training set is larger than the test set, so the system is able to use as much data as it can to learn patterns in the data. The test set is also selected such that it is large enough to evaluate a wide range of the systems possible outputs.

Another issue machine learning systems have to learn general patterns in the training data without focusing too closely on the particular examples in the training data. If a system tends to focus too closely on particular data elements, or simply memorizes data from the training set, we say that this system is overfitting the data. It is critical in machine learning to use the appropriate techniques to train a system such that it adjusts according to the general patterns in the training data it encounters, and avoids simple memorization.

One way to prevent overfitting is to use another held-out dataset called a validation set (also called development set). Periodically during the training process, a machine learning system is evaluated based on its performance on the validation set. At the beginning of the training process, the validation score should be very low, as the system has not been able to adjust to its training data yet. As more training data is presented to the system, the validation score should increase, hopefully indicating that the system is learning patterns in the training set. However, if the training process runs too long, it will start to overfit to the training data, that is, it will rely too much on particular training examples rather than general patterns. A system which overfits its training data will have a low validation score, as it will be unable to produce correct outputs for data which do not closely resemble its training data. In summary, as the training process is carried out, the validation accuracy should increase to a maximum, after which it gradually decreases. We can then the system at the point where its validation accuracy is highest as our optimal model.

2.1.2 Precision and Recall

There are multiple metrics which are used to measure the quality of a machine learning system, based on its ability to produce correct outputs for its test set. Two of the most important metrics are called "precision" and "recall".

Precision is calculated by dividing the amount of the system's correct outputs by the total number of the system's outputs. Precision is used to indicate how reliable a machine translation system's output can be. Precision is an important metric used in the BLEU score, an automatic machine translation metric, which will be described later.

The recall metric is calculated by dividing the number of the system's correct outputs by the total number of correct data elements the system could have produced. The recall metric is used to indicate how much of the correct answers the system is able to produce.

The importance of precision and recall varies depending on the task. If it is more important that the system's outputs are reliable, such as a weather forecast for a satellite launch, then high precision would be more important. Models with high precision should be used when false positives should be avoided. If a model is used to make a disease diagnosis, it would be more important that all individuals who have the disease are diagnosed than it would be for each diagnosis to be correct. Models with high recall are therefore less concerned with the quality of each result, so long as the amount of false negatives is low.

2.2 Neural Networks Fundamentals

Neural networks, (sometimes called Artificial Neural Networks to distinguish their biological counterpart), are mathematical functions applied in the fields of machine learning and artificial intelligence. They are called "neural" networks, because they were inspired by the function of biological neurons.

The fundamentals concepts of neural networks were developed in the early days of computing by McCulloch and Pitts (1943) and Rosenblatt (1957). However, the lack of available computing power did not make neural networks useful until recently, when the hardware necessary to produce the amount of calculations needed to train a neural network had become more readily available.

The research area of neural networks has become a much more active area of research in recent years. In 2012, a neural network system called AlexNet (Krizhevsky et al., 2017) was entered into an image classification competition, and outperformed its competitors by a wide margin. This advancement received a great deal of attention, and is considered to be the point at which the current wave of neural network research began.

Research in neural networks increased substantially after 2012, and neural network solutions were developed for a broad range of tasks with significant gains in performance. Neural networks have been developed which can detect faces with 99.63% accuracy (Schroff et al., 2015), perform speech recognition with quality significantly exceeding prevous paradigms (Hinton et al., 2012), and produce state-of-the-art machine translation systems (Bahdanau et al., 2014; Vaswani et al., 2017). The increases in translation quality provided by neural networks have been so large that the current field of machine translation focuses almost exclusively on neural machine translation. The dominance of NMT over all other translation paradigms developed very recently (2016), in an event which Philipp Koehn calls "The Neural Turn" (Koehn, 2020, p.57-58).

Although significant advancements have been made because of neural networks, there is still much research to be done on neural networks. Little research has been done thus far on how a neural network actually learns patterns in its training data. Additionally, the field of Neural Machine Translation has focused primarily on the translation of higher resource languages, and has neglected languages like Yupik. Thereefore, the aim of this project was to attempt to extend the benefits of NMT to Yupik translation if possible, and to determine which aspects of NMT need to be improved so it can handle the translation of low-resource polysynthetic languages.

2.2.1 How Neural Networks Work

Biological neural networks consist of millions of cells called neurons, which send electrical impulses to each other. Each neuron receives these electrical impulses from other neurons via structures called dendrites. If a neuron receives enough input from other neurons through its dendrites, the neuron will send an electrical impulse out along its axon, which connects to the dendrites of other neurons. The connections between neurons in a biological network can change, causing the network to adjust its outputs according to the inputs it receives. Because of this, a biological neural network is capable of learning.

Artificial Neural Networks are mathematical functions which can "learn" patterns in their input data in a manner similar to that of biological networks. As the name implies, an artificial neural network is a network of smaller functions, which are called nodes, which are analogous to biological neurons. Instead of outputting electrical pulses, these artificial neurons, called nodes, output a real number, which could be any value between positive and negative infinity. The output of a node is calculated by combining inputs from other nodes in the network. The inputs from other nodes are multiplied by weights, which have the effect of forcing the node to pay more attention to the outputs from some nodes which it is connected to more than others.

The function describing a node in a neural network can be written like this:

$$o = f(\sum_k w_k i_k + b)$$

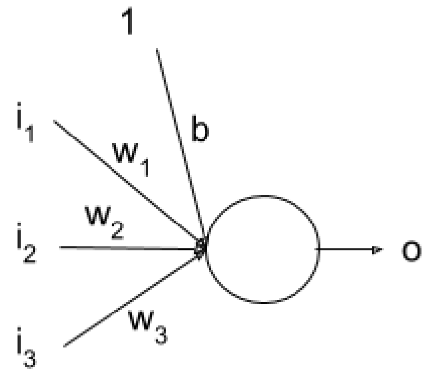


Figure 2.1: A node in a neural network

Where i_k is the input from the k^{th} node which this node is connected to, w_k is the weight for the input from the k^{th} node, b is a bias term, and f is a non-linear function. Each of these components in the node's function will be described below.

If it the case that each of the input numbers to a node are zero, or are close to zero, then its output must also be close to zero, even when very large values are used for the weights. To alleviate this problem, a special bias term is added to the equation as shown above. This bias term is modified during the training process just as the weights are. The bias term could also be conceptualized as an additional weight to a node's input, where that additional input is always 1 (this is why there is an input labelled "1" above). Even if all the inputs to a node are zero, the bias term enables the node to output a non-zero value, which adds more flexibility to the pattern of outputs the network is capable of producing. The collection of weights and biases form what are called the "parameters" of a neural network[1].

Once the weights have been multiplied to each of the node's inputs, and a bias term has been added, this sum is put through a non-linear function. Essentially, a non-linear function is any function which, when plotted on a graph, does not appear as a single straight line. Rather, it may appear as a bent line, or as a curved line. Two commonly used non-linear functions (sigmoid and ReLU),

[1]not to be confused with hyperparameters, which will be explained later

are plotted below. The non-linear function adds a further level of complexity to the node, which drastically increases the number of functions which a neural network can represent. The reasons for this will be argued shortly.

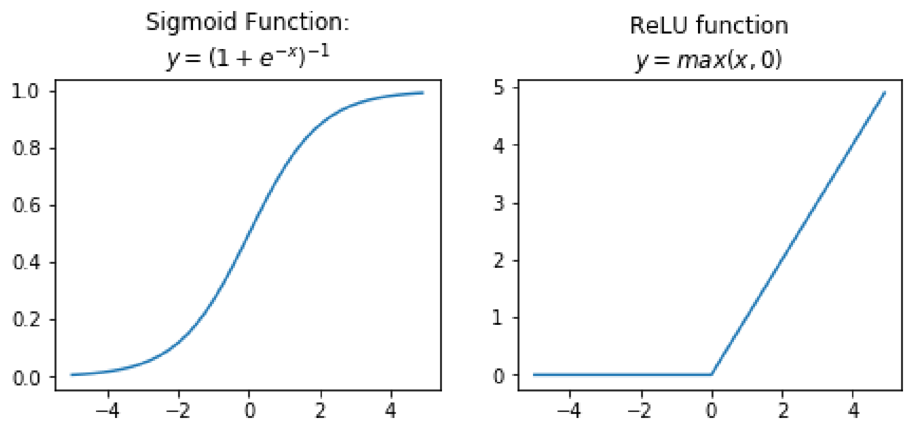


Figure 2.2: Plots of common non-linear functions

A neural network is constructed by linking many nodes together as shown below. Neural networks are usually organized into layers, as the figure above shows. The data of the input is given to the nodes in the input layer, which produce outputs that go to each node in the next layer. This process unfolds for each layer, until an output is produced at the output layer. As the figure above shows, the input to a neural network consists of multiple numbers, as does its output. The term for this kind of series of numbers is a vector. For example, we could represent the input to this neural network as $\vec{i}$, where $\vec{i} = [i_1, i_2, i_3]$. Linear algebra is the field of mathematics which deals with functions of vectors like this, and topics from linear algebra are frequently applied to neural networks.

The figure below depicts a network with a single hidden layer, that is, a single set of nodes with connections coming in from the input layer, and connections going out to the output layer. However, neural networks may have arbitrarily many hidden layers. A network which uses a large number of such hidden layers is called a "deep" network, which is what the term "deep learning" refers to.

The complexity which a non-linear function adds to the node is in fact essential to making neural networks capable of solving complex problems. The reason for this can be explained through linear algebra [2] which demonstrates that without the non-linear function, no matter how many hidden layers it has, the network is not able to model functions any more complex than a network

[2] Let f and g represent layers in out network without non-linear functions. These layers can be represented as affine transforms, such that $f(\vec{x}) = A_1\vec{x} + \vec{b}_1$, and $g(\vec{x}) = A_2\vec{x} + \vec{b}_2$. The composition $g(f(x))$ is then $A_2(A_1\vec{x} + \vec{b}_1) + \vec{b}_2$ or $A_2A_1\vec{x} + A_2\vec{b}_1 + \vec{b}_2$. Since A_2A_1 is a matrix, we may rewrite it as A_3, and since $A_2\vec{b}_1 + \vec{b}_2$ is a vector, we may rewrite it as $\vec{b}_3$. Therefore $g(f(\vec{x})) = A_3\vec{x} + \vec{b}_3$, and is itself an affine transform.

with no hidden layer at all (Neubig, 2017).

The process of providing a neural network with inputs, and propagating the numbers through each node is called "feeding forward". A network which simply sends its data in a single direction from the input to the output nodes is therefore called a "feed-forward network". Feed-forward networks are contrasted with recurrent neural networks (RNNs), which allow a sequence of inputs to be given, rather than a single input vector. The functionality of RNNs will be discussed later.

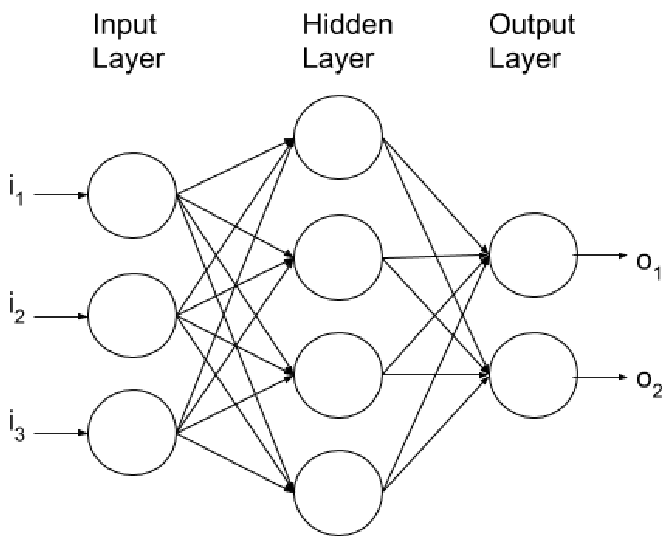


Figure 2.3: A basic structure of a neural network

Vectors and linear algebra are also applied in graphics rendering on computers. Determining the image that needs to appear on a computer screen requires calculating the color and intensity of over 1 million pixels multiple times per second. Therefore, the task of graphics rendering is very computationally expensive. To remedy this, specialized hardware, called Graphical Processing Units or GPUs were developed. GPUs are specialized computer processors designed to perform large amounts of vector calculations in parallel, in order to produce well-rendered images. Since GPUs are designed to handle large amounts of vector calculations, they can also be used to train neural networks quickly.

2.2.2 Training a Neural Network

Training a neural network refers to the process in which the network's weights and biases are adjusted in order to produce output similar to the training data. The manner in which the weights and biases are modified is determined through a process called backpropagation (Rumelhart et al., 1985), which will be described in general terms below.

When a neural network is first initialized, it is common that each of the weights and biases in the network are initialized to random values. Naturally, if an input from the training dataset is put into this network, the network's output would be highly different from the desired output. However, through the application of calculus and linear algebra, it is possible to calculate how much each weight and bias in the network needs to change, such that the network would output the expected output given the current input. The degree which each weight and bias in the network needs to change is called the "gradient". As a neural network trains, its weights and biases are updated according to the gradient, which is calculated for each training example that the network is given.

Of course, we would not want to adjust the weights and biases too drastically according to a single training example. The adjustments which would improve the output for one training example could differ with other training examples in the test set. A solution to this problem is to reduce the gradient by multiplying it by a small number (commonly around 0.001) called a learning rate. By multiplying this learning rate, the weights and biases are updated in a more gradual fashion, which allows the neural network to settle on a set of weights and biases which works well for all data in the training set.

To summarize, the general process of training a neural network is as follows:

1. Initialize the weights and biases in the network to random numbers
2. Repeat the following until the maximum number of early-stopping penalties has been reached:
    (a) Repeat for a given number of training examples:
        i. Feed the training input through the network
        ii. Use the gradient calculation to determine how much each weight and bias in the network should change
        iii. Reduce the gradient by multiplying it by a learning rate
        iv. Modify each weight and bias in the network according to the scaled-down gradient

    (b) Evaluate the network on the validation set
    (c) If the networks accuracy has not increased, increment the number of early-stopping penalties.

To ensure that the neural network does not overfit its training data, the validation process outlined previously will be applied. That is, the network will adjust to its training data repeatedly, until it reaches its maximum accuracy on the validation set.

Unfortunately, validation scores typically do not increase monotonically. That is, we cannot expect validation scores to consistently increase between each validation step as the network is trained. Validation accuracy may decrease briefly, only to increase again later on in the training process. Therefore, a method called early-stopping is used to determine when the network has reached its optimal validation accuracy. Early-stopping works by allowing a network to have n penalties, where a penalty is given when the networks validation accuracy decreases between training steps. After all of the penalties have been exhausted, the training process stops, and the model with the highest validation accuracy up to that point is selected as the optimal model. In my project, for example, I used early-stopping with $n = 10$ to determine the optimal translation system (Sennrich and Zhang, 2019).

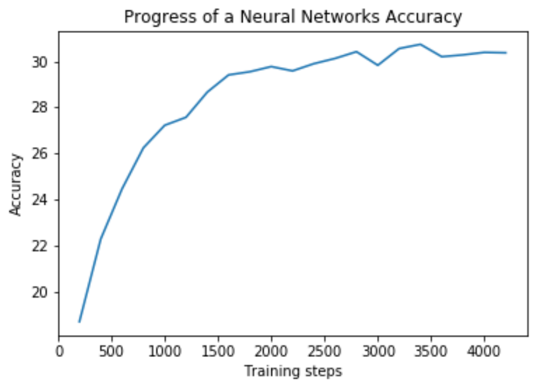


Figure 2.4: The validation score vs training step for an NMT model in this study

2.2.3 Engineering a Good Neural Network

The key strength of neural networks is their range of functions which they are able to produce. Since each weight and bias can be any number between positive and negative infinity, and since there are a large number of weights and biases, the number of possible combinations of weights and biases is an astronomical figure, even for a small neural network. This large number of combinations of weights and biases allow neural networks to represent an extremely large number of functions. In fact, Hornik et al. (1989) proved that neural networks are "universal function approximators", meaning that a neural network is capable of modelling any possible function, provided that there are enough layers in the network. This entails that neural networks are able to solve any problem which involves determining the proper output for a given input. Therefore, it is mathematically possible for a neural network to be constructed such that it can map every sentence from a source language to its correct translation in a target language. The challenge of developing neural networks is to collect a high-quality training set, and to construct a neural architecture which is able to converge to this function.

The flexibility of neural networks allows it to be applied to a very broad range of problems. Along with machine translation, neural networks are applied to face recognition, optical character recognition, speech recognition, and many other problems.

Although the sheer number of functions a neural network can represent makes it inevitable that there is a set of weights and biases which can solve any problem, it is also inevitable that there are large numbers of functions which are close to a solution, but fail for many of the outputs. That is, there are many possible functions which will closely resemble the best solution, but still be different in ways a network may not be able to pick up. It is a major challenge of the field of neural networks to ensure that the parameters a network settles on is a good solution to a problem, rather than a solution which seems good given our training data, but cannot generalize to other inputs. This section will describe techniques used to develop neural networks such that they are more likely to be trained to an optimal solution.

Dropout

It is sometimes the case that as a network is trained, the network comes to rely on a small number of weights and biases to produce its output. In this situation, only small number of weights and biases pick up the slack, while the majority of the weights and biases produce little effect on the output. This is not an ideal outcome, as we would like to ensure that as many of the weights and biases are used as possible to exploit the full expressive power of our network. A technique called dropout, developed by Srivastava et al. (2014), is used to alleviate this problem. When dropout is applied to a network, at each training step, a random number of nodes in the network are selected with probability p to be dropped-out, that is, they are temporarily excluded from the training process. The weights and biases are then updated for that timestep, but only between the nodes which have not been dropped out. In the following training step, a different set of nodes (again selected at random with probability p) will be dropped out, and the connections to those nodes will be trained. Randomly excluding nodes from the training process ensures that changes to the weights and biases will be more evenly distributed across the network. Therefore, the network should not rely too much on a few weights or biases, and instead use the full combinatorial power of the network. Choosing an ideal value for p is important: if p is too small, few nodes will be dropped out at all, and therefore the training will not be distributed across the network. If p is too high, too many connections in the network will be removed, and there is a risk that the connection between two layers may be severed. A value commonly used for p is around 0.1.

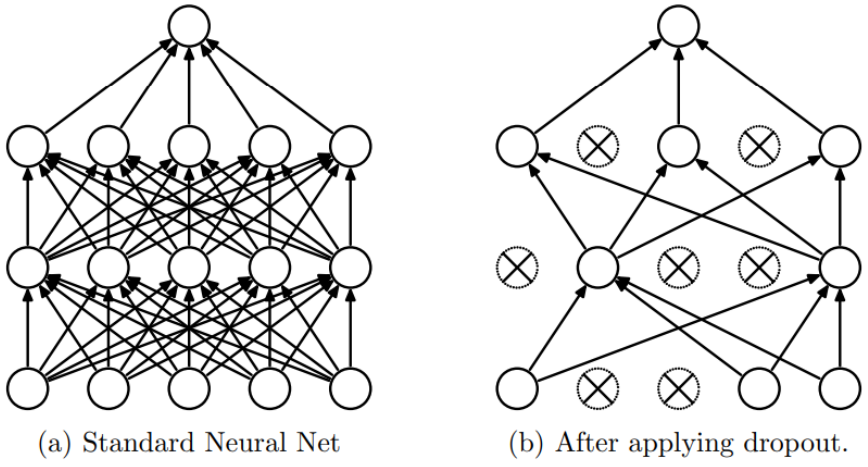


Figure 2.5: Figure 1 from Srivastava et al. (2014) demonstrating dropout

Hyperparameters and Grid Search

Neural network researchers will attempt to ensure that a network is able to model a solution function properly by adjusting aspects of the network, such as how many nodes there are in each layer, how many layers there are, and what the learning rate should be. These factors (among others) are known as hyperparameters, which in addition to the parameters (i.e. weights and biases), control how the neural network operates. Hyperparameters are typically selected via intuitive priors, using previous hyperparameters which worked well in other applications[3], or to empirically determine which combination of hyperparameters produces the optimal neural network.

It may not be the case that hyperparameters are independent of each other. For instance, it might not be the case that a dropout probability of 0.1 is optimal for all neural networks. It may be that a neural network which uses 20 hidden layers will perform optimally with a dropout of 0.05, while a network with 2 hidden layers performs optimally with a dropout of 0.2. If this is the case, we could make the intuition that a high dropout probability could make it difficult for information to propagate through a deep network, and therefore deep networks should have lower dropout probabilities compared to shallower networks. However, neural networks are very complex entities, and it could be the case that other hyperparameters we haven't accounted for would cancel this effect, or even produce the reverse effect. It is difficult to rely purely on intuitions when selecting hyperparameters because of this complexity. Therefore the ideal hyperparameters are often determined empirically, through a process called grid search.

[3]Note that this approach does not always work; hyperparameters often vary depending on the type of application

As the example above demonstrates, we cannot assume that a single hyperparameter which is optimal for one network is also optimal for all other networks. Therefore, multiple combinations of hyperparameters are tested, to determine which combination is optimal. An example with synthetic data is shown in the table below:

Dropout	0.05	0.1	0.2
5 layers	0.8	0.75	0.72
10 layers	0.82	**0.95**	0.8
20 layers	0.9	0.92	0.93

Figure 2.6: A hypothetical grid search, accuracy measured between 0-1.0

As the table, or “grid” above shows, the optimal combination of hyperparameters is 10 hidden layers, with a dropout probability of 0.1. Had we chosen 5 as our number of hidden layers, and only varied the dropout probability, we would have concluded that the optimal dropout probability was 0.05, when it would have been possible to achieve a higher accuracy for a different number of layers. The only way to determine the global maximum accuracy, each pair of hyperparameters needs to be tested.

A major drawback of performing grid search is that it requires fully training a neural network for each combination of hyperparameters. Training a single neural network is a time-consuming and computationally intensive process, therefore training many of them is even more expensive. For this reason, grid searches are very limited in practice; researchers will perform grid search on a small number of hyperparameters, relying on intuitions to determine the other hyperparameters. A balance between intuitive and empirical selection of hyperparameters is therefore critical for training an optimal neural network.

Unfortunately, grid searches which focus on a small set of hyperparameters can find misleading results. An example of this occurred in the development of the Transformer (Vaswani et al., 2017), a special neural network architecture designed to improve NMT quality while reducing training time. The Transformer makes use of multiple attention-heads, which are components of neural networks which determine the dependence between words in a sentence (attention will be described in more detail later). Vaswani et al. had the intuition that using multiple attention heads in the network would allow the network to discover different patterns in the relationships between words [4]. For instance one attention head may pick up on syntactic relations, while another picks up semantic relations. Vaswani et al. tested the hyperparameters in isolation, holding all other hyperparameters constant while the hyperparameter of interest was allowed to vary. In their experiment, Vaswani et al. found that the optimal number of attention heads was 16[5]. However, a study by Michel et al. (2019) which focused more

[4]Section 3.2.2 “Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.” (Vaswani et al., 2017)

[5]The numbers of attention heads tested was 1, 4, 8, 16, and 32. See Table 3 in Vaswani

closely on the effect of multi-headed attention on Transformers failed to reproduce these results. They found that most of the attention heads in the network could be removed with no statistically significant effects on the translation quality ($p < 0.01$). In fact, they found in some cases that removing all attention heads but one could have no effect on performance, contradicting Vaswani et al.'s intuition. Michel et al. claim that benefits of multi-headed attention depend on the manner in which a network is trained, i.e. that other hyperparameters are responsible for this result. This demonstrates that a hyperparameter search which is too narrow may produce misleading results. Narrow hyperparameter searches are nonetheless used frequently in neural network research, due to the expense of testing a large number of combinations of hyperparameters. It is therefore important to remain cautious when choosing hyperparameter values.

2.2.4 Recurrent Neural Networks

Applying neural networks to problems in natural language processing requires networks to take sequences of varying lengths as inputs. These sequences could be a series of words making up a sentence, or a series of phonemes making up a pronounced word. Since feed-forward neural networks can only take a vector of a fixed size as input, it is challenging to use them to model sequences of arbitrary length. Therefore, NLP often makes use of an architecture called Recurrent Neural Networks (RNN), which can take inputs of arbirary length. Translation systems which use RNNs will be used in this project.

A basic RNN can be constructed with only slight modifications to the feed-forward network described previously. Instead of taking a single input vector, an RNN takes two vectors as inputs: an input vector $\vec{w}_n$, and a hidden vector $\vec{h}_{n-1}$ (I will use w here to indicate that this input vector represents a word). Note that the hidden vector is not the same thing as a hidden layer, which is a layer of nodes inside the network. The RNN also outputs two vectors, an output vector $\vec{w}_{n+1}$, and another hidden vector $\vec{h}_n$. We can represent this with the following equation:

$$[\vec{w}_{n+1}, \vec{h}_n] = RNN(\vec{w}_n, \vec{h}_{n-1})$$

The key difference of the RNN is that once the hidden and word vectors are produced by the network, they are fed again into the network to produce the next word vector and hidden vector. The word and hidden vectors are repeatedly fed back into the RNN until the entire sequence is generated. To ensure that an RNN will not generate the sequence indefinitely, a special end-token is added to the end of each sentence in the training data. The network will continue to generate word vectors until this end-token is reached. Through this process, an RNN is able to model a sequence of data. The process of an RNN generating a sequence is shown below.

et al. (2017) for more details on their grid search

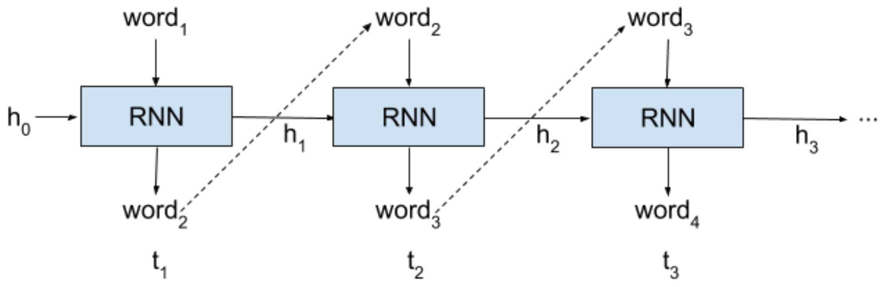


Figure 2.7: An unrolled RNN

Note that the figure above portrays a process unfolding over time, rather than the structure of a network. Each box labelled RNN here represents the same network, that is, it is the same collection of nodes, with the same weights and biases. This diagram is used to illustrate how sequential data is passed through the network over time.

The purpose of the hidden vector is to give the network a kind of memory, a way to remember what it has seen in the sequence thus far. As each step in the input process unfolds, the hidden vector accumulates information according to the word vectors it has seen. For example, at timestep t_3 in the diagram above, the vector $\vec{h}_2$ contains information about $\vec{word}_1$ and $\vec{word}_2$. The RNN then uses $\vec{h}_2$ to predict the word which most likely follows $\vec{word}_3$. The first hidden vector, $\vec{h}_0$, usually contains all zeros. We do not want our model simply to predict a word given only the previous word; we would like our network to make a prediction about the next word based on all of the words we have seen in the sentence thus far. The hidden vector is a way to represent this kind of information.

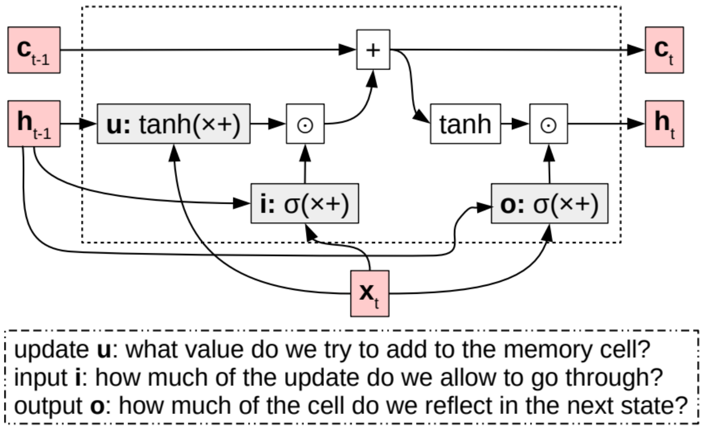
The use of a vector of a fixed size to represent all of the information seen in a sentence up to a certain point has come under criticism. As Professor Ray Mooney of UT Austin famously put it, "You can't cram the meaning of a whole %&!$#ing [*sic*] sentence into a single $&!#vector!" (Mooney, 2014). Mooney's intuition is blunt but straightforward: sentences with many words in them are more informative than sentences with fewer words. Therefore, using a vector of a fixed size to represent the information of a sentence of any length is not ideal. Approaches to represent sentences in better manners will be described later, but for now, a fixed-size vector will be used to represent the information of a series of words.

Additionally, the network generates a sentence monotonically. That is, it generates a sentence one word at a time from start to finish, using the immediately preceding word as input. Therefore, RNNs do not model sentences according to syntactic constituencies. Additional components, such as attention, need to be added to the network to model syntactic structure in sentences. Attention will be described later.

There is a critical problem with the RNN as it has been configured thus far.

This problem is that backpropagation, when applied to a recurrent network, will cause the gradients to compound as they propagate back through time. If the gradient at the last time step is less than one, it will cause the gradient of the previous timestep to be even smaller, and the gradients will continue to diminish until they have effectively vanished at the first timestep. This effect is known as the "vanishing gradient problem". One might propose a solution to this problem by multiplying the gradient by a constant greater than one at each timestep, so the gradient will not decrease. However this will cause the gradient to increase as it propogates back through the timesteps, and we will have an "exploding gradient problem", rather than a vanishing gradient problem. Therefore, a simple RNN structure as has been defined above is not ideal for modelling sequential data, as it is not able to adjust its parameters over timesteps.

To alleviate this problem, special neural architectures were developed to prevent the vanishing gradient problem in RNNs. Two popular approaches are called Long Short-Term Memory cells (LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Unites (GRUs)(Cho et al., 2014b). Essentially, the ways these architectures work is that they introduce components to the network such that they mathematically guarantee that the gradient will be exactly equal to one. This prevents the vanishing gradient problem, while still allowing information to propagate through each of the time steps (Neubig, 2017).



update **u**: what value do we try to add to the memory cell?
input **i**: how much of the update do we allow to go through?
output **o**: how much of the cell do we reflect in the next state?

Figure 2.8: An LSTM cell (diagram by Neubig (2017))

Although LSTM and GRU cells allow information to propagate through a network without any problems in the gradient, a problem still remains. That is that as the sentence increases in length, the information in the hidden vector accumulates. For example, if an RNN model is generating a 20 word sentence, $\vec{h}_1$ will model the single previous word, while $\vec{h}_{20}$ will model the information of all previous 19 words. Therefore, as an RNN approaches the end of a sentence, the context information in the hidden vector starts to get crowded, and therefore RNNs struggle with generating words at the ends of sentences.

This phenomenon was noticed empirically early on in the development of NMT. Cho et al. (2014a) found that the translation quality of sentences peaks when the sentence length is roughly 15 words, and falls off significantly when sentences are more than 20 words long. This phenomenon is especially relevant for our purposes, as our training data consists of Bible verses, which tend to contain longer sentences.

A simple solution to this problem is to add another set of RNN components (LSTMs or GRUs), but which model the sequence backwards, rather than forwards. This way, information is allowed to propagate in both directions when a sentence is generated. Words can therefore be generated not only based on the words which precede it, but based on the words which follow it as well. Networks with this structure are called bidirectional recurrent neural networks. Later, this project will use BiLSTM networks in order to improve translation results.

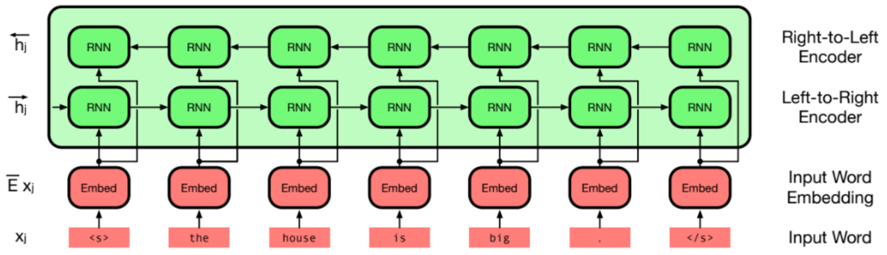


Figure 2.9: A bidirectional RNN (figure 8.2 from Koehn (2020))

2.2.5 Limitations of Neural Networks

It is important to recognize that despite the large expressive capacity of neural networks, there are still many limitations to neural networks to keep in mind when engaging in neural network research. Limitations which have been mentioned thus far will be summarized here.

Neural networks derive their power from the complexity of combinations of their weights and biases. This complexity allows neural networks to represent all possible functions, provided there are enough layers. However, this complexity also allows neural networks to represent functions which are close to an optimal solution, but not the global optimum. It's important to note that when a neural network converges on a solution, it is most likely one of these suboptimal solutions, rather than the global optimum.

Another issue with neural networks is the pattern of weights and biases is hard to interpret. That is, even if a set of weights and biases is found which produces a highly accurate neural network, it is very difficult to interpret how each of those weights and biases contribute to a solution. Although neural networks may be successful at solving a problem, it is very difficult to determine how the network solved that problem, or what patterns the network picked up on. For this reason, neural networks are not ideal for investigating the patterns among data, but are useful in replicating patterns from its training data. It is therefore critical when training a neural network to ensure that its training data is optimal, and that the network is properly configured and train to replicate the general patterns from the training data to produce accurate solutions.

Since the quality of a neural network depends heavily on its hyperparameters, that is, its structure and training strategies, it is important to determine optimal values for these hyperparameters. Unfortunately, there are a large number of hyperparameters for any network. Simply finding the optimal hyperparameter one at a time while holding all others constant, while being more efficient, may not find the true optimal hyperparameters, as the previous example with multi-headed attention exemplifies. However, testing all combinations of hyperparameters is not ideal either, as it requires a network to be trained from start to finish for each combination, which takes a great deal of time and resources.

Another issue with grid search not mentioned above is that it has a significant environmental cost. Training a neural network requires large amounts of electricity over a long period of time. Because of this, they have a large carbon footprint. This footprint is even larger for models which are trained on supercomputing clusters, which use even more energy to power their cooling systems. In their research, Strubell et al. (2019) found that performing grid search on the Transformer, a recent innovation in NMT using a large neural architecture, produces 626,155 lbs of CO_2. This is greater than the amount of carbon that 17 average Americans produce in a year, or the amount that 5 average cars emit in their lifetime. To ensure that advances in NLP are made while being environmentally sustainable, it is imperative for NLP research to focus on working smarter rather than harder, and developing NLP solutions which are more resource-efficient.

Neural networks, when applied to NLP, often rely on representations of words and sentences which deviate quite significantly from their representations in linguistics. An example of this mentioned previously is syntax; RNN networks typically generate a word sequence monotonically, with no account for syntactic constituents. Additionally, the semantics of a neural network can suffer, given that a single vector is used to represent the meaning of a sentence, regardless of its length.

In summary, the drawbacks of neural networks are as follows:

1. the large number of suboptimal solutions a network may converge to
2. the uninterpretability of the weights and biases of a trained network
3. the large number of hyperparameter combinations to test
4. the expense (in time, energy, and the environment)
5. the lack of applied concepts from formal linguistics

These challenges prove that producing good neural networks is a challenging research problem. Nonetheless, the high accuracy of neural network applications makes neural networks a worthwhile research field. Neural networks have proven highly useful when applied to tasks such as face recognition, speech recognition, and machine translation.

Although there are drawbacks to neural networks, their strength in solving complex problems is very useful. The popularity of neural networks has also made it necessary to develop ways of structuring and training neural networks, such that they can find a solution in a manner which is resource efficient. Additionally, it is important to extend the benefits of neural networks to a broader array of problems, such as the translation of low-resource languages.

2.3 Machine Translation Fundamentals

The field of machine translation began just after World War II, and has developed from proof-of-concept systems with 200 word vocabularies to high-quality translation systems available on mobile devices. The history of machine translation can be divided into roughly three periods, each of which was dominated by a particular paradigm of machine translation.

The first approaches to machine translation were developed in the late 1940's. Early MT researchers, such as Warren Weaver, suggested approaching machine translation like a code-breaking problem:

> *"When I look at an article in Russian, I say 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.'"* (Weaver, 1947)

What followed was the first wave of optimism in machine translation, with significant increases in funding for MT research. In 1954, the first machine translation system, developed by IBM and Georgetown University was demonstrated, which could translate sentences with a limited vocabulary from Russian to English. This system worked by pairing each word in Russian with corresponding words in English, along with a limited set of reordering rules to produce output which matches the syntax of English (Dostert, 1955).

The difficulty in constructing such a system, and its reliance on essentially verbatim translation, caused the first translation systems to be of poor quality. For this reason, the Automatic Language Processing Advisory Committee found that expense and poor quality of machine translation with respect to human translation did not justify continued government funding. This began a winter in machine translation research which persisted for roughly twenty years (Schwartz, 2018).

Starting in the 1980's research into machine translation was rejuvinated. These new approaches were developed to learn translations from corpora, as opposed to the previous approaches in the 1950s and 60s, which required a set of translation rules to be developed by linguists and programmers. These approaches used statistical methods to determine the most likely translation of a given sentence. For this reason, these approaches are referred to as statistical machine translation (SMT).

Bayes' theorem could be applied to a translation task as follows. If we want to translate a sentence F from a foreign language to English, we would want to find the sentence E such that the probability of E given F is maximized:

$$\hat{E} = \underset{E \in English}{\arg\max}\, P(E|F)$$

Bayes' theorem allows this formula to be restated as the following:

$$\hat{E} = \underset{E \in English}{\arg\max}\, P(F|E)P(E)$$ [6]

The calculation of the translation probability has therefore been broken into two problems: estimating $P(F|E)$, and estimating $P(E)$. The model which estimates $P(F|E)$ is called the translation model, which aims to determine which words or phrases in English are the best translation of the words in the foreign language. The model which estimates $P(E)$ is called the language model, which determines the likelihood that a sequence of words is a valid sentence of English (Jurafsky and Martin, 2008). A likely sentence of English is one which follows the syntactic rules of English, and potentially is semantically plausible. The translation model ensures that the translation is "adequate", meaning that the generated sentence accurately captures the meaning from the source sentence as much as possible. The language model ensures that the output is "fluent", meaning that the output sentence is a syntactically and semantically acceptable sentence in English. Good adequacy and fluency are the two main goals of machine translation (Koehn, 2020, p.63).

The statistical method above, especially when applied to phrases, remained the state-of-the-art in machine translation until 2014. Although neural networks and RNNs had been developed many years ago, it was the invention of the attention mechanism (Bahdanau et al., 2014) which made NMT competitive with SMT.

By 2015, neural networks began to be applied to NLP applications. What followed was a near complete takeover of the field of machine translation by neural network approaches, an event which Philipp Koehn has named "The Neural Turn". The Neural Turn unfolded only in the space of a few years; in 2015 a single neural system was entered in the WMT shared translation task, but it was not competitive compared with the other systems. In 2016, neural systems achieved the highest performance across all language pairs. By 2017, nearly all submissions to the WMT competition were neural systems, entirely replacing the previous paradigm of statistical machine translation (Koehn, 2020, p.58). The speed at which the Neural Turn unfolded underscores the relative youth of the field of neural network applications, and the gains in performance which neural networks are able to provide.

It's important to note here that although NMT has become the new state of the art in machine translation, SMT still outperforms NMT for many low-resource translation applications (Koehn and Knowles, 2017).

[6]Bayes' theorem would have these terms divided by P(F). But P(F) is constant in the argmax, which ranges over E. Therefore, P(F) may be removed from this equation.

This section will describe how neural networks are used in machine translation, and elaborate as to how they are relevant for this project. Additionally, this section will discuss the methods for automatic evaluation of a machine translation system, along with the drawbacks of these methods.

2.3.1 Neural Machine Translation

This subsection will discuss neural architectures for machine translation. In particular, it will discuss the use of RNNs in machine translation, and components and strategies which make NMT competitive. This section will also discuss the OpenNMT project, an open source project designed to encapsulate the common elements of machine translation. Finally, this subsection will discuss the Transformer architecture, the current state of the art, and how it is able to model sentences without the use of recurrent neural networks.

Word Embeddings

A key aspect which was not mentioned in the RNN section above is how words are actually represented by neural networks in NLP applications. Neural networks are mathematical functions, so in order to apply them to NLP, there needs to be a system of representing words with vectors.

Initially, words are represented in a neural network by the use of one-hot vectors. A one hot vector is a vector in which one of the values is 1, while all the other values are zero. Each word in a neural networks vocabulary is assigned one of these one-hot vectors, such that each word has a different one hot vector. For example, if we had a small vocabulary consisting of 5 words: “the”, “a”, “saw”, “dog”, and “cat”, these words could be represented by the following one hot vectors

$$\begin{aligned}
\vec{the} &= [1,0,0,0,0] \\
\vec{a} &= [0,1,0,0,0] \\
\vec{saw} &= [0,0,1,0,0] \\
\vec{dog} &= [0,0,0,1,0] \\
\vec{cat} &= [0,0,0,0,1]
\end{aligned}$$

Figure 2.10: Example one-hot vectors for a small vocabulary

Of course, one-hot vectors for a real NLP application will be much larger. Since each one-hot vector needs to be different for every word, the length of each one-hot vector must be equal to the size of the vocabulary. The vocabulary size of an NLP application is typically around 10,000 - 30,000 words.

However, simply creating an input layer to a neural network with a size of over 10,000 causes there to be a very large number of weights and biases in the network. Consider, for example, a simple feed forward network, with an input layer of size 10,000, an output layer with size 10,000, and no hidden layer. Each node in the output layer needs a weight for the inputs from each of the 10,000 input nodes. This means that this network would have 100,000,000 weights, all of which would need to be adjusted at each step of the training process. Simply using one-hot vectors like this is therefore not ideal for a neural NLP application, as it would take a great deal of time and computing power to train it.

To solve this problem, neural networks compress the information from these one-hot vectors into vectors with a smaller dimension, called word-embeddings. A word embedding typically consists of 100 to 1,000 floating point numbers, each of which may be a positive or negative real number. Each of these values in a word embedding is treated like the weights and biases in a neural network: that is, they are initialized to random values, and gradually updated through the training process such that they can mimic patterns in the training data.

$$\begin{aligned}\vec{the} &= [-0.1, 0.694, 0.436, 0.499, 0.474, ...]\\ \vec{a} &= [-0.735, -0.593, 0.997, -0.463, -0.547, ...]\\ &...\end{aligned}$$

Figure 2.11: Hypothetical word embeddings

An interesting effect of training word-embeddings is that when the training process completes, words which are semantically similar will tend to have word embeddings which have similar values. In our small vocabulary example, the words for “dog” and “cat” should develop embeddings which are similar, and similarly the determiners “the” and “a” should develop similar vectors. Through the dimension reduction process of principle-component analysis, (PCA), researchers have plotted trained word embeddings in two dimensional spaces, where clusters of semantically similar words can be visualized:

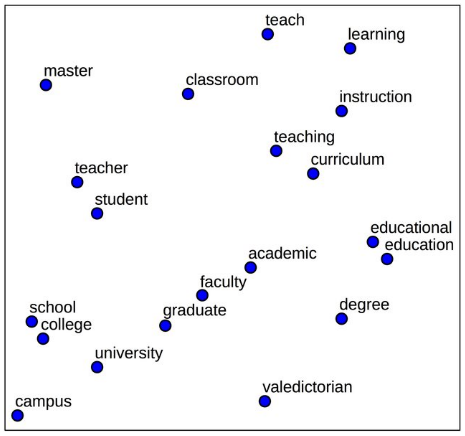


Figure 2.12: A PCA plot of trained word embeddings (Butnaru and Ionescu, 2018)

In the plot above, we can see certain clusters emerge among words related to education. For example, in the lower-left corner we can see a cluster of words related to locations or institutions, namely, “school”, “college”, and “campus”. In the top-left, we can see words relating to people, such as “master”, “teacher”, and “student”. In the top-right, we can see words related to pedagogy, such as “teach”, “learning”, “instruction”, and “curriculum”.

It’s important to note that two-dimensional plots cannot capture the richness of the relationships between word embeddings. As these plots are two dimensional, each point on the graph uses only two coordinates (i.e. two real numbers) to represent the location of a word embedding, and its relation to other word embeddings. However, word embeddings contain much more information than that. In this example, the vectors being plotted originally contained 300 real numbers. If we were able to visualize coordinates in a space with 300 dimensions, we would likely be able to make out semantic clusters in a much better way than plots like the one above can portray. Nonetheless, we are able to see basic semantic relationships in these simple graphs, which shows that word-embeddings are picking up on semantic information. We can also see efficacy of word embeddings in their applications, given that they have produced more accurate solutions to NLP problems.

An issue with embeddings as they have been described so far is that they cannot disambiguate between word senses. For example, the word “bank” will be paired with a single vector embedding, regardless of whether it refers to the side of a river, or a financial institution. More recent systems which generate word embeddings, such as BERT (Devlin et al., 2019) and ELMo (Peters et al., 2018), produce vector embeddings for words depending on their context, with the intuition that the context provides clues which can be used to disambiguate the word sense.

A recent analysis of language models which produce such context depended vectors has found that word embeddings also contain syntactic information. Hewitt and Manning (2019) found that a simple linear transformation on the embedding vectors for each word in a sentence would be laid out in a pattern which resembled its syntax tree[7].

The high dimensionality of word embeddings allow them to encode a great deal of information about words, such as their semantic and syntactic properties, while still being much smaller than one-hot vector representations. For these reasons, word embeddings have become very popular in recent neural network research, and are therefore important for neural machine translation.

Encoder-Decoder Models

RNNs in NMT make use of a particular kind of neural architecture called the encoder-decoder model (Cho et al., 2014b). This model consists of two RNNs, one of which is called the encoder, and one which is called the decoder. The encoder is provided with a sentence in the source language, and outputs a hidden vector once it generates an end token for that sentence. We can describe this hidden vector as an "encoding" for the sentence it was presented with. If the encoder is trained properly, this hidden vector should model the meaning of the sentence presented to the encoder.

This hidden vector is then passed to the second RNN, called a decoder. The decoder is trained to generate a sentence in the target language based on the hidden vector it receives from the encoder. If this input hidden vector accurately represents the meaning of the source sentence, a well trained decoder will be able to use that meaning to produce its equivalent sentence in a target language.

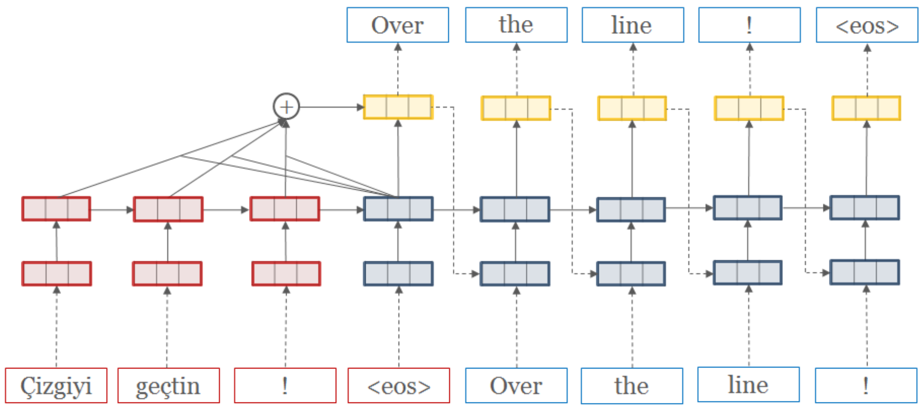


Figure 2.13: An encoder-decoder model for a Turkish to English translation system (Klein et al., 2017)

[7]The minimum spanning tree calculated for each of the transformed vector embeddings resembled the tree produced by a dependency grammar of English

Attention

The attention mechanism is a critical component for neural machine translation. The development of attention by Bahdanau et al. (2014) enabled NMT to be competitive with other machine translation approaches, and to eventually become the dominant paradigm in machine translation (Koehn, 2020, p.58).

Attention aims to address the problems mentioned previously about the hidden vector in RNNs. The first problem was that a single vector was being used to represent the meaning of an entire sentence, regardless of the sentence's length. There is also an additional problem which occurs in the encoder-decoder framework. Each word that the decoder generates is based off of the word which immediately precedes it and the hidden vector, which contains information about all of the previously generated words (both in the encoder and decoder). Intuitively, we can see that we do not need all the words in the source sentence to determine each output word. Rather, a network should pay attention to some words in the source sentence more than others. For example, if an RNN were translating the sentence "Das Haus ist weiss" into English, we would want our network to generate the word "house" using the information primarily from "Haus" in the source sentence (and possibly some information from "das"). We can conclude that other information in the source sentence is not pertinent to the generation of this word, but should be used to generate other words instead.

Bahdanau et al. (2014)'s approach was to add a mechanism to the decoder which can make these kinds of decisions. This involves calculating an attention score $\alpha_{i,j}$ for a target word w_j and each word in the source sentence w_i. In the previous example, the value for $\alpha_{Haus,house}$ would be high, while the values for $\alpha_{das,house}$, $\alpha_{ist,house}$, and $\alpha_{weiss,house}$ would all be low. The value of this attention score is then used to modify[8] the hidden vectors, such that the hidden states which are less important to the decoding process will be discounted, and more important hidden vectors will be prioritized. The attention mechanism enables the decoder to focus on the parts of the input sentence which are most relevent for generating the current word, rather than focusing on all of these words equally.

A useful aspect of attention is that it can be easily visualized. Here is an example from Bahdanau et al. (2014)'s study, showing the attention scores calculated between the words of French and English sentences:

[8]There are many ways in which the attention scores can be applied to the hidden vectors (Luong et al., 2015; Neubig, 2017)

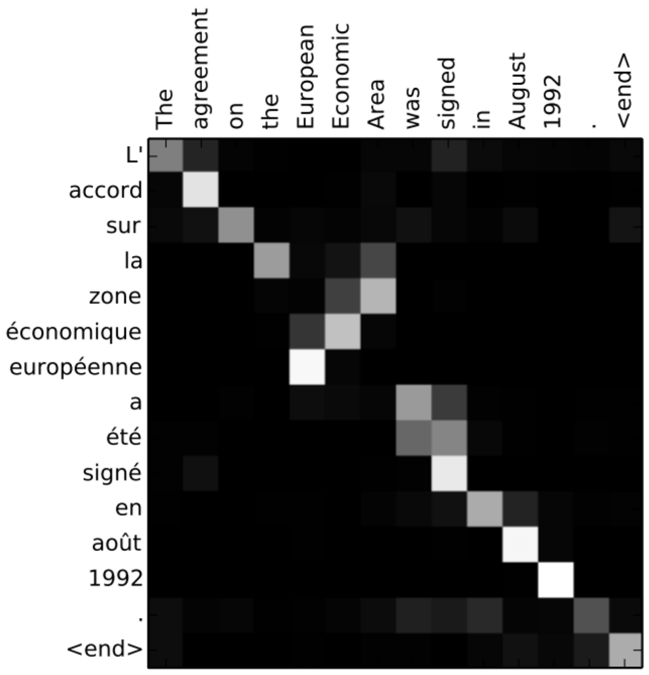


Figure 2.14: Attention between words in a French to English translation (Bahdanau et al., 2014)

Here we can see that each French word and its corresponding translation in English have the highest attention scores (colored white). We can also see that attention picks up some interesting information in phrases. For example, the generation of the word "was" in the English sentence had high attention values to the words "a" (has) and "été" (been). The attention mechanism here indicates that the network likely picked up that the word "was" is being used to form the passive voice, as the passive voice construction "a été" also occurs in the French sentence.

Another interesting phenomenon is that the network picked up the French adjective order. Most of the high attention cells in this graph form a diagonal line from the top left corner to the bottom right. However we see this pattern reverse in the phrase "European Economic Area", reflecting the syntactic differences between English and French. We also see that the attention in this phrase is paying secondary focus to the adjacent words in the same phrase, which is likely the key to how the attention mechanism and the neural network model the syntactic structure of this tree.

Finally, we can see that attention scores are much higher between content words than between function words. This is probably due to the fact that content words such as "agreement", "European", "signed", and "August" are more likely to have a single verbatim translation when compared to words like "on", "in", or "the", which may correspond to many words in French.

These phenomena in this figure demonstrate that the NMT system with attention is able to focus on a few relevant words in the source sentence to produce a translation, rather than all of the words in a sentence. This ability to focus on just the relevant words was key for neural networks to be a viable solution for machine translation problems.

Byte-Pair Encodings: NMT morphology and Rare Word Translation

A key issue of neural machine translation (and for machine translation in general) is the translation of low frequency and out-of-vocabulary (OOV) words. As the previous section on word embeddings shows, NLP applications typically represent words using a fixed vocabulary. In practice, however, languages do not have vocabularies which are fixed. Languages may form words productively through the following processes:

- Compounding (e.g. "Sonnensystem", German for "solar system")
- Phonologically transcribed proper names (e.g. "Barack Obama", "Барак Обама")
- Morphologically complex words (e.g. "pagunghalighnaqaqa", Yupik for "I am going to put crowberries in it"

Morphologically complex words are especially common in languages like German, which has many compound words, and for languages like Yupik with agglutinative morphology.

One solution for translating rare or OOV words is to use what is called a dictionary backoff. With this solution, if a translation system encounters a word in its input which is not in its vocabulary, it will look up that word in a separate bilingual dictionary. Otherwise, the system will translate each word according to the patterns it learned from the training data. Translation systems which use a backoff dictionary outperform those which copy the unknown words from the source sentence, or those which use an "UNK" token[9]. However, a backoff dictionary may not to contain all instances of compound words, morphologically complex words, or proper nouns. For this reason, it is useful to break words into more translatable components.

As the backoff dictionary approach cannot process morphologically complex words, other approaches are necessary to correctly translate these words. Another is to model a sentence as a sequence of characters, rather than a sequence of words. This approach gives the task of word-segmentation to the neural network, which through training, should learn how characters should be grouped together to form words and subword units, such as morphemes. Cherry et al. (2018) found that such character-based models outperform BPE based models (which will be described later) across a variety of language pairs. However, this

[9]See Table 2 in Sennrich et al. (2016b)

comes at the cost of requiring the neural network to be deeper, which entails a larger number of weights and biases, and therefore a longer training time. Our study did not pursue character-based translation for this reason, although future work may demonstrate that character-based translation models are ideal for polysynthetic languages. The BPE algorithm which Sennrich et al. (2016b) propose is much more efficient than using a character-based model, and will therefore be used in this study.

Sennrich et al. (2016b) analyzed a set of 100 rare words[10] in a German-English parallel corpus. They found that 56 words were compound words, 21 were proper names, 6 were loanwords with predictable orthographic/phonological changes (e.g. “emancipate” → “emanizipieren”, and 5 were formed through affixation (“süßlich” → “sweetish”). Although each of these word categories form words not seen in the corpus, they are potentially translatable, provided that they are broken up into their appropriate units (i.e. morphemes or phonemes). Based on these figures, we can estimate that a system which segments these words into the appropriate units can allow roughly 88% of rare words to be properly translated.

Sennrich et al. (2016b) applied the byte-pair encoding (BPE) algorithm to produce such a segmentation. The BPE algorithm was originally designed to produce file compressions (Gage, 1994). File compression algorithms generally look for frequently recurring sequences of characters, and replace them with short codes to produce a compressed file. Such compressed files are shorter than the original file, but nonetheless contain the same amount of information.

The BPE Algorithm The BPE algorithm begins by splitting all words in a text file into separate characters. These characters are the first items added to the vocabulary produced by the BPE algorithm. Each item in the vocabulary produced by the BPE algorithm is called a “subword”, as each word of the input file may be segmented into one or more units.

The algorithm proceeds by counting the frequency of each pair of subwords in the file. The most frequent subword pair is joined, forming a new subword which is added to the vocabulary. For example, if the BPE algorithm is applied to an English text, the first subwords merged are “t@@”, “h@@” → “th@@”, as “th” is the most frequent character pair in English texts. The fact that “th” represents a single phoneme in English (/θ/) demonstrates that the BPE algorithm can pick up on phonological properties of a language reflected in the orthography.

Once the most frequent subwords are joined, the frequencies of the subword pairs are recalculated, and the next most frequent subword pair is merged. This process is repeated N times, where N is the desired size of the vocabulary. Finding an optimal value for N is important for optimal results; if N is too small, the vocabulary will not be able to group characters together into words or morphemes very well. If N is too large, words will not be split into useful units, such as morphemes. Therefore, the size of the BPE vocabulary becomes

[10]In this case, a rare word is not one of the 50,000 most frequent words

another hyperparameter, which must be optimized for translation quality.

The BPE algorithm does not combine whole words together, rather only subwords may be merged. Therefore, the BPE algorithm will not add common phrases to its vocabulary. It may be the case that common phrases would be beneficial to add to the vocabulary, however doing so would require a large increase in the computation time of the algorimth. For this reason, Sennrich et al. (2016b) do not merge subwords across word boundaries.

1. Split a file into its characters
2. Repeat the following N times:
    (a) Calculate the frequencies of each subword pair in the file
    (b) For the most frequent subword pair:
        i. Combine each of its subwords in the source file
        ii. Add the new subword to the vocabulary set

Figure 2.15: The BPE algorithm

After the BPE algorithm has been applied to a text document, what remains are common words which remain unsegmented, and words which have been split into a series of more frequent subwords. An example of an English sentence with BPE segmentation is shown below:

N	**English sentence**
$1,000$	`The Inuit tea@@ ch@@ ers , we en@@ v@@ y them because they are the only g@@ rou@@ p of enti@@ re@@ ly Inuit tea@@ ch@@ ers in Nunavut .`
$6,000$	`The Inuit teachers , we en@@ v@@ y them because they are the only group of enti@@ rely Inuit teachers in Nunavut .`
$12,000$	`The Inuit teachers , we en@@ vy them because they are the only group of entirely Inuit teachers in Nunavut .`

Figure 2.16: Various BPE tokenizations with varying vocabulary sizes

The figure above shows the same sentence after BPE segmentation with varying vocabulary sizes (N). The symbol "@@" indicates that the token should be joined with the token to the right of it to form a word, so the tokens "enti@@ re@@ ly" should be considered as the single word "entirely". As the value for N is increased, infrequent and morphologically complex words, such as "entirely" and "teachers" gradually emerge as unsegmented words. Varying the value for N causes different linguistic elements to surface in the document. The first tokens to merge, (often "t" and "h" in English) may represent phonemes represented by multiple letters. Subwords produced later may represent morphemes or components of compound words (e.g. "-ly", "-ers". Finally, subwords may represent entire words ("group", "Inuit"). Therefore, subword units provide a manner of representing phonological as well as morphological information from the input text.

It's important to note that even if the BPE algorithm finds a segment which matches a morpheme in a language, the algorithm will not necessarily find that morpheme in all words. For instance, in the example above, the word `entirely` does not have the `-ly` suffix identified, because the sequence `entirely` is among the 12,000 most occurring sequeneces of characters in the text. However, a less common word, such as "explicitly", does not occur frequently enough to be merged into a single sequence may be tokenized as `explic@@ it@@ ly`. Therefore, the BPE segmentation will make use of these subword units only for the less frequent words. Subwords corresponding to morphemes and phonemes are not identified consistently across all words.

A key benefit of the BPE algorithm is that it is unsupervised. That is, it is able to uncover phonological and morphological information based only on the source text, without needing any external analysis. Unsupervised algorithms have the strengths of being able to find linguistic patterns in much less time compared to human linguists. However, unsupervised algorithms suffer from the lack of intuitive knowledge, and can be thrown off by noise in the data, such as typos. For these reasons, it is important to recognize the limitations of the BPE algorithm.

BPE Limitations Occasionally, the BPE algorithm will identify subwords which have no linguistic basis. An example of this could be that the word "station" is tokenized as `st@@ ation`. The suffix "-ation" does carry linguistic meaning in some words, as it nominalizes the word it attaches to, as in "nominalization". However, the "-ation" in "station" does not carry this meaning, and the `st@@` is neither a morpheme nor a phoneme. The BPE algorithm is not able to account for cases like this. Unlike a human linguist, it cannot identify that the suffix "-ation" is applied to verbs, and that other words ending in "-ation", such as "station" or "nation" should not be segmented in this way. Nonetheless, Sennrich et al. (2016b) find words which are segmented in ways counter to linguistic expectations are nonetheless accurately translated in most cases. For example, they found that the German word "Forschungsinstituten" ("Research Institutes"), instead of being segmented into the more linguistically plausible `Forschungs@@ institut@@ en`, is segmented as `Forsch@@ ungsinstitu@@ ten`, but nevertheless correctly translated to "research institutes" by their translation systems.

Another limitation of the BPE algorithm is that it relies on splitting words according to their letters, and therefore cannot represent phonological alternations, or orthographic conventions which do not reflect the phonology. For example, the final "e" in "nominalize" is silent, and does not form part of the phonological representation. Therefore, when "-ation" is suffixed to it, the "-e" is deleted. Tokenizers such as the BPE algorithm do not account for the deletion of characters like this, and will tokenize this word as `nominaliz@@ ation` rather than `nominalize -ation`. This problem is particularly pronounced in languages like Yupik, where there is a high degree of morphophonological alternations. Simply splitting whole words into commonly occurring subsequences will therefore not be able to accurately identify morphemes which may appear as different allomorphs in a word.

Alternatives to BPE have been proposed, which aim to more accurately reflect the morphology of the language. One of these approaches is Morfessor, a statistical system for morphological segmentation. The approach of Morfessor is to determine the set of word segmentations which is most probable given the source data. Such an approach would not make the mistakes like the `st@@ ation` segmentation mentioned above, because although "ation" may be a frequently occurring suffix in a text, "st" is a very infrequent (if not nonexistant) prefix. Therefore, Morfessor rejects segmentations such as `st@@ ation` in favor of segmentations in which all segments can be found in the training corpus with high probability (Smit et al., 2014).

However, like BPE, Morfessor simply splits words into character sequences, and does not account for character deletion or alternation. A handful of parsers, such as Allomorfessor (Virpioja et al., 2009) have been developed which can account for such mutations. However, accounting for possible character mutations significantly increases the computational complexity of the problem, and thus far, parses which account for these mutations have not been competitive with other parsers, such as Morfessor.

Sennrich et al. (2016b) found that translation systems using BPE outperformed systems using other methods of handling unknown words, such as backoff dictionaries, or UNK replacement. BPE translation systems demonstrate the best performance in general translation quality by a slim margin as measured by the BLEU scores (to be described later). In terms of translating individual words accurately[11] (called unigram accuracy), the BPE systems also outperformed the other methods, especially for the accuracy of rare words. These refinements provided by BPE enabled further improvements in NMT performance.

The quality of BPE tokenization was also tested empirically on the Inuktitut language by applying it to machine translation. Inuktitut is an Inuit language closely related to Yupik, and has a similarly productive and complex morphological system. Micher (2018) tested machine translation systems using a variety of tokenizers: namely a BPE tokenizer, Morfessor, and a tokenizer specifically developed for Inuktitut. Their findings demonstrated that BPE tokenization performed competitively in all tasks, however, in some tasks BPE was outperformed by the pre-made tokenizer or Morfessor.

Because of its empirical successes, BPE has become the state-of-the-art method of tokenization in NLP, especially in NMT (Micher, 2018). For these reasons, BPE tokenization will be used in this project. It may be the case that a tokenization system could be developed which outperforms BPE by produc-

[11] Measured by F_1 score

ing segmentations which more accurately represent the true linguistic units of a text. However, this question will not be explored in this project.

OpenNMT

OpenNMT is an open-source neural machine translation system. It was developed by the Harvard NLP group and SYSTRAN, and has received contributions from many other developers and researchers. ONMT was designed to provide a framework which is common across all NMT architectures, while allowing the architecture and hyperparameters to be customized by the user. Using its default settings ONMT is able to achieve state-of-the-art performance on machine translation tasks (Klein et al., 2017).

OpenNMT is able to produce an NMT system between any language pair, requiring only a bilingual corpus to use as training data, and a computer with a GPU available for the training process. ONMT contains a set of scripts which can clean and tokenize the corpus, including learning and applying a BPE tokenization. OpenNMT breaks the translation process into three steps, each of which has a dedicated python script:

1. preprocess.py - This script prepares the necessary inputs for the training script, such as an object representing the vocabulary
2. train.py - This script initiates the training process using the preprocessed data as input. It performs the training process according to the hyperparameters which the user sets, and continues the training process until user specified conditions are reached. At multiple intervals during the training process, train.py will save the current state of the translation system.
3. translate.py - This script will load a translation system produced by the training script. The script will then produce an output file, which is the translation of the input file.

Once the training process has been completed, the translate.py script is used to translate a test file. This translation is then compared to the human translation, and automatic evaluations are calculated. How these evaluations are calculated will be described below.

The benefit of ONMT is that it allows NMT systems to be created which are highly customizable, without needed to write the code for an NMT system from scratch. The train.py script allows a wide range of hyperparameter values to be adjusted, such as the size and structure of the network, the learning rate, dropout percentages, among many others[12] By default, the hyperparameters of the train.py script were those which achieved state-of-the-art results, however, note that these were determined for high-resource languages.

One of the hyperparameters which OpenNMT allows users to customize is the NMT architecture type. This project will focus on the RNN translation

[12] A full set of hyperparameters and their default values can be found here (http://opennmt.net/OpenNMT-py/options/train.html).

system, however, OpenNMT may also handle the Transformer architecture as well.

The Transformer Architecture

The Transformer architecture was introduced by Vaswani et al. (2017). Broadly speaking, the main innovation of the Transformer is that it entirely removes recurrence from the neural architecture, relying solely on the attention mechanism. Hence the title of the paper describing the Transformer, "Attention Is All You Need".

Removing the recurrence from the architecture has two main benefits. The first is that it eliminates the problems of representing the meaning of an entire sentence with a single fixed length vector. Instead, the Transformer uses the attention mechanism to determine how much focus to give to each word in the input sentence, when generating a word in the output. The second benefit is that the Transformer is much more parallelizable compared to RNN networks. The calculations required to train a Transformer are relatively independent of each other, and therefore the training process can be distributed to a large number of GPUs working simultaneously. Running the training process in parallel allows the training process to be completed in a much shorter span of time, provided that one has access to large computational resources. For RNNs, each calculation at a timestep requires that all of the calculations for a previous timestep to be completed. The interdependence of the calculations of RNN networks is therefore difficult to parallelize.

The Transformer also made innovative uses of the attention mechanism. One innovation is that attention mechanisms are used both between words of the input sentence, called "self-attention", along with the usual attention mechanism from words in the output sentence to each word in the input sentence. Additionally, each attention mechanism contains multiple attention "heads", that is, multiple separate attention mechanisms with the aim of detecting multiple patterns of relations between words. Although, recent research which focused specifically on multi-headed attention found that in many cases it does not significantly increase performance over single-headed attention (Michel et al., 2019).

Because the Transformer removed the recurrent architecture, it removes the ability of the network to determine the relative locations of words in a sentence. Without recurrence, a Transformer would not know whether two words were immediately adjacent, or farther apart in the sentence. To solve this problem, a "positional encoding", based off of the word's position in the sentence, is added to each word embedding. The positional encoding is based off of a sinusoidal function, enabling the network to learn the importance of the relative positions of words.

Like the RNN translation models, the Transformer can be broken into two parts: an encoder, which produces a series of encodings of the input sentence, and a decoder, which reads this encoding and produces an output sentence. The encoder consists of a stack of N cells, where a cell consists of a single application of the multiheaded self-attention mechanism. The decoder similarly consists of N cells, where each cell consists of an application of self-attention between the embeddings for each of the input words, followed by an application of the attention to the words in the source sentence. Vaswani et al. (2017) found an ideal value of N to be 6.

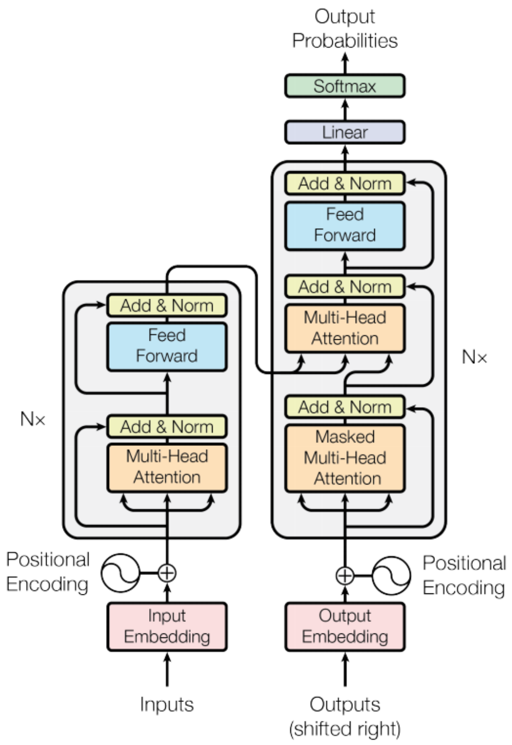


Figure 2.17: The Transformer architecture from (Vaswani et al., 2017)

Vaswani et al. (2017) demonstrated that the Transformer could achieve new gains in translation quality, and therefore this new architecture became the state of the art. Lakew et al. (2018) also perform a through comparison between RNN and Transformer based translation systems. They found that Transformers consistently outperform RNNs, both in terms of general metrics such as BLEU and TER, and produce fewer word-level errors, morpheme-level errors, and word-ordering errors.

The increase in quality provided by the Transformer comes at the cost of an increase in the numbers of parameters (i.e. weights and biases) to train. Although the Transformer allows its calculations to be more easily completed in parallel, Transformers nonetheless require a larger number of calculations overall compared to RNNs. For this reason, Transformers are more computationally expensive to train than RNNs to train. This computational expense can also become an environmental expense; training a single Transformer model can produce 192 lbs of CO_2 (Strubell et al., 2019).

Although applying the Transformer to the task would likely improve performance for Yupik-English translation, the expense of the training process makes this project difficult. Therefore, the current project will use the RNN architecture in all experiments, as RNN translation systems require fewer parameters, and can be trained in a shorter period of time. A future study could apply the findings of this study to the Transformer architecture to further increase translation performance.

2.3.2 Evaluation Metrics

Producing a perfect evaluation of a translation is a very difficult task. Determining if a translated sentence accurately reflects the meaning of the source sentence is fundamentally subjective, and there are multiple aspects of meaning which need to be accounted for. For these reasons, assessing the quality of a translation can even be difficult for human to do, much less an algorithm. It is important to keep these factors in mind when understanding how a machine translation system is evaluated. Nonetheless, automatic metrics, such as the BLEU score, have been developed which successfully correlate with human judgements of translations, while being much faster and cheaper than human evaluation. Therefore, automatic evaluation methods are useful for assessing the general quality of translation systems.

Human Evaluation

The most accurate evaluations possible for a machine translation system come from human evaluators. Human evaluations of MT systems involve presenting an example translation to a person, and asking that person to rate the quality the translation in terms of its ability to convey the meaning of the source sentence (“adequacy”), and how natural the output sounds in the target language (“fluency”). Assembling the results of multiple human evaluators can be a challenge, because some evaluators may be more generous than others, and evalatuators may disagree as to how well the target sentence captures the meaning of the source sentence (Koehn, 2020, p.63-67). Researchers therefore need to calibrate their human judgements to have a fixed basis of comparison.

Another major challenge of human evaluation is that it is expensive, in terms of time and money. Human evaluators perform much more slowly than their automatic counterparts, and often need to be paid for the evaluation tasks. This problem is compounded by the fact that MT systems are evaluated with thousands of sentences, because simply evaluating the quality of a small number of sentences would not indicate the overall quality of the system.

Some researchers have opted to collect human evaluators through crowdsourcing platforms, such as Mechanical Turk (Bloodgood and Callison-Burch, 2010). However, crowdsourcing introduces additional problems, particularly of quality. Researchers using crowdsourced data need to design their evaluation questionaires in such a way that they can indicate if each evaluator is reliable or not, and then filter out the unreliable evaluations (Koehn, 2020, p.68).

The difficulties in producing accurate and inexpensive translations make it very difficult to evaluate MT systems with human judges. Therefore, automatic evaluation metrics which approximate human judgements with markedly less time and money have become popular in assessing MT quality.

The BLEU score

The most popular evaluation metric in machine translation is the Bilingual Evaluation Understudy, or BLEU score (Papineni et al., 2002). The BLEU score is calculated by determining the similarity of an MT systems translation of a document and one or more human translations of the same document. The similarity is usually expressed as a number between 0 and 100 with 100 being the best possible score.

The BLEU score is generally calculated by determining the percentage of words in the machine generated translation which also occur in at least one of the reference translations. Therefore, BLEU is an example of a precision-based score, meaning that it calculates the percentage of correctly translated words among the total number of words the system produced in its output.

The BLEU score does not need to be calculated on words; the same metric could be calculated on pairs of words, or bigrams, as well as trigrams, or sequences of length n (n-grams). Papineni et al. (2002) found that precision scores for n-grams with n ranging from 1 to 4 all correlated well with human translation evaluations. To leverage the quality measurements from each n-gram precision, the geometric mean is calculated for each of the n-gram precisions from 1 to 4, as the equation below shows:

$$BLEU = brevity \times exp\left(\sum_{i=1}^{4} \frac{\text{matching i-grams}}{\text{total i-grams in machine translation}}\right)$$

$$brevity = min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)$$

Figure 2.18: Equation 4.10 from Koehn (2020)

An issue with using precision-based scores is that they can produce artificially high scores for short sentences, if those sentences contain words which are also in the reference translation. For example, if an MT system produced a sentence which is just “the”, it would have a precision score of 100% if the reference translation is “there is a cat is on the mat”, because all of the words in the MT system’s output are also in the reference translation. To prevent this problem a brevity penalty is multiplied to the geometric mean of the n-gram precision scores, such that an output sentences which are shorter than the reference translations are discounted.

The BLEU score was found by its designers and Coughlin (2003) to correlate strongly with human judgements of translation quality. BLEU scores are therefore a good method of approximating human judgements of machine translation, while being much faster and cheaper to produce. For these reasons, the BLEU score is the most popular metric for evaluating MT systems. BLEU scores, calculated using the multibleu.perl script from the Moses project (Koehn et al., 2007), will be used to evaluate the systems produced in this project.

Unfortunately, there is not a specific range of BLEU score which is considered by the field to be indicative of a "good" translation system (Koehn, 2020, p.76). Additionally, the BLEU score which may indicate an optimal BLEU score varies depending on the languages and domain of the training data. BLEU scores are typically used to compare different translation architectures trained on the same dataset, rather than provide an absolute indication of an MT system's quality. State-of-the-art MT systems will usually achieve a BLEU score between 20 and 40, as exemplified in the top performing models of the newstest2019 shared task of WMT19 (Koehn, 2019). For the purposes of this project, a Yupik translation system will be considered to be of decent quality if it achieves a BLEU score above 15, i.e. slightly lower than the state of the art.

An issue with BLEU scores as well as other automatic translation metrics is that an MT model will be able to translate some sentences more accurately than others. If by chance the test set contains a large number of such sentences, the score for the model will be artificial high. Therefore, any machine translation metric is not statistically certain, and a margin of error should be appropriately determined. Calculating a margin of error for a BLEU score is mathematically difficult, so a margin of error is estimated through a process called bootstrap resampling. This process involves calculating the evaluation metric for a random sample of sentences from the test set multiple times. The range of scores produced can be used to approximate the true statistical uncertainty of the metric (Koehn, 2004). Translation metrics in this project will use this method to estimate uncertainty.

Limitations of BLEU scores

Since its development, the BLEU score has received a great deal of criticism (Koehn, 2020, p.75-76). It is important to be aware of these criticisms when interpreting a BLEU score.

The first major issue of a BLEU score is its difficulty to interpret. For example, if a system attains a BLEU score of 20, it is hard to determine just from this score whether the translation system performs well or poorly, and if it performs poorly, what aspects of the translation are poor. Unlike metrics such as word-error rates used in speech recognition, which correspond to an understandable quantity, BLEU scores do not correspond to a single property like this. Although BLEU scores provide a good indication of the relative performance of translation systems, it is hard to interpret in an absolute sense what a BLEU score means.

BLEU scores also have difficulty assessing overall coherence of the sentence, as they do not consider matching sequences longer than four words. Because of this, two MT systems which perform well at producing matching N-grams, but differ in their ability to represent larger syntactic phenomena, will be indistinguishable in terms of BLEU scores.

Another issue of the BLEU score is that it requires words in the output and reference translations to match exactly. Words which differ only by a single morpheme, or semantically synonymous words, will be considered as entirely incorrect. Other translation metrics attempt to account for these discrepencies, either by calculating BLEU scores based on morphemes rather than whole words, or considering allowing synonymous words to match.

BLEU scores also put equal emphasis on each word in the sentence. Therefore, if the word "not" were eliminated from a sentence, the meaning of the sentence could drastically change, even though only a single word was deleted. BLEU is not able to account for the differing importance of various words.

As BLEU scores are based off of precision, which is a percentage, the maximum achievable BLEU score is 100. However, even very high quality translation systems will not score higher than 50. Therefore, a BLEU score which is much lower than 100 does not necessarily indicate that the machine translation system is performing poorly. Additionally, comparing two human translations using the BLEU score result in scores which are only slightly higher than MT BLEU scores, although the human translations are of much better quality (Koehn, 2020, p.76).

Alternative metrics

Although the BLEU score is the most popular evaluation metric, several alternative evalution metrics have been developed aimed at addressing the shortcomings of BLEU scores. These scores aim to provide a metric which is easier to interpret (TER), or aim to reflect semantic similarity (METEOR, YiSi).

The Translation Error Rate score is a score which is calculated in a manner very similar to word-error rates, used in speech recognition systems. The TER rate is calculated by counting the number of added, deleted, and shifted words between an output translation and a reference translation. TER scores are quite intuitive; a TER of 11% indicates that roughly one in nine words will be translated erroneously by the system. The main drawback of TER is its computational complexity. TER scores are designed to be flexible in terms of the position of words in a sentence. Therefore, TER scores will consider the sentences below as nearly identical, even though sequentially, the sentences are quite different:

"A spokesperson announced today: 'The plan will go forward.'"
"'The plan will go forward,' a spokesperson announced today."

Figure 2.19: Nearly identical sentences with different word sequences (Koehn, 2020, p.72)

Developing an algorithm which is able to calculate all possible sentence reorderings as the sentence above shows is an NP-complete problem, meaning that the search space grows exponentially with the length of the sentence.

Another translation metric is METEOR (Banerjee and Lavie, 2005). METEOR aims to address two flaws of the BLEU score, namely the inability to match morphologically similar words, and the inability to match synonymous words. METEOR first evaluates translation quality by determining the words which match in the output and reference translation. If the words don't match, METEOR backs off to using a basic morphological parser, called a stemmer. For example, if the word "responsibility" does not match the reference translation, METEOR checks to see if "responsible" is a match. If that match fails as well, METEOR backs off to comparing the words by semantic classes, as determined by WordNet (Miller, 1995), a rich ontological database classifying words into various levels of synonymous classes. Banerjee and Lavie (2005) found that METEOR correlated much more strongly to human evaluations compared to BLEU scores ($\rho = 0.964$ vs. 0.817), proving that incorporating morphological and semantic information in evaluation metrics is beneficial. However, METEOR requires a large amount of external resources, namely, a morphological parser and semantic ontology, for a language. METEOR is therefore very difficult to apply to a low-resource language, for which such parsers and ontologies may not have been developed.

Another, more recent evaluation metric based on semantic information is the YiSi metric (Lo, 2019). In general, YiSi works by determining the semantic similarity of words in an output and reference translation. These matches are then aligned into semantic frames, representing predicates and roles in the sentence. Once these alignments are produced. The precision and recall are calculated between the output and reference semantic frames, and are combined into the F1 measure to return the final YiSi score. A YiSi score is a real number ranging from 0 to 1.0, where 1.0 is given to exactly matching output and reference translations.

One key benefit of YiSi scores is that they give more weight to words which have lower frequency. This corresponds to the intuition that the most common words of a language are typically function words, and primarily perform syntactic functions, rather than providing semantic information. Low frequency words, on the other hand, are more likely to be content words (i.e. nouns and verbs), and therefore carry the majority of the meaning in the sentence. Applying different weights to words in this manner is a key improvement over BLEU scores, which assign equal importance to all words in the sentence, regardless of their semantic importance.

There are three types of YiSi scores which differ in the manner that they calculate semantic similarity of words. Each type requires a varying level of resources. The lowest resource type is YiSi-0, which computes word similarities simply by measuring the length of the longest common substring between two words. YiSi-0 therefore does not rely on exact matches, and can identify words which are morphologically related.

YiSi-1 improves off of YiSi-0 by using word embeddings to calculate the semantic similarity of words. As mentioned previously, a well-trained embedding model will assign semantically similar words to word embeddings which are similar as well. Therefore, YiSi-1 calculates the similarity of two words by calculating the cosine similarity between the embeddings. YiSi-1 is able to determine if two words are semantically related to each other, even if they do not share common character sequences.

YiSi-2 aims to make further improvements by calculating lexical similarity using BERT embeddings, rather than the embeddings from a simpler embedding model. As mentioned previously, a BERT embedding for a word is produced by taking the word's context into account. This allows words with multiple senses, such as "bank", to be assigned different embeddings depending on the sentences which contain them. Provided that BERT embeddings are available for the given language, YiSi-2 represents the maximal measure of semantic similarity between an output and reference translation.

Lo (2019) evaluated YiSi scores according to human evaluations and BLEU scores between English and a variety of languages (Czech, German, Estonian, Finnish, Russian, Turkish, Mandarin Chinese) in both translation directions. While BLEU scores achieved correlations with human judgements at around 0.97 across all translation directions, certain languages, such as Turkish and Chinese, had much lower correlations. YiSi-0 on the other hand correlated much more stably with human judgements at around 0.98, regardless of the language. YiSi-0 scores did not always produce an increased correlation when compared to BLEU scores, but were nonetheless comparable to BLEU scores in terms of correlation with human evaluations. YiSi-1 was found to be the metric which correlated the best across all translation directions, showing especially marked improvement for Turkish. This improvement in correlation with Turkish may suggest that YiSi scores are more adept at evaluating translations involving agglutinating languages. Therefore, YiSi-0 scores will be measured along with BLEU scores when evaluating the MT systems for this project.

Chapter 3

Experiments and Results

This chapter discusses related work towards the task of low-resource NMT, describes the experiments carried out in this study, and the findings of these experiments.

3.1 Low-Resource NMT (Related Work)

Although neural machine translation has become the dominant paradigm in machine translation, the quality of NMT depends on the size and quality of available training data. Most NMT systems have been developed for what are called high-resource languages, for which there are large amounts of potential training data. However, when large datasets are not available NMT systems typically suffer, and often are not able to outperform statistical MT systems (Koehn and Knowles, 2017). Therefore the default NMT frameworks which have been developed are not suitable for the translation of low-resource languages, which include many of the world's indigenous languages.

A variety of approaches have been developed towards solving the problem of low-resource NMT. Generally speaking, these approaches attempt to find ways of increasing the size of the available training data, or tuning the neural network to better converge to a solution. Such approaches will be discussed in this section, and will be applied to the problem of producing an NMT system for Yupik.

3.1.1 Translating Inuktitut

In his PhD thesis proposal (Micher, 2018), Jeffrey Micher of Carnegie Mellon University addresses the difficulties of producing a machine translation system for Inuktitut, an Inuit language spoken in the Canadian province of Nunavut. Inuktitut is a language which is closely related to Yupik, and therefore Micher's findings have proven to be quite helpful to develop a translation system for Yupik.

The first problem of Inuktitut is that it is a low-resource language. Inuktitut is spoken by a small population of around 40,000 speakers, and consequently the amount of digital materials to use as training data is quite limited. However, the Parliament of Nunavut publishes their proceedings in both English and Inuktitut, which is ideal to use as training data for a machine translation system. Although the only major dataset available for Inuktitut is in the parliamentary domain, the size of the Nunavut Hansards is large enough to produce NMT system of decent quality.

The second issue is that Inuktitut is part of a dialect continuum which spans from the Arctic coast of the Yukon territory to the far north of Quebec, and therefore has many varying dialects. The dialect used in the Nunavut Hansards is likely the dialect spoken in Iqaluit, the capital city of Nunavut. However, a machine translation system using the Hansards as training data may not be able to produce translations for the easter dialect of Labrador Inuktitut, or the western dialect of Inuvialuktun.

Along with dialect variations, there are many irregularities in Inuktitut orthography. Many written materials in Inuktitut use a Latin-based orthography which was initially developed by missionaries, and standardized in 1976. As this orthography was developed primarily by nonnative speakers, it occasionally contains elements which are not intuitive to native Inuktitut speakers. Consequently, there is wide variety in the spelling of many Inuktitut words (Mallon, 1999). Micher even found that the Inuktitut word for “inmate” was spelled three different ways in the same sentence (Micher, 2018).

The last challenging aspect of Inuktitut is its complex morphology. Like Yupik, Inuktitut is able to form words out of long sequences of morphemes, each of which may surface differently according to Inuktitut’s complex morphophonology. Therefore, a translation system must find some way to parse this complex morphology.

Despite these limitations, Micher was able to train translation models capable of decent quality translation (roughly 30 BLEU for Inuktitut $\rightarrow$ English). Micher evaluated the BLEU scores for Inuktitut using the m-BLEU variant of the traditional algorithm, which calculates a BLEU score over segmented morphemes, rather than whole words. Although materials written in Inuktitut are rare, the Hansards are large enough (about 340,000 sentences) to produce a decent NMT model. However, Micher also noticed that the translation quality for English $\rightarrow$ Inuktitut was much lower (less than 20 BLEU). There are two explanations Micher proposes for this: the first is that the BLEU score is not suitable for evaluating the quality of machine-generated polysynthetic languages such as Inuktitut, and therefore other evaluation metrics should be used. The second is that the BLEU scores are correct, and indicate that it is more difficult for NMT models to generate sentences in polysynthetic languages than it is to generate more analytic languages like English. To determine which of these is the case, Micher plans in the near future to obtain human evaluations of translation quality from Inuktitut speakers.

The challenges which face developing a translation system for Inuktitut, namely limited training resources and domains, dialect and orthographic variations, and complex morphology, are aspects which beset many low-resource languages. The scores which Micher achieved on his model show promising results for the performance of NMT on these languages. However, languages such as Yupik do not have the good fortune of having a large corpus of available data. While the Nunavut Hansards contain about 340,000 sentences, Yupik only has around 8,000 sentences suitable to use as training data. It is therefore necessary to apply other strategies to develop a translation system which produces decent quality results for Yupik.

3.1.2 Multilingual Models

In machine learning in general, it is common to use an approach called transfer learning when there is a small amount of training data available for a particular task. Transfer learning approaches involve training a machine learning system on a large dataset which is closely related to the problem of interest, an later continuing the training process on the smaller dataset, through a process called "fine-tuning". The intuition behind transfer learning is that the parameters developed for the larger dataset will be quite similar to the parameters needed for the smaller dataset. It is also assumed that providing the network with the larger dataset, although it is not in the exact domain, is more helpful than no additional data at all. In the context of machine translation, transfer-based approaches involve training a translation model between a language pair for which there are large amounts of bilingual data, and later fine-tuning that model to a different language pair of interest.

Along with transfer learning, machine translation approaches have been developed which use more than two languages in their training data. Typically, this involves adding a special token to each sentence, indicating to the model which language each sentence belongs to (Johnson et al., 2017). These tokens can then be used by the translation architecture to generate the correct language. Such systems are called multilingual models.

Zoph et al. (2016) present a transfer learning approach to low-resource NMT. They began by training a "parent" model on a high-resource language pair (in this case French-English). They then copy these model parameters to "child" models, which were fine tuned to translate Hausa, Turkish, Uzbek, and Urdu. Zoph et al. find performance gains of 5.6 BLEU on average when parameters are transferred, and note that the effect is more pronounced for Urdu (+8.6 BLEU), for which the corpus was only 4,000 sentences in length. They found that allowing all parameters in the child model to be trained, except for the target language word embeddings, produced the optimal results. Zoph et al.'s approach is a simple but effective approach to low-resource NMT.

The approach of Gu et al. (2018) was a more direct approach to attempt low-resource NMT. Gu et al.'s approach was to train word embeddings on multiple languages, then transform these embeddings into a single Universal Lexical Representation (ULR) space. They also added a Mixture of Language Experts (MoLE) component to their network topology to select between multiple source languages when producing a translation. Gu et al. found that near state-of-the-art performance on English-Romanian translation could be achieved using a parallel corpus of only 6,000 sentences. However, they note that their ULR and MoLE components work when the target language is related to the training languages, whereas unrelated languages suffer in translation accuracy.

Imankulova et al. (2019) note that a frequent problem in low-resource NMT is that corpora are not available in a wide variety of domains. They suggest to solve this problem using a three step approach: training high-resource language pairs on out-of-domain data, fine-tuning these models to in-domain data on the same high-resource language pairs, and finally fine-tuning these models to the in-domain data in the low resource language pair. Imankulova et al. focus in particular on training sentences from the news domain from Russian to Japanese, using out-of-domain corpora for Russian-English and Japanese-English, as well as in domain data for all three language pairs. They found that using all three of these steps produces models with higher performance than models trained on any of two steps alone, and find an improvement in 3.7 BLEU over a strong baseline. Additionally, they found that further increases in performance could be made through the use of back-translation Sennrich et al. (2016a). Additionally, Imankulova et al. found that the ideal architecture to use was a multi-lingual Transformer based architecture.

Although there are a wide array of approaches to using auxiliary data in machine translation, each approach finds the following consistent results: using auxiliary data improves translation quality in all cases, and that the use of auxiliary data is particularly helpful among related languages. Therefore, improving the performance of a Yupik $\rightarrow$ English translation system should be possible through the use of Inuktitut auxiliary data. This is one of the main research questions of this project.

3.1.3 Hyperparameter Optimization

While many NMT researchers have aimed to solve the problem of low-resource machine translation using transfer based methods, Sennrich and Zhang (2019) criticize these approaches, and argue instead that low-resource translation can be improved by selecting the appropriate hyperparameters for low-resource settings, such as a smaller BPE vocabulary, while also applying techniques such as whole-word dropout. Sennrich et al. note that the use of transfer based methods still require large amounts of auxiliary data, which requires longer training time and more electricity and carbon usage (Strubell et al., 2019). Sennrich et al. perform experiments on translation from German to English and Korean to English, while varying the corpus size to determine its effect on translation quality. The sizes of these corpora range from 80k sentences to 5k sentences in the ultra-low resource condition. Sennrich et al. found significant performance gains using their hyperparameters and model architecture, in particular the low resource translation condition increased from 7.6 BLEU to 16.6 BLEU. These gains in translation quality were found for both German-English and Korean-English, suggesting that Sennrich et al.'s methods are robust across languages which differ typologically. In the ablation study, Sennrich et al. find that aggressive word-level dropout (i.e. randomly removing a certain percentage of words from the training data) provided the most gains to translation quality (+3.4 BLEU). This process of word-level dropout has interesting similarities with cloze tests, which are commonly used in psycholinguistic experiments to evaluate language learning (Taylor, 1953).

Sennrich and Zhang (2019)'s findings contradict the idea that that while NMT systems perform better in high-resource settings, SMT settings are ideal for low-resource translation (Koehn and Knowles, 2017). Through a simple selection of hyperparameters, Sennrich and Zhang (2019) demonstrated that it is possible to train NMT models which outperform SMT baselines, even without the use of auxiliary data. Therefore, another research question of this project is to determine if Sennrich and Zhang's techniques also apply to polysynthetic languages such as Yupik.

3.1.4 Backtranslation

To train an NMT system, a training dataset is required which contains a set of sentences in a source language, each of which is paired with its translation in a target language. Such datasets can be difficult to come by, even for high-resource languages. However, it is often the case that there are large amounts of monolingual data, even for low-resource languages. Sennrich et al. (2016a) aim to exploit monolingual data in machine translation through the process of backtranslation.

The following example will outline the backtranslation process, in order to produce an optimal English $\rightarrow$ Yupik using Yupik monolingual data. Initially, two translation models are trained using a bilingual corpus: one for English $\rightarrow$ Yupik, and one for Yupik $\rightarrow$ English. Once the models have been trained, monolingual Yupik data is given to the Yupik $\rightarrow$ English model to produce a set of synthetic English data, which will be labelled as English$'$. The English$'$ and monolingual Yupik data are then appended to the original bilingual corpus to produce a larger bilingual corpus, which is then used to train a third model. Although using synthetic English data in this corpus will likely suffer in quality compared to English text written by a human, it would still be more useful for a translation system compared to no data at all (Koehn, 2020, p.263-264).

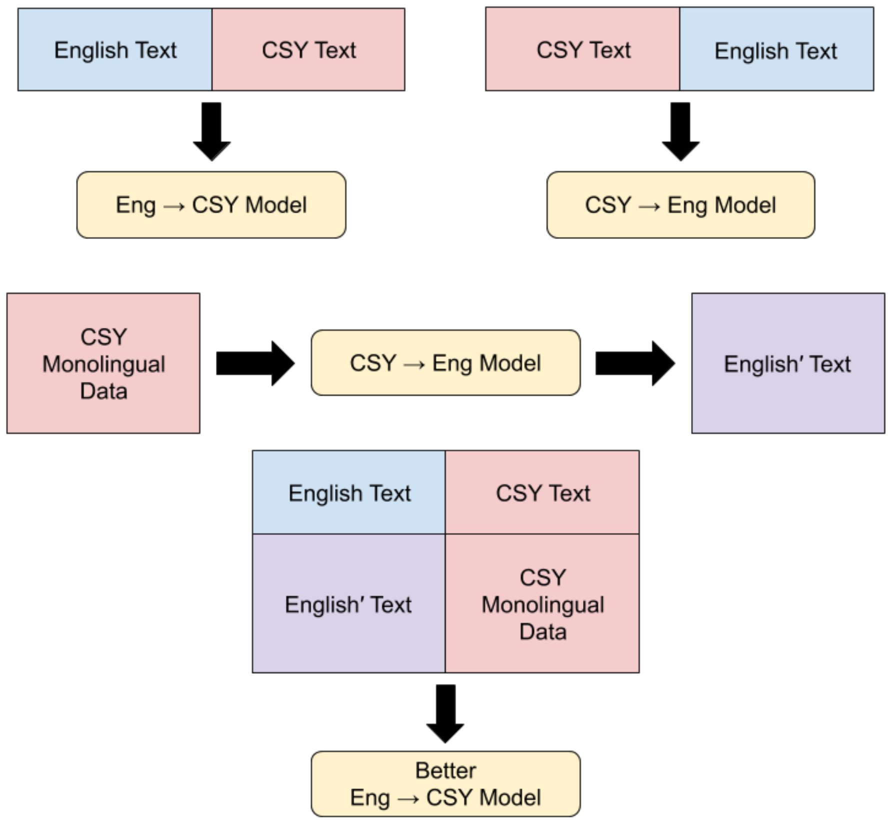


Figure 3.1: The backtranslation process (based off of a similar figure 14.1 by Koehn (2020)

This process could also be used to generate synthetic Yupik data, and thus improve the Yupik → English direction as well.

Sennrich et al. (2016a) found that using such a strategy did indeed improve translation quality as measured by BLEU scores, confirming that the use of monolingual data and its synthetic counterpart proved beneficial to the NMT model. Additionally, Imankulova et al. (2019) found in their study that repeating this process over multiple iterations can produce models which produce successively better translation performance.

Backtranslation was not implemented in this project, however it would be a good topic for future research.

3.2 Outline of Experiments

The aim of this project was to evaluate approaches for low-resource NMT on the English → Yupik and Yupik → English translation directions. An experiment was conducted which evaluated the effectiveness of using an Inuktitut as auxiliary data in Yupik translation systems. Another experiment evaluated the use of Sennrich and Zhang's hyperparameters. An additional experiment tested the effectiveness of word-level dropout, the factor which Sennrich et al. (2016a) found to be the strongest factor for increasing translation metrics. Finally, an SMT system was trained for the English → Yupik and Yupik → English translation directions to determine if the previous experiments could produce an NMT system which outperforms an SMT system despite low amounts of training data.

The quality of each translation system in each experiment was calculated using three metrics:

- BLEU, using BPE subwords as tokens
- BLEU, using whole words as tokens
- YiSi-0, using whole words as tokens

Evaluating on BPE subwords should approximate the m-BLEU scores, which Micher used in his experiments (Micher, 2018; Luong et al., 2010). BLEU scores were also calculated using whole words (for which BLEU was developed (Papineni et al., 2002)), to determine if there were conditions in the experiments which would cause these two scores to diverge.

Each of the three scores are accompanied by a 95% margin of error, which is estimated by through the process of bootstrap resampling (Koehn, 2004). Each BLEU score and its error margin were calculated using the script developed by Peris (2019)[1], which uses the multi-bleu.perl script from the Moses project (Koehn et al., 2007). Error margins for YiSi-0 scores were calculated by applying the algorithm described by Koehn (2004) to the sentence scores produced by the YiSi-0 script[2].

Each model was trained on a single NVIDIA GeForce GTX 1080 Ti GPU.

3.2.1 Datasets

These experiments make use of two datasets: the first if a bilingual corpus of Yupik and English sentences, and the second is a bilingual corpus of Inuktitut and English.

The Yupik-English corpus consists primarily of the Yupik New Testament produced by the Wycliffe Bible Translators (Yupik Translation Committee, Wycliffe Bible Translators, Inc., 2012). Each verse is separated by a newline in the file, meaning the corpus is aligned according to verses, rather than sentences. Therefore, each verse will be treated by the translation model as a single sentence, and will be referred to as a "sentence" in the rest of this paper. The verses from the New Testament are supplemented with a small number of sentences from translated books written in Yupik, to bring the total number of sentences to 8,001.

[1]https://github.com/lvapeab/confidence_intervals
[2]https://github.com/chikiulo/yisi

The Inuktitut-English corpus consists entirely of the proceedings of the parliament of Nunavut. The same corpus was used by Micher (2018) in his research on Inuktitut translation.

Information about the corpora is listed below. Each English side of the corpora contains roughly twice as many words as the Inuit sides, a result of the polysynthetic morphology of the Inuit languages. Sentences in the Yupik-English corpus are also much longer than the sentences in the Inuktitut-English corpus. This is primarily due to the fact that the Yupik-English corpus consists of New Testament verses, each of which may contain multiple sentences.

		Sentences	Tokens	Mean Sentence Length
Yupik-English:	Yupik	8,001	122,000	15.3
Yupik-English:	English	8,001	290,000	36.2
Inuktitut-English:	Inuktitut	340,526	2,180,000	6.4
Inuktitut-English:	English	340,526	4,050,000	11.9

Figure 3.2: Information on the datasets used in this project

Both of the corpora are lowercased, and are tokenized using the tokenizer from the Moses project (Koehn et al., 2007), where each token is separated by a space.

These corpora were divided into three datasets for each trained model: a training set (80%), a validation set (10%), and a test set (10%). Each sentence in the corpus was assigned randomly to one of these datasets.

3.2.2 Research Questions

The experiments in the project aim to answer the following research questions.

1. How does the addition of Inuktitut auxiliary data affect the translation quality of Yupik?
2. Does the BPE vocabulary size which Sennrich and Zhang recommend apply to a polysynthetic language (Yupik) as well?
3. How does the use of specialized hyperparameters for low-resouce NMT affect the translation quality of Yupik?
4. How does the effect of word-level dropout affect the translation of Yupik?
5. Can the suggested techniques for low-resource NMT (multilingual models and optimized hyperparameters) produce translation models which outperform an SMT baseline?

3.2.3 Experiment Overview

Five main experiments were carried out for this project:

1. **SMT Baselines:** This experiment trained SMT models for the Yupik→English and English→Yupik translation directions using the Moses system (Koehn et al., 2007). These models will be used as baselines to determine if NMT optimizations are able to exceed the performance of SMT in a low-resource setting, as Sennrich and Zhang found.
2. **Determining the Optimal BPE Vocabulary Size (NMT Baselines):** This experiment trained Yupik→English, English→Yupik, Inuktitut→English, and English→Inuktitut models using the OpenNMT toolkit with its default hyperparameters. The only variable in this study is the size of the BPE vocabulary, which is varied to determine if the optimal BPE size for a low resource setting is in agreement with Sennrich and Zhang's findings (i.e., a size of 2,000 is ideal for low-resource settings, while a size greater than 12,000 is ideal for high resource settings). These models will also be used as baselines, to determine how multilingual modelling and optimized hyperparameters affect performance vs models trained solely on bilingual data with the default hyperparameters.
3. **Multilingual Models:** This experiment trained multilingual models on the combination of the Inuktitut-English dataset and the Yupik-English dataset using the default hyperparameters. Additionally, the Yupik portion of the training data was duplicated, to determine if a more balanced multilingual corpus could produce better translation results.
4. **Applying Sennrich and Zhang's Hyperparameters:** This experiment trained Yupik→English, English→Yupik models using the hyperparameters per the recommendations of Sennrich and Zhang for low resource setting. Like the NMT baseline experiment, only the BPE vocabulary size was allowed to vary.
5. **Word-level Dropout:** This experiment trained Yupik→English, English→Yupik models with Sennrich and Zhang's hyperparameters, while also removing varying percentages of word types from the training data to approximate the effects of word-level dropout. The BPE vocabulary size used in this experiment was 2,000, per Sennrich and Zhang's recommendation.

3.2.4 Hypotheses

The following hypotheses were made towards the outcomes of this experiment:

1. Using Inuktitut as auxiliary data will improve the translation quality of Yupik→English and English→Yupik translation models.
2. The optimal BPE vocabulary size for Yupik will be 2,000 subwords, in agreement with Sennrich and Zhang's recommendations for low resource languages.
3. The optimal BPE vocabulary size for Inuktitut will be around 14,000, in agreement with Sennrich and Zhang's recommendations for high-resource languages.

4. Applying Sennrich and Zhang's recommeded hyperparameters (aside from word-level droupout) low resource NMT will improve the translation quality of Yupik→English and English→Yupik translation models.
5. Applying word-level dropout will significantly improve the translation quality of Yupik→English and English→Yupik translation models.
6. Yupik→English models in all cases will outperform the English→Yupik models, in agreement with Micher (2018)'s findings on Inuktitut translation.
7. Applying multilingual models or optimized hyperparameters will be able to produce a decent translation model for the Yupik→English or English→Yupik, where a decent model can produce a BLEU score greated than 15.
8. Applying multilingual models or optimized hyperparameters will be able to produce a translation model for the Yupik→English or English→Yupik which outperforms an SMT basline model.

3.3 Experiment 1: SMT Baseline

The purpose of this experiment is to attempt to replicate the findings from Sennrich and Zhang (2019)'s study, namely that NMT systems, with the appropriate hyperparameters, can outperform an SMT baseline. Previous research (Koehn and Knowles, 2017) has found that in low-resource settings, SMT systems are superior NMT systems as measured by BLEU scores. The strategies for NMT optimization outlined in this project will be evaluated by comparing the performance of the optimized NMT models to the performance of the SMT baselines.

SMT systems were trained using Moses (Koehn et al., 2007) for the Yupik→English and English→Yupik translation directions. GIZA++ (Och and Ney, 2003) was used to train word alignments, and a 3-gram language model was trained with lmplz (Heafield et al., 2013). Feature weights were optimized on the validation set using Minimum Error Rate Training (Och, 2003)[3]. Like Sennrich and Zhang (2019)'s experiments, language models were not trained with supplemental monolingual data[4].

The datasets used for this experiment (train, validation, test) are identical to the datasets used in the experiment testing optimized hyperparameters. This was done to mitigate the effects of bias in the datasets[5]. The validation set is used to optimize the feature parameters, as mentioned above.

[3]The training process for these SMT baselines is identical to the process outlined here, replacing the French-English corpus with the Yupik-English corpus

[4](Koehn and Knowles (2017), however, did use supplemental data for their LM, but as Sennrich and Zhang (2019)) note, the use of supplemental monolingual data is no longer an exclusive benefit of PBSMT.

[5]If two models are trained on different training sets, and tested on different test sets, a difference in performance between these two models could simply arise by chance, even if these datasets were selected from the same corpus. One model may receive training data which by chance contains less noise, or be evaluated on a test set which by chance contains simpler sentences which are easier to translate. Therefore, the best way to determine if one

3.3.1 Results of Experiment 1

The table below contains the results produced by the Moses models trained for each translation direction:

Translation Direction	**Yupik → Eng**	**Eng → Yupik**
BPE BLEU	9.33 ± 0.69	4.48 ± 0.61
Whole Word BLEU	9.31 ± 0.67	4.51 ± 0.62
YiSi-0	0.4401 ± 0.0084	0.3638 ± 0.0082

Figure 3.3: Performance of SMT baselines

The Yupik→English model significantly outperformed the English→Yupik model, in agreement with Micher (2018)'s findings.

3.4 Experiment 2: Determining the Optimal BPE Vocabulary Size (NMT Baseline)

Sennrich and Zhang (2019) found that in low-resource settings, a smaller BPE vocabulary size is optimal. In high-resource settings, a BPE vocabulary size around 15,000 is typically used. Sennrich and Zhang suggest using a size of 2,000 instead. These vocabulary sizes were determined for high-resource languages, which are typically more analytic than agglutinative languages. Because BPE subwords can encode morphological information, it could be the case that typologically different languages will also have different optimal BPE vocabulary sizes. This experiment aimed to determine if this is the case.

This experiment trained four translation models: English→Yupik, English→Inuktitut, Yupik→English, and Inuktitut→English. Each of the models was trained using the default OpenNMT hyperparameters, allowing only the BPE vocabulary size to vary.

The BPE vocabulary sizes for the English→Yupik and Yupik→English translation directions was tested in increments of 100 from 100 to 6,000 subwords. Each model was trained using the identical train, validation, and test sets. These datasets only differed in the size of the BPE vocabulariy applied to them. BPE vocabulary sizes larger than 6,000 were not tested, as these subword pairs occured with frequency 1, and therefore would have harmed translation quality (Sennrich et al., 2016b). The Yupik models were validated every 200 training steps, with early-stopping after ten penalties.

The BPE vocabulary sizes for the English→Inuktitut and Inuktitut→English translation directions was tested in increments of 1,000 from 1,000 to 13,000 subwords. As with the Yupik models, each of these models was trained with the

model architecture outperforms another is to train and evaluate the models with as similar conditions as possible.

same train, validation, and test set. The Inuktitut models were validated every 5,000 training steps, with early-stopping after ten penalties.

Fewer BPE vocabulary sizes were trained for the Inuktitut models compared to the Yupik models due to time constrains. The time needed to train an NMT model depends on the size of the training data. For this reason, Yupik models would complete the training process in about 5 minutes, whereas the Inuktitut models would finish training after two hours.

The models were evaluated with BLEU scores applied on BPE subwords, BLEU scores applied to whole words, and YiSi-0 scores.

3.4.1 Results of Experiment 2

The results of varying the BPE sizes are shown in the graphs below.

Due to the small scores for the Yupik translation models, there is a high amount of noise in these BLEU scores compared to the scores of the Inuktitut models.

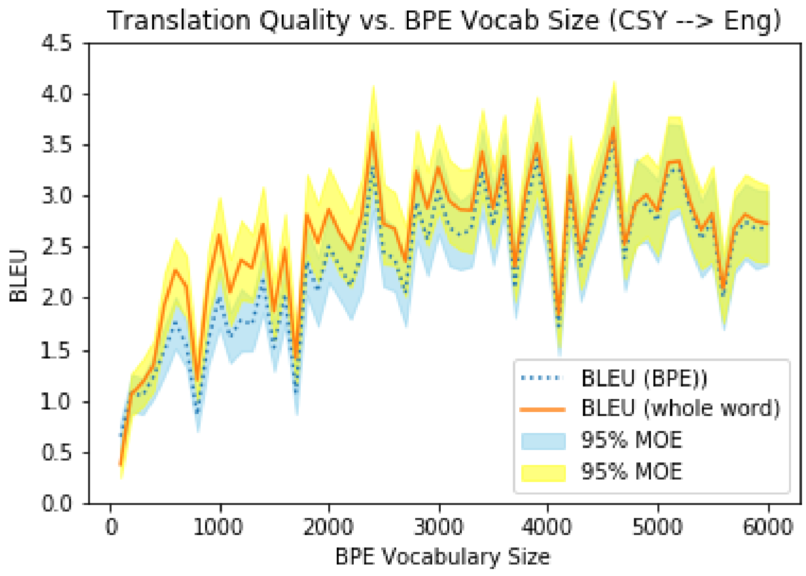


Figure 3.4: Yupik→English translation quality

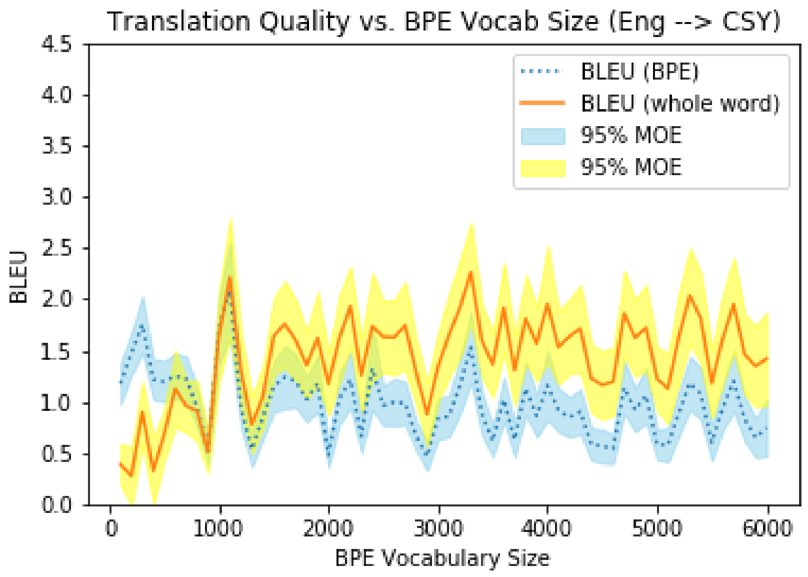


Figure 3.5: English→Yupik translation quality

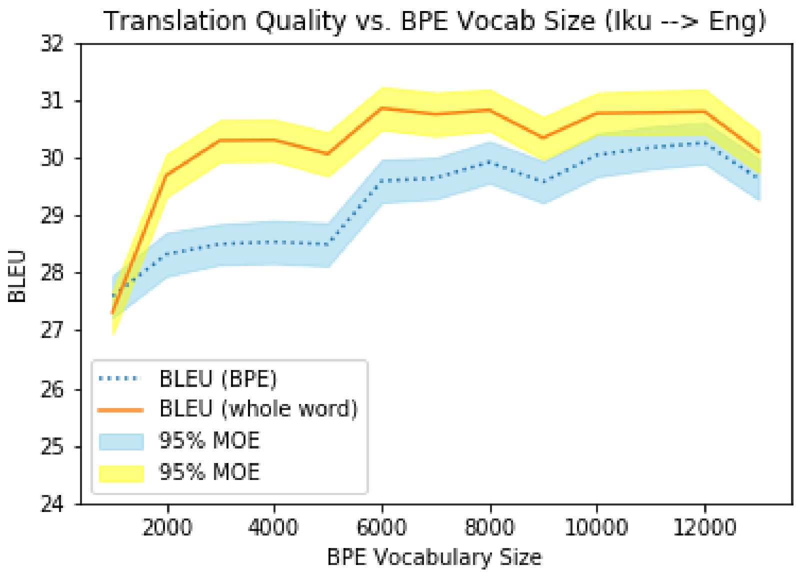


Figure 3.6: Inuktitut→English translation quality

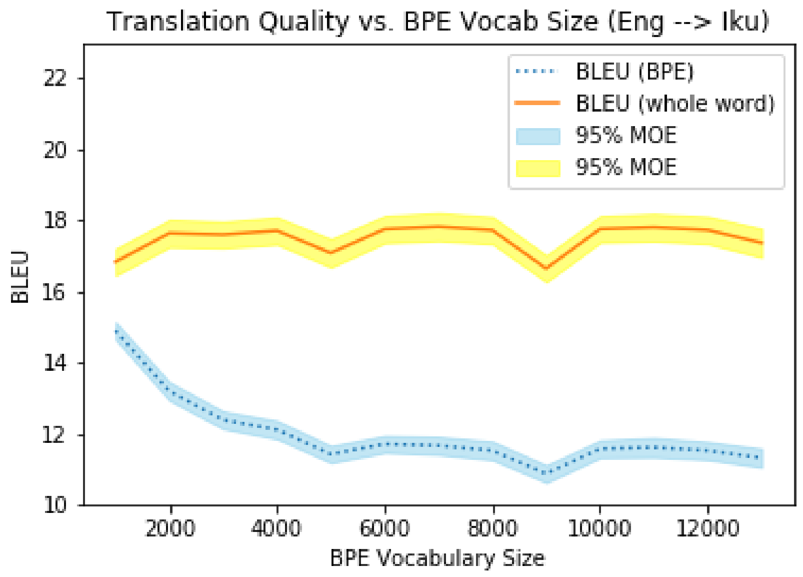


Figure 3.7: English→Inuktitut translation quality

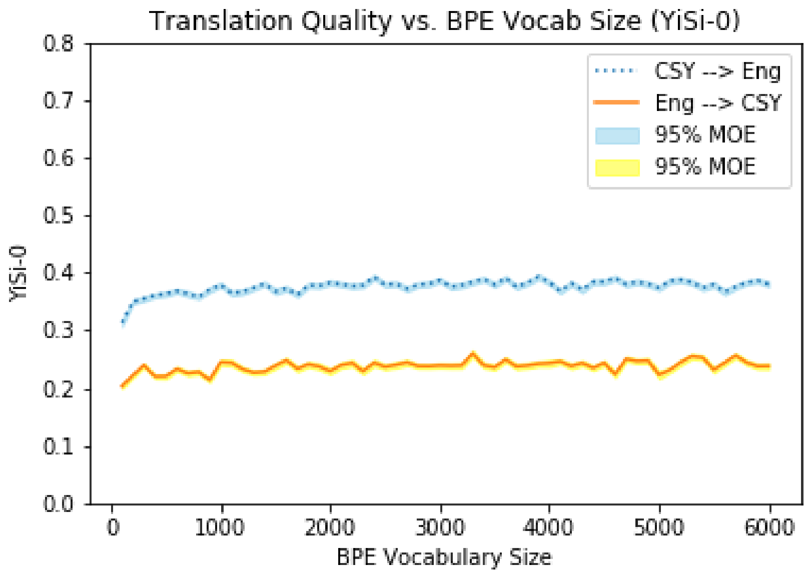


Figure 3.8: Translation quality for Yupik as measured by YiSi-0

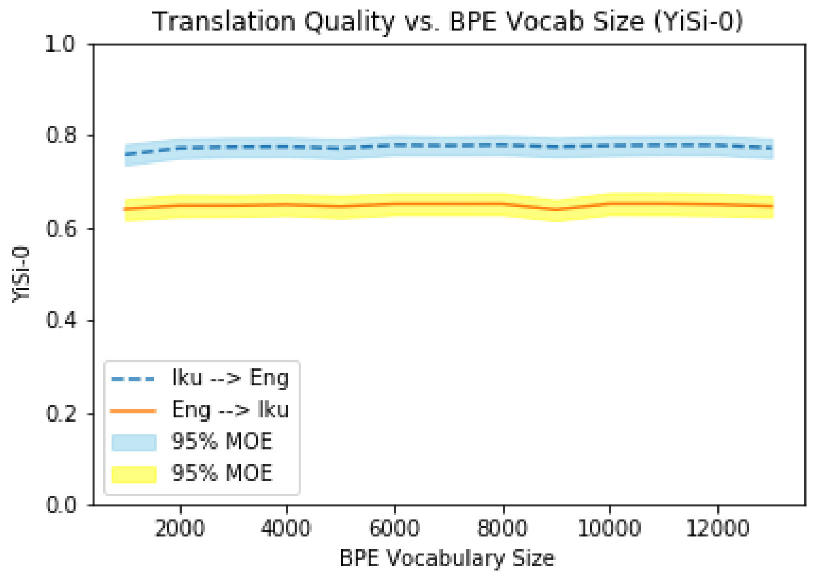


Figure 3.9: Translation quality for Inuktitut as measured by YiSi-0

The graphs comparing BPE BLEU scores with whole word BLEU scores show that these two scores are fairly consistent with each other, with the whole word BLEU scores being slightly higher than the BPE BLEU scores. However, the BPE BLEU scores perform better when the BPE size is very small. This should be expected, because a smaller BPE size means that it is more likely for the output tokens to match the reference tokens by chance. Therefore, results for the very small BPE vocabulary sizes should be taken with caution.

The table below contains the highest performing models from this experiment, along with the BPE vocabulary size they used. As predicted, the Yupik→English and Inuktitut→English models outperformed the English→Yupik and English→Inuktitut models. The Inuktitut models also drastically outperformed the Yupik models, likely due to the amounts of training data available for these languages. Note however, that many well performing models used BPE vocabulary sizes which are very different from the size which produced the optimal model, as is apparent in the graph. Therefore, these figures may not give the best indication of the relationship between BLEU score and vocabulary size.

The table below lists the BLEU and YiSi-0 scores for the best performing models for each translation direction, along with the BPE vocabulary size it used. These scores will be used to used as a baseline to which the optimized NMT models in the following experiments will be compared to. If an optimized NMT system for a particular translation direction outperforms the best performing model for the same direction, we can conclude that this optimized system will also outperform any ONMT model with the default hyperparameters, regardless of the BPE size it uses.

Translation Direction	**Inuktitut → English**	**Yupik → Eng**	**Englist → Inuktitut**	**English → Yupik**
BPE BLEU	30.25 ± 0.36	3.56 ± 0.44	14.89 ± 0.25	2.09 ± 0.47
BPE Vocab Size	12,000	4,600	1,000	1,100
Whole Word BLEU	30.85 ± 0.37	3.66 ± 0.46	17.82 ± 0.39	2.26 ± 0.48
BPE Vocab Size	6,000	4,600	7,000	3,300
YiSi-0	0.778 ± 0.0206	0.3932 ± 0.0047	0.6512 ± 0.022	0.2601 ± 0.0059
BPE Vocab Size	8,000	3,900	1,000	3,300

Figure 3.10: Best performing models and BPE size

The finding that the peak model used a BPE vocabulary size of 4,600 seems to be in conflict with Sennrich and Zhang's recommended vocabulary size of 2,000. However, the findings of this experiment show that those differences are small. Additionally, Sennrich and Zhang have found that it is better to use a smaller BPE vocabulary size rather than a larger one, as smaller BPE vocabulary sizes will uncover more phonological and morphological information in its subwords. This kind of information may not lead to increased quality in terms of BLEU scores, however it may allow it to more accurately translate low-frequency words, as BPE was designed to do. For these reasons, Sennrich and Zhang's recommended BPE vocabulary size (2,000) was used for Yupik translation models in all of the other experiments.

3.5 Experiment 3: Multilingual Models

This experiment involved training a model using both Yupik-English data and Inuktitut-English data, with the aim of determining the effect of auxiliary data on the translation quality of the English→Yupik and Yupik→English translation directions.

The multilingual models were trained using a corpus which combined the Yupik-English and Inuktitut-English corpora. Special tokens indicating the source and target language were appended to the source sentences, in an approach similar to that of Johnson et al. (2017). Based on the presence of these tokens, the model should learn to produce its output in the target language matching the special token, although the target side of its corpus contains sentences in multiple languages. An example of these tokens is shown in the figure below:

Source: `_src_eng_ _tgt_Yupik_ and they said unto him, in bethlehem of judaea : for thus it is written by the prophet`
Target: `kiimsiqaat , bethlehem-emi judea-melngughmi . uuknaliqistem whaten igaqegkaqaa :`

Figure 3.11: Example sentences in the multilingual corpus

The example above shows a sentence used for the English→Yupik translation direction. Note that for the Yupik→English direction, the tokens `_src_Yupik_` `_tgt_eng` are used instead, and they are appended to the Yupik sentence rather than the English sentence.

The Yupik portion of the corpus is broken into BPE subwords using a vocabulary size of 2,000, applying Sennrich and Zhang's recommendation. The English and Inuktitut portions apply a BPE vocabulary size of 12,000, which was found to be optimal in Experiment 2. The BPE tokenization is applied before adding the language tokens to the beginning of the source sentences. The OpenNMT default hyperparameters were used for this experiment as opposed to Sennrich and Zhang's hyperparameters optimized for low resource settings. This was done for two reasons: firstly, to determine the effect that a multilingual model alone on translation quality, holding all other variables constant. Secondly, the ONMT hyperparameters were developed to produce state-of-the-art results for high-resource language pairs (Klein et al., 2017), and should not need the optimizations developed by Sennrich and Zhang.

Two multilingual models were trained, one for Inuit→English and one for English→Inuit. Like the Inuktitut→English and English→Inuktitut models, the multilingual models were validated every 5,000 training steps, with 10 early-stopping penalties.

The test sets of the multilingual corpus were divided into a Yupik test set and an Inuktitut test set, so the multilingual models could be evaluated on their ability to translate the languages individually. BLEU and YiSi-0 scores were calculated for the Inuit→English model according to its ability to translate Yupik→English and Inuktitut→English, and similarly the English→Inuit model is evaluated for its ability to translate English→Yupik and English→Inuktitut.

Since the Yupik portion only makes up 2.3% of the multilingual corpus, another subexperiment was carried out, where the amount of Yupik data in the multilingual corpus was duplicated. This experiment aimed to determine whether a multilingual corpus with a balanced number of sentences for each language could improve translation results. To accomplish this, five new multilingual training sets were created with different amounts of Yupik duplication, namely ×2, ×5, ×10, ×20, and ×50. With Yupik duplicated in the corpus 50 times, the amount of Yupik sentences would roughly equal the amount of Inuktitut sentences. Duplication was applied only to the training set to ensure that the same Yupik sentence would not by chance appear in both the training set and the test set, which would lead to an inaccurate measure of translation quality.

Five multilingual models were trained both for the Inuit→English and English→Inuit directions, and were each evaluated according to their ability to translate the Yupik and Inuktitut as described above.

3.5.1 Results of Experiment 3

The multilingual models were evaluated according to their ability to translate each language direction. The results of the multilingual models and their performances are listed below. These models are compared to the best performing bilingual models, determined in the previous experiment (i.e. the NMT baselines).

Translation Direction	**Inuktitut → English**	**Yupik → Eng**	**Englist → Inuktitut**	**English → Yupik**
		BPE BLEU		
Multilingual	**30.86 ± 0.38**	**6.93 ± 0.58**	11.21 ± 0.26	**2.55 ± 0.38**
Bilingual	*30.25 ± 0.36*	*3.56 ± 0.44*	***14.89 ± 0.25***	*2.09 ± 0.47*
		Whole Word BLEU		
Multilingual	**31.2 ± 0.38**	**7.34 ± 0.61**	17.43 ± 0.37	**3.37 ± 0.57**
Bilingual	*30.85 ± 0.37*	*3.66 ± 0.46*	***17.82 ± 0.39***	*2.26 ± 0.48*
		YiSi-0		
Multilingual	0.7746 ± 0.0204	**0.4616 ± 0.0074**	0.6373 ± 0.0241	**0.3124 ± 0.0084**
Bilingual	***0.778 ± 0.0206***	*0.3932 ± 0.0047*	***0.6512 ± 0.022***	*0.2601 ± 0.0059*

Figure 3.12: Multilingual model performance (*vs. best performing bilingual systems with default hyperparameters*)

As hypothesized, the application of multilingual modelling to Yupik translation did increase performance for the Yupik translation systems, particularly for the Yupik→English translation direction. The high quality of the Inuktitut model was likely a key factor for this improvement.

Interestingly, the performance increased for the Inuktitut→English translation direction as well. This could be because there is linguistic information in the Yupik sentences which helped the model learn patterns in Inuktitut, as these languages are closely related. However, the increase in performance is very slight, and should not be considered significant. At any rate, it was not the aim of this experiment to improve the translation of Inuktitut.

Unlike the other translation directions, the English→Inuktitut direction performance was higher for the bilingual models rather than the multilingual models. There are many reasons which could explain this phenomenon:

- It may be difficult to generate a language with orthographic variations such as Inuktitut

- Multilingual models typically use multiple source languages, rather than multiple target languages.
- The BPE vocabulary size for high resource languages was used (14,000), rather than the optimal size of 1,000 as found in Experiment 2.

Additionally, it is interesting that there is a large difference in performance with respect to BPE BLEU (3.6) between the bilingual and multilingual models, whereas the difference as measured by whole word BLEU was much smaller (-0.4 BLEU). This suggests that the BPE vocabulary used by the multilingual model is being disturbed by the addition of Yupik data. This effect may not have occured if a single BPE vocabulary was trained for the entire Inuktitut side, but further research is needed to determine if this is the case. Additionally, as will be mentioned later, these score discrepancies may not correlate with human translation judgements.

The table below shows the results of duplicating the Yupik-English portion in the training data. If a model acheived a BLEU score higher than the best performing bilingual model, its score is indicated in bold. Models which outperformed the SMT baseline are indicated with asterisks.

Translation Direction	**Inuktitut → English**	**Yupik → Eng**	**Englist → Inuktitut**	**English → Yupik**
BPE BLEU				
NMT Baseline	**30.25 ± 0.36**	**3.56 ± 0.44**	**14.89 ± 0.25**	**2.09 ± 0.47**
SMT Baseline	**N/A**	**9.33 ± 0.69**	**N/A**	**4.48 ± 0.61**
Yupik×1	**30.86 ± 0.38**	**6.93 ± 0.58**	11.21 ± 0.26	**2.55 ± 0.38**
Yupik×2	**30.65 ± 0.37**	**7.14 ± 0.41**	11.24 ± 0.25	***4.63 ± 0.40**
Yupik×5	29.23 ± 0.38	**7.05 ± 0.28**	11.05 ± 0.25	***5.37 ± 0.31**
Yupik×10	26.83 ± 0.34	**6.23 ± 0.18**	9.75 ± 0.23	**3.09 ± 0.14**
Yupik×20	21.31 ± 0.34	2.96 ± 0.10	7.49 ± 0.23	**2.91 ± 0.12**
Yupik×50	15.22 ± 0.27	2.07 ± 0.05	7.33 ± 0.18	1.24 ± 0.04
Whole Word BLEU				
NMT Baseline	**30.85 ± 0.37**	**3.66 ± 0.46**	**17.82 ± 0.39**	**2.26 ± 0.48**
SMT Baseline	**N/A**	**9.31 ± 0.67**	**N/A**	**4.51 ± 0.62**
Yupik×1	**31.20 ± 0.38**	**7.34 ± 0.61**	17.43 ± 0.37	**3.37 ± 0.57**
Yupik×2	**31.03 ± 0.39**	**7.48 ± 0.42**	17.46 ± 0.39	***4.71 ± 0.52**
Yupik×5	29.60 ± 0.36	**7.33 ± 0.31**	17.40 ± 0.38	***5.14 ± 0.39**
Yupik×10	27.16 ± 0.34	**6.50 ± 0.19**	15.95 ± 0.34	**3.46 ± 0.18**
Yupik×20	21.68 ± 0.36	3.13 ± 0.10	15.32 ± 0.38	**3.39 ± 0.14**
Yupik×50	15.31 ± 0.29	2.15 ± 0.05	13.00 ± 0.30	1.68 ± 0.05
YiSi-0				
NMT Baseline	**0.778 ± 0.0206**	**0.3932 ± 0.0047**	**0.6512 ± 0.022**	**0.2601 ± 0.0059**
SMT Baseline	**N/A**	**0.4401 ± 0.0084**	**N/A**	**0.3638 ± 0.0082**
Yupik×1	0.7746 ± 0.0204	***0.4616 ± 0.0074**	0.6373 ± 0.0241	**0.3124 ± 0.0084**
Yupik×2	0.7744 ± 0.0212	***0.4692 ± 0.0080**	0.6393 ± 0.0227	**0.3401 ± 0.0086**
Yupik×5	0.7663 ± 0.0206	***0.4601 ± 0.0080**	0.6349 ± 0.0231	**0.3422 ± 0.0094**
Yupik×10	0.7508 ± 0.0216	***0.4433 ± 0.0071**	0.6219 ± 0.0239	**0.3044 ± 0.0082**
Yupik×20	0.7130 ± 0.0237	0.3926 ± 0.0065	0.5725 ± 0.0272	**0.2908 ± 0.0082**
Yupik×50	0.6492 ± 0.0270	0.3671 ± 0.0051	0.5612 ± 0.0263	0.2463 ± 0.0061

Figure 3.13: Duplicating the Yupik Portion of the Multilingual Corpus

Duplicating the Yupik portion of the corpus did not improve the Inuktitut translation models (aside from an extremely slight gain of 0.03 BLEU and 0.002 YiSi-0 for English→Inuktitut). However, increases were made for the Yupik models, especially for the English→Yupik translation direction.

Increases in translation quality were found when Yupik data were duplicated a small number of times (2 - 5). Duplicating the Yupik sentences to the degree that it roughly matches the quantity of the Inuktitut sentences had a detrimental effect on translation quantity, with the BLEU scores being lower, or in some cases much lower, than the baselines.

With duplicated Yupik data, the Yupik translation models were able to outperform the SMT baselines, however the metrics show mixed results. The multilingual Yupik→English models outperformed the SMT baseline, but only in its YiSi-0 score, while the English→Yupik models outperformed the SMT baseline, but only in its BLEU scores. A deeper, more qualitative analysis is needed to determine why this is the case. Although, these results demonstrate the benefit of using multilingual auxiliary data over bilingual data, the results do not strongly exceed the performance of the SMT baselines.

3.6 Experiment 4: Applying Sennrich and Zhang's Hyperparameters

This experiment aimed to compare Sennrich and Zhang's recommended hyperparameters on the Yupik→English and English→Yupik translation directions with the OpenNMT default hyperparameters. The BPE vocabulary size is allowed to vary as it was for the previous experiment, to determine if the optimal BPE is different if different hyperparameters are used. Sennrich and Zhang also found that word-level dropout had the most significant effect on translation quality, so word-level dropout will be tested in a separate experiment.

The ONMT default hyperparameters and Sennrich and Zhang's hyperparameters are listed in the table below:

Hyper-parameter	OpenNMT Default	Sennrich & Zhang
Dropout	0.3	0.5
Shared embeddings	No	Yes
Encoder type	RNN	BiRNN
Subword Vocab Size (BPE)	14k	2k
Batch size	64 sentences	1,000 tokens
Learning Rate	0.001	0.0005
Optimizer	SGD	Adam
Encoder layers	2	1
Decoder layes	2	1
Label smoothing	0	0.2

Figure 3.14: Sennrich and Zhang (2019)'s hyperparameters vs the ONMT defaults (Klein et al., 2017)

This experiment trained two models: a Yupik→English and English→Yupik model with the optimized hyperparameters. Each of these models was tested on BPE vocabulary sizes in increments of 500 from 1,000 to 4,000 subwords. These models were validated every 200 training steps, with early-stopping after ten penalties. These models will be compared to the models from Experiment 2 (i.e. the bilingual ONMT models with default hyperparameters) in terms of BLEU and YiSi-0 scores.

3.6.1 Results of Experiment 4

Experiment 4 evaluated Sennrich and Zhang's hyperparameters with the exception of word-level dropout, while allowing the BPE vocabulary size to vary. The BPE vocabulary sizes tested ranged from 1,000 to 4,000 in increments of 500. The results of these experiments are shown in the graphs below.

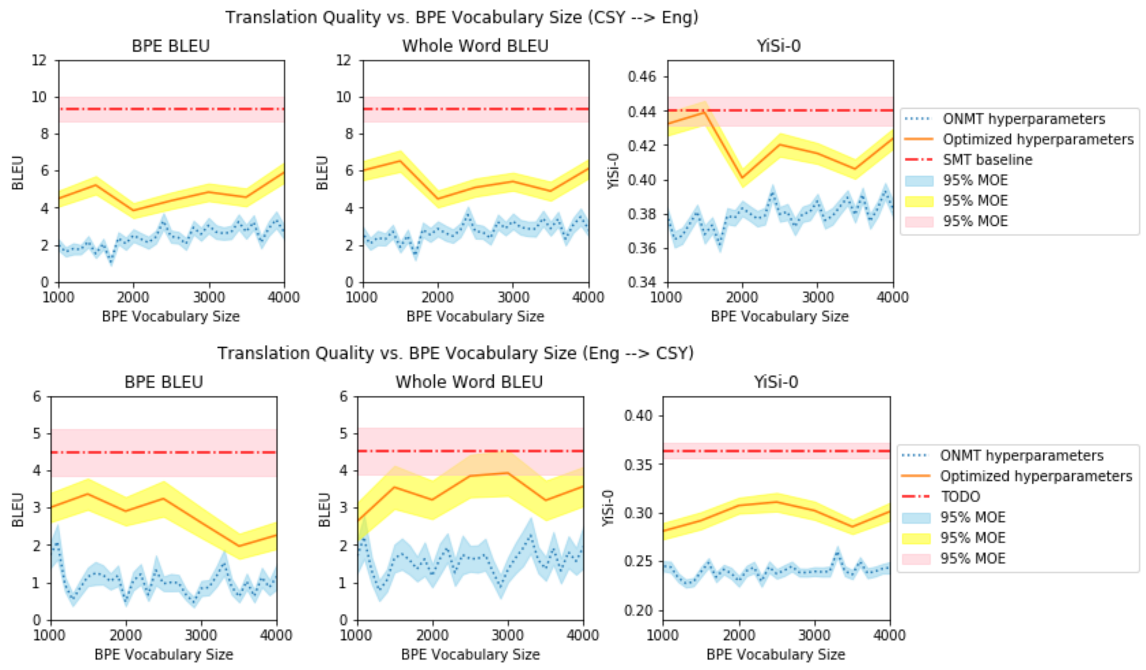


Figure 3.15: ONMT default hyperparameters vs. Sennrich and Zhang's hyperparameters

Translation Direction	**Yupik → Eng**	**Eng → Yupik**
	BPE BLEU	
Optimized	5.88 ± 0.55	3.37 ± 0.42
Default	*3.56 ± 0.44*	*2.09 ± 0.47*
SMT Baseline	**9.33 ± 0.69**	**4.48 ± 0.61**
	Whole Word BLEU	
Optimized	6.52 ± 0.58	3.93 ± 0.61
Default	*3.66 ± 0.46*	*2.26 ± 0.48*
SMT Baseline	**9.31 ± 0.67**	**4.51 ± 0.62**
	YiSi-0	
Optimized	0.4390 ± 0.0069	0.3108 ± 0.0096
Default	*0.3932 ± 0.0047*	*0.2601 ± 0.0059*
SMT Baseline	**0.4401 ± 0.0084**	0.**3638 ± 0.0082**

Figure 3.16: Best performing models (BLEU)

These graphs clearly demonstrate that applying Sennrich and Zhang's recommended hyperparameters improve the translation quality of Yupik→English and English→Yupik models. They also demonstrate that the benefit of these optimized hyperparameters is not contingent on BPE vocabulary size; there is no BPE vocabulary size for which the models with the default hyperparameters outperform the optimized models. These findings confirm the hypothesis that Sennrich and Zhang's recommendation would improve Yupik translation.

The table above lists the best performing optimized models for each translation direction from the graphs above. The optimized models are compared with the models using the default hyperparameters, and the SMT baselines. In each translation direction and across all metrics, the SMT baselines were the best performing models. However, this finding not very significant in the English→Yupik direction as measured by whole word BLEU, and is not significant when measured by YiSi-0 for the Yupik→English direction.

The findings of this experiment therefore show that Sennrich and Zhang's optimizations (as applied in this experiment) can produce NMT models which are competitive with, but do not surpass, SMT models, when the languages in question are Yupik and English.

3.7 Experiment 5: Word-level Dropout

This experiment will evaluate the quality of Yupik→English and English→Yupik models using Sennrich and Zhang's hyperparameters, while allowing the word-dropout percentage to vary.

Typically, dropout in a neural network is implemented in the programming of a neural network. However, because of time constraints, this approach was not possible for this project. Instead, word-level dropout was implemented by performing the analagous transformation to the training data, to simulate the effect of performing dropout in the code.

Word-level dropout is implemented by randomly selecting a certain percentage p of the word embeddings, and setting them to zero for each training step. This is equivalent to selection p percent of the words of the systems vocabulary at each training step, and removing those words in the sentence at that training step, should they occur in that sentence.

Given that the dropout process can be implemented in this way, separate corpora were generated by applying word-level droupout at varying percentages. The process for a given file occurred as follows:

1. Process the entire file to determine the set of words which it contains (i.e., determine the vocabulry represented by this file)
2. For each line (i.e. sentence) of the file:
   (a) Randomly select p percent of the words in the vocabulary as the current dropout set d
   (b) For each word w:
      i. if w is in d, replace w with the dropout token (`_dropout_`)

Figure 3.17: The process for generating a dropout corpus

Note that the above process applies dropout to word types, rather than word tokens. This means that if the word “the” is selected to be dropped out for a particular sentence, every instance of “the” in the sentence will be replaced with `_dropout_`. Also, each word type is selected with equal probability from the vocabulary without replacement, which means low frequency words are more likely to be dropped out compared to high frequency words[6].

The process above was applied to the Yupik-English corpus using dropout percentages from 0% to 50% in increments of 2%. The dropout was applied only to the training data, with no words being removed from the validation or test sets. The dropout was applied after the BPE tokenization, which used a vocabulary size of 2,000 per Sennrich and Zhang’s recommendation. For each of these dropout corpora, a Yupik$\rightarrow$English and English$\rightarrow$Yupik model were trained using Sennrich and Zhang’s hyperparameters[7]. Each of these models were evaluated according to BLEU and YiSi-0 scores.

[6]Given that natural language text follows a Zipf distribution, a large majority of the word types will be low-frequency words (Jurafsky and Martin, 2008)

[7]Sennrich and Zhang’s experiments evaluated word-level dropout by holding all other hyperparameters constant. This experiment will replicate this setup by using the optimized hyperparameters listed in the table above, varying only the word-level dropout

3.7.1 Results of Experiment 5

This experiment aimed to determine if applying word-level dropout would further improve the Yupik translation systems with respect to the models trained with the other optimized hyperparameters. The dropout percentage varied between 0% and 50% in 2% increments.

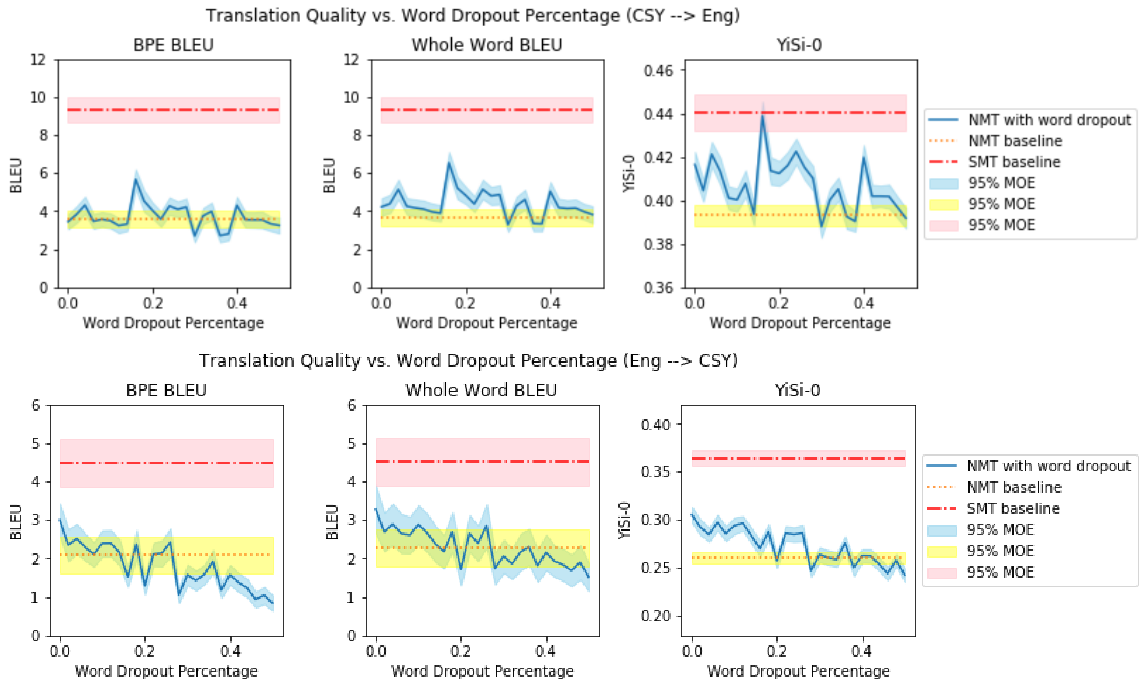


Figure 3.18: Varying the word-level dropout percentage

Like the other graphs containing the scores of the Yupik models, this graph is fraught with noise. This noise is compounded by the fact that the words which are dropped out are chosen randomly, and therefore some sentences may have words removed which are more important for the translation, while others have less significant words removed. This discrepancy in the quality of sentences produced by this randomness adds to the noisiness of this data.

Nonetheless, improvements in the translation quality were found when word-level dropout was applied. With the word-level dropout percentage of 16%, the Yupik→English system achieved a BPE BLEU score of 5.68±0.52 , a whole word BLEU score of 6.54±0.57 and a YiSi-0 score of 0.4389±0.0065 (roughly an improvement of +2 BPE BLEU, +2 whole word BLEU, and +0.05 YiSi-0). This model's performance according to YiSi-0 score is also not significantly different from the SMT baseline.

However, the translation quality of the English→Yupik models did not show the same increases in quality.

Interestingly, the translation system with extremely high levels of dropout produced translations with roughly the same BLEU and YiSi-0 scores as the system with no dropout at all. This could be an indication of the efficacy of word-level dropout, or it could be that the translation quality of the Yupik$\rightarrow$English baseline is so poor that it does not perform better than chance. Further experimentation is needed to determine if this is the case.

Chapter 4

Conclusion

4.1 Discussion

Like other low-resource NMT systems in the literature, the quality of Yupik translation suffers due to its lack of resources. Approaches to improve low-resource translation, such as multilingual models and tuned hyperparameters, have proven to be beneficial for Yupik as well. However, translation quality greater than 7.48 BLEU (whole word) for Yupik→English, and 5.37 BLEU for English→Yupik was not achieved. Therefore, the methods tested in these experiments alone are not enough to produce decent quality translations for Yupik. However, future research directions remain, which will be described in a later section.

A major problem affecting each of these experiments is the presence of noise in the BLEU scores. Because the translation quality of the Yupik models ranged from 2 to 7 BLEU, the signal-to-noise ratio for the Yupik models was very low, making it difficult to draw conclusions from the experimental results. It is hard to determine in some cases if a change in translation quality is the result of a change in parameters, or simply occured by chance. Nonetheless, some significan results were found, which will be explained in this section.

A consistent finding across all experiments was that the Yupik→English always outperformed the English→Yupik models, all else being equal. This finding was found for both types of BLEU scores and YiSi-0 scores, suggesting that this discrepancy is not specific to the BLEU score. This finding is in agreement with Micher (2018)'s findings on Inuktitut translation. It is not clear from these findings whether this discrepancy in performance is an inherent difficulty in generating a polysynthetic language, or the inefficacy of these metrics when the target language is polysynthetic. Micher's future work using human judgements of Inuktitut-English MT quality should provide very helpful information towards answering this question.

In each experiment when English was the target language, there were limited discrepancies between BLEU scores as applied to whole words, and BLEU scores applied to BPE tokens. Each whole word BLEU score is roughly 5% greater than the BPE BLEU score. However, when Inuktitut or Yupik were the target languages, the discrepancy was much larger: these whole word BLEU scores were 20-50% greater. It is not clear which of these metrics would correlate better with human judgements of translation quality, as BLEU was designed to do.

Experiment 2, which varied the BPE vocabulary size while using the ONMT default hyperparameters, confirmed Sennrich and Zhang (2019)'s findings that a smaller BPE vocabulary size is ideal for the translation of low-resource languages. This finding indicates that the BPE vocabulary size found to be optimal for analytic languages such as German applies to Yupik as well. However, this experiment also found that increasing the vocabulary size well past the recommended vocabulary size did not produce significant changes in translation quality, with the peak translation quality for Yupik→English occurring with a vocabulary size of 4,600. Sennrich and Zhang (2019) note that a smaller BPE size is preferable over a larger one, as it allows the model to learn more phonological and morphological information. Therefore, the lowest BPE vocabulary size which produces optimal translation results should be chosen. The translation quality of Yupik→English begins to peak at a vocabulary size around 2,000, which is in agreement with Sennrich and Zhang's recommendation.

Using Inuktitut auxiliary data to aid the development of Yupik translation models also proved beneficial for both the Yupik→English and English→Yupik translation directions. Additionally, duplicating the Yupik-English portion of the multilingual corpus also proved beneficial, especially for the English→Yupik direction. Both translation directions saw an improvement of roughly 3.5 BLEU with these added techniques. Many multilingual models were able to surpass the performance of the SMT baselines, however this effect was only seen in the YiSi-0 scores of the Yupik→English direction, and only in the BLEU scores (BPE and whole word) for the Englist→Yupik direction. These results suggest that the BLEU and YiSi-0 scores are picking up on different aspects of the translation, and further research is needed to determine why there is a discrepancy between these scores. However, because of these conflicting scores, and the slight differences between the scores of the best multilingual models and the SMT baselines, we can conclude that the multilingual models from this experiment are not of a significantly higher quality than the SMT baselines. However, the efficacy of duplicating the Yupik portion of the corpus, especially for translation English→Yupik, suggests that the implementation of data augmentation techniques such as backtranslation would be a fruitful topic for future research.

Applying Sennrich and Zhang's recommended hyperparameters also consitently improved the quality of translation over the OpenNMT default hyperparameters, as was hypothesized. This finding was not contingent on BPE vocabulary size. Future NMT architectures should therefore apply hyperparameters similar to these in low-resource settings. However, the optimized models are not very competitive with the SMT baselines. There were a few models which performed close to the SMT baselines: the best performing model for the English→Yupik direction acheived a whole word BLEU score of 3.93±0.61

BLEU (vs. the SMT model's 4.51±0.62), while the best performing Yupik→English model achieved a YiSi-0 score of 0.4390±0.0069 (vs. the SMT model's 0.4401±0.0084). However, these results failed to replicate Sennrich and Zhang's finding that an optimized NMT model could significantly outperform an SMT model.

In both the multilingual and optimized hyperparameter experiments, the Yupik→English models which were competitive with the SMT baselines were only competitive with respect to YiSi-0 scores, while the competitive English→Yupik models were only competitive with respect to BLEU scores. This is an interesting discrepancy which needs further investigation. It may be that these scores perform better with different aspects of translation, and it would be helpful to determine what aspects these metrics are more capable of measuring.

The word dropout experiment suffered from a high amount of noise in the data, but the findings for the Yupik→English models seem to be generally consistent with Sennrich and Zhang (2019)'s finding that word-level dropout improves MT quality. The findings for the English→Yupik models, however, do not seem consistent with this finding, at least not at a level which is significant.

Sennrich and Zhang recommended using a word-level dropout percentage of 20%, and the peak performing model in this experiment used a dropout of 16%. However, based on the level of noise in this data, it is not clear if the peak performing model is indicative of a trend, or is simply an outlier. If further experimentation confirms that an optimal word-level dropout percentage is around 20% this would make an interesting connection to BERT, which also drops out 20% of the words in its training process (Devlin et al., 2019). It may be that there is an information-theoretic property of sentences such that 20% of their words contain a ratio of meaning ideal to training language models. However, a more statistically robust experiment is required to determine if this is the case.

4.2 Conclusion

Despite the lack in certainty in much of this project's findings, several trends are clear:

- Using Inukitut as auxiliary data improves the translation quality of Yupik translation systems (confirmation of hypothesis 1).
- Applying Sennrich and Zhang's optimized hyperparameters (excluding word level dropout) improves the quality of Yupik translation systems (confirmation of hypotheses 2, 3, and 4).
- Translation quality of Yupik→English as measured by BLEU consistently exceeds that of English→Yupik (confirmation of hypothesis 6).
- The optimizations as applied in these experiments alone are not able to produce translation models which significantly outperform SMT models (rejection of hypothesis 8).

The success of these five hypotheses confirms that the methods considered in this project developed to improve low-resource NMT can also be also effective at polysynthetic languages such as Yupik. However, the word dropout experiment did not determine with significance that applying word-level dropout increases translation quality, so hypothesis 5 was not satisfied.

Despite the proven effectiveness of these approaches, the increases in translation performance were only marginal, and could not produce a machine translation system with a BLEU score greater than 7.48 (hypothesis 7 was not satisfied). Additionally, the highest performing systems in the experiments were not able to significantly outperform the SMT baselines (hypothesis 8 was not satisfied). This suggests that future work towards creating a high quality Yupik translation system should focus more attention on SMT models, rather than the NMT models which are popular today. However, there remain many methods which could lead to the development of a high-quality neural machine translation system, which will be described in the next section, and explored in future research.

The results of this project emphasize the difficulties for producing an NMT system for a low-resource polysynthetic language such as Yupik. These findings show that methods exist which can optimize NMT systems to low-resource settings, but that much more research is required to develop translation systems of decent quality for these kinds of languages. Given that most of the world's languages are low-resource languages, it is critical for NLP researchers to develop applications which work well with low amounts of resources, so the benefits provided by the latest advancements in NLP can be enjoyed equally by all peoples of the world.

4.3 Future Directions

Due to time constraints, the NMT systems evaluated in this project all used the RNN architecture. However, Lakew et al. (2018) have demonstrated that the new Transformer architecture outperforms RNN translation systems in many respects, though at the cost of increased translation time and resources. A future research project could replicate the experiments of this project to determine if better translation systems can be produced with the Transformer model.

Future research should also aim to solve the problem of translating languages which contain spelling irregularities or dialect variations. Such a system would be more robust for low-resource languages, which often do not have a single, consistent orthography (Inuktitut is such a language, Micher (2018)).

As mentioned previously, bilingual datasets are difficult to come by, especially for low-resource languages. However, even if bilingual data is not available, even low-resource languages may have large amounts of monolingual data available. This is also the case for Yupik, which has only 8,000 sentences of bilingual data, but over 20,000 sentences of digitized monolingual data. The backtranslation strategy, developed by Sennrich et al. (2016a) has been demonstrated to improve the translation quality using this kind of monolingual data. Given that low-resource languages have such limited amounts of low-resource data, research into the techniques which make backtranslation effective are critical. It could be the case that a model with poor initial quality, such as the models trained in this project, would not be suitable for generating the synthetic data necessary for backtranslation. This is a key research question that should be answered in future research on Yupik translation.

Another potential approach is to exploit the phonological similarities of Inuktitut and Yupik given that they are related languages. Although the multilingual models in this project likely learned correspondences between Yupik and Inuktitut which originated from their common ancestry, the process to learn these correspondences is computationally intense, and difficult to interpret at the end. On the other hand, an approach which identies the sound correspondences between cognates in Yupik and Inuktitut would provide a wealth of data useful for translation, and potentially be much less computaionally intense to determine. Although the field of computational historical linguistics is very young (Jäger, 2019), methods have been developed which can automatically find potential cognates in related languages. However, these approaches do not account for polysynthetic languages. If an approach could be developed which can directly translate Inuktitut to Yupik using phonetic correspondences between cognates, a large amount of high quality synthetic data could be generated, which would be highly useful for a machine translation system. Additionally, if this approach is successful, it could generate much more interest for historical linguistics in the NLP community.

One final approach which could be useful for Yupik translation is a direct neural speech to text translation system. Many low-resource languages have limited written materials, or even no written materials at all. For these languages, translation systems which take speech as input instead of text would be useful. Such a system was developed by Bansal et al. (2019), which uses a neural architecture which takes a speech waveform of a source language as input, and converts it to a text representation in the target language. Using a transfer approach, Bansal et al. (2019) trained a Mboshi-French translation systems using only 4 hours of Mboshi speech. Their highest performing translation model achieved a BLEU of 7.1, close to the highest performing model in this study. Luckily, the Yupik translation of the New Testament is also available in audio form, which contains much more than 4 hours of speech. One way we can understand the effectiveness of a speech-to-text translation model is that a translation model based on speech will be able to access the phonology of the language directly, rather than through an imperfect or irregular representation in the orthography. A speech-to-text translation system therefore may be very adept at translating languages which are both low-resource and polysynthetic. Therefore, extending Bansal's translation model to Yupik would be a very beneficial research project.

If these techniques fail to produce significant improvements, more research should be devoted to SMT systems for low-resource languages, as these experiments found SMT models to perform strongly when compared to optimized NMT systems.

Therefore, there are many new directions which could be applied to producing high quality Yupik translation. These approaches will be applied in future studies.

Bibliography

Bahdanau, D., K. Cho, and Y. Bengio
2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Banerjee, S. and A. Lavie
2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, Pp. 65–72.

Bansal, S., H. Kamper, K. Livescu, A. Lopez, and S. Goldwater
2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Pp. 58–68, Minneapolis, Minnesota. Association for Computational Linguistics.

Bloodgood, M. and C. Callison-Burch
2010. Using mechanical turk to build machine translation evaluation sets. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, Pp. 208–211, Los Angeles. Association for Computational Linguistics.

Butnaru, A. and R. T. Ionescu
2018. UnibucKernel: A kernel-based learning method for complex word identification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Pp. 175–183, New Orleans, Louisiana. Association for Computational Linguistics.

Chen, E. and L. Schwartz
2018. A morphological analyzer for st. lawrence island/central siberian yupik. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

Cherry, C., G. Foster, A. Bapna, O. Firat, and W. Macherey
2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in*

Natural Language Processing, Pp. 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Cho, K., B. van Merriënboer, D. Bahdanau, and Y. Bengio
2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Pp. 103–111, Doha, Qatar. Association for Computational Linguistics.

Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio
2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Pp. 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Coughlin, D.
2003. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT summit IX*, Pp. 63–70.

Dauenhauer, N. M. and R. Dauenhauer
1998. Technical, emotional, and ideological issues in reversing language shift: Examples from southeast alaska. *Endangered languages: Current issues and future prospects*, Pp. 57–98.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova
2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Pp. 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dorian, N. C.
1998. *Western language ideologies and small-language prospects.* na.

Dostert, L. E.
1955. The georgetown-ibm experiment. *1955). Machine translation of languages. John Wiley & Sons, New York*, Pp. 124–135.

Fortescue, M.
1993. Eskimo word order variation and its contact-induced perturbation. *Journal of Linguistics*, 29(2):267–289.

Fortescue, M. et al.
1994. *Comparative Eskimo Dictionary with Aleut Cognates.* ERIC.

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

Green, S., J. Heer, and C. D. Manning
2013. The efficacy of human post-editing for language translation. In *Proceedings of the SIGCHI conference on human factors in computing systems*, Pp. 439–448. ACM.

Grenoble, L. A. and L. J. Whaley
1996. Endangered languages: Currentissues and future prospects. *International journal of the sociology of language*, 118(1):209–223.

Gu, J., H. Hassan, J. Devlin, and V. O. Li
2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Pp. 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Heafield, K., I. Pouzyrevsky, J. H. Clark, and P. Koehn
2013. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Pp. 690–696.

Hewitt, J. and C. D. Manning
2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Pp. 4129–4138.

Hinton, G., L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al.
2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29.

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

Hornik, K., M. Stinchcombe, and H. White
1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hulden, M.
2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, Pp. 29–32. Association for Computational Linguistics.

Imankulova, A., R. Dabre, A. Fujita, and K. Imamura
2019. Exploiting out-of-domain parallel data through multilingual transfer learning for low-resource neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, Pp. 128–139, Dublin, Ireland. European Association for Machine Translation.

Iutzi-Mitchell, R. D. and N. H. Graburn
1993. Language and educational policy in the north: Status and prospectus report on the eskimo-aleut languages from an international symposium. *International Journal of the Sociology of Language*, 99(1):123–132.

Jacobson, S. A.
1977. A grammatical sketch of siberian yupik eskimo.

Jacobson, S. A.
2001. *A Practical Grammar of the St. Lawrence Island/Siberian Yupik Eskimo Language*. ERIC.

Jäger, G.
2019. Computational historical linguistics. *Theoretical Linguistics*, 45(3-4):151–182.

Johnson, M., M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al.
2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Jurafsky, D. and J. H. Martin
2008. Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing. *Upper Saddle River, NJ: Prentice Hall.*

King, B. P.
2015. Practical natural language processing for low-resource languages.

Klein, G., Y. Kim, Y. Deng, J. Senellart, and A. Rush
2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, Pp. 67–72, Vancouver, Canada. Association for Computational Linguistics.

Koehn, P.
2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, Pp. 388–395.

Koehn, P.
2019. "Evaluation Matrix: newstest2019 using metric BLEU", howpublished = http://matrix.statmt.org/matrix. Accessed: 2019-12-16.

Koehn, P.
Forthcoming 2020. *Neural Machine Translation*. Cambridge University Press.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al.
2007. Moses: Open source toolkit for statistical machine translation. In

Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions, Pp. 177–180.

Koehn, P. and R. Knowles
2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, Pp. 28–39, Vancouver. Association for Computational Linguistics.

Krauss, M. E.
1975. St. lawrence island eskimo phonology and orthography. *Linguistics*, 13(152):39–72.

Krizhevsky, A., I. Sutskever, and G. E. Hinton
2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Lakew, S. M., M. Cettolo, and M. Federico
2018. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, Pp. 641–652, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lo, C.-k.
2019. Yisi-a unified semantic mt quality evaluation and estimation metric for languages with different levels of available resources. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, Pp. 507–513.

Luong, M.-T., P. Nakov, and M.-Y. Kan
2010. A hybrid morpheme-word representation for machine translation of morphologically rich languages. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Pp. 148–157. Association for Computational Linguistics.

Luong, T., H. Pham, and C. D. Manning
2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Pp. 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Mallon, M.
1999. Inuktitut linguistics for technocrats. *National Research Council Canada. http://www. inuktitutcomputing. ca/Technocrats/ILFT_1. html (accessed January 28, 2011).*

McCulloch, W. S. and W. Pitts
1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Michel, P., O. Levy, and G. Neubig
2019. Are sixteen heads really better than one? *arXiv preprint arXiv:1905.10650*.

Micher, J. C.
2018. Addressing challenges of machine translation of inuit languages. Technical report, US Army Research Laboratory Adelphi United States.

Miller, G. A.
1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Miyaoka, O.
2012. *A Grammar of Central Alaskan Yupik (CAY)*, volume 58. Walter de Gruyter.

Mooney, R. J.
2014. Semantic parsing: Past, present, and future. In *ACL workshop on semantic parsing. Presentation slides.*

Neubig, G.
2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

Och, F. J.
2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, Pp. 160–167. Association for Computational Linguistics.

Och, F. J. and H. Ney
2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu
2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, Pp. 311–318. Association for Computational Linguistics.

Peris, A.
2019. confidence_intervals. https://github.com/lvapeab/confidence_intervals.

Peters, M., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer
2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Pp. 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Rosenblatt, F.
1957. *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams
1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Schroff, F., D. Kalenichenko, and J. Philbin
2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Pp. 815–823.

Schwartz, L.
2018. The history and promise of machine translation. *Innovation and Expansion in Translation Process Research*, P. 161.

Schwartz, L.
Fall 2019. Personal communication.

Schwartz, L. and E. Chen
2017. Liinnaqumalghiit: A web-based tool for addressing orthographic transparency in st. lawrence island/central siberian yupik. *Language documentation and conservation*, 11.

Sennrich, R., B. Haddow, and A. Birch
2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Pp. 86–96, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R., B. Haddow, and A. Birch
2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Pp. 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sennrich, R. and B. Zhang
2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Pp. 211–221, Florence, Italy. Association for Computational Linguistics.

Smit, P., S. Virpioja, S.-A. Grönroos, and M. Kurimo
2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, Pp. 21–24.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov
2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Strubell, E., A. Ganesh, and A. McCallum
2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Pp. 3645–3650, Florence, Italy. Association for Computational Linguistics.

Taylor, W. L.
1953. "cloze procedure": A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

UNESCO
2019. "About IYIL 2019". https://en.iyil2019.org/about/. Accessed: 2019-11-25.

Vakhtin, N.
1998. Endangered languages in northeast siberia: Siberian yupik and other languages of chukotka. *Bicultural Education in the North*, Pp. 159–173.

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

Virpioja, S., O. Kohonen, and K. Lagus
2009. Unsupervised morpheme discovery with allomorfessor. In *CLEF (Working Notes)*.

Weaver, W.
1947. Letter to norbert wiener, march 4, 1947. *Rockefeller Foundation Archives*.

Weide, R.
2014. The Carnegie Mellon Pronouncing Dictionary [cmudict. 0.7b]. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Yupik Translation Committee, Wycliffe Bible Translators, Inc.
2012. The New Testament in Yupik of Saint Lawrence Island, Alaska. Wycliffe Bible Translators, Inc.

Zoph, B., D. Yuret, J. May, and K. Knight
2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Pp. 1568–1575, Austin, Texas. Association for Computational Linguistics.