Evaluating the Complexity of Welsh to English Machine Translation Tasks

Msc AI & Machine Learning

Alexander Mainstone

Supervised by Dr. Peter Hancox

UNIVERSITY OF BIRMINGHAM

Submitted September 2021

Abstract

In this project, multiple Models for Machine Translation were trained and evaluated across multiple languages, with the aim of evaluating how difficult Welsh to English translation is compared to other languages (specifically; Welsh, English, Danish and French). This paper has proven the initial hypothesis (that Welsh is a difficult language to translate) wrong, and instead proven that compared to all other tested languages, the Welsh language is a relatively easy translation task, likely due to the infrequency of Welsh inflection compared to inflection in other morphologically rich languages. This infrequency does however result in niche cases that unprepared models have difficulty translating, this can be remedied by having a method to handle synonyms or having a corpus large enough to cover all possible mutation cases (likely unfeasible). Models and languages have been evaluated using automated metrics, primarily BLEU scores, but also using NIST and METEOR. This paper details the implementation of a phrase-based statistical translator from scratch, and an attention based transformer model in PyTorch. Detailed in this paper, is the training and evaluation of machine translation toolkits for both neural machine translation (OpenNMT) and statistical translation (Moses). Translations were also fetched by Google Translate for evaluation. No significant differences were found across models for the ease of translation. This paper attempts to quantify why might Welsh be an easier language to translate compared to the other selected languages.

Having evaluated the difficulty of Welsh translation tasks, this project has been considered a success. Further work however is needed at improving these results with higher quality models and datasets.

Acknowledgements

Thank you to my supervisor, Dr. Peter Hancox for the support and feedback on my work throughout development. Thank you also to Dr. Mohammed Bahja for pointing me in the right direction during my project inspections.

Table of Content

1 Introduction

The research area of Natural Language Processing has relatively little academic research in the context of the Welsh language. This paper will explore the area of Machine Translation in the context of the Welsh language and the unique problems that may present, attempting to evaluate the difficulty of Welsh translation tasks relative to other language translation tasks.

We will attempt to explore methods for Machine Translation between the Welsh and English Languages and comparing them with their corresponding French-English models (as well as Danish and Spanish in a limited capacity). Only Statistical and Neural methods of translation will be explored in this paper as these two methods have been shown to perform the best, and it is important to note that we will not be developing a general-purpose translation system, but rather a system that aims to specifically translate Welsh, French, Danish and Spanish Assembly transcripts, as defined by our corpuses, focusing on political language. Restricting our domain to this corpus vastly simplifies the problem, and will result in more accurate translations from a limited corpus running on a machine with limited hardware.

The goal of this paper is not to develop new methods of translating Welsh to English, but rather to evaluate how difficult a translation task the Welsh language presents when compared to other European languages from different language families with different levels of morphological richness.

We will focus on translating from Welsh to English, entirely for clarity reasons, as it will be assumed that the reader has no knowledge of the Welsh language, and thus a thorough and complete explanation on why a Welsh phrase is grammatically incorrect can be avoided. In saying this, Welsh does have some interesting and complex grammatical patterns that will likely influence machine translation efforts, and we will explore those to discuss why results may have been influenced one way or another as well as potential solutions to some unique problems that Welsh can present. We will however, have difficulty identifying why the French, Spanish and Danish models are performing the way they do, as we have very limited linguistic knowledge of those languages.

The motivation behind implementing both a Neural and Statistical translator is to see if a given language will perform better on one relative to other languages. It is unlikely that any language will perform better using Statistical Translation than a Neural Translator, but perhaps Statistical translation presents unique challenges for certain languages, by looking at the variation between language scores, we can see if one of our languages struggles with a certain form of Machine Translation. Another reason is simply to learn about all forms of Machine Translation. Neural Translation is considered the standard for Machine Translation today, but much of what Neural methods do are inspired by their statistical counterparts and predecessors. Not only this, but techniques common in Statistical Translation can be used to complement Neural translation (Beam Search etc).

The statistical implementation of the translation model will be implemented from scratch, using python and some helper libraries (json, numpy, operator, re).

The Neural implementation of translation model will be implemented using PyTorch, the complexity of implementing a Neural Machine Translator from scratch is beyond the scope of this thesis, and would likely take all the time available. PyTorch's "ignite" library will be used for its implementation of the BLEU metric.

We will also implement models using state-of-the-art Machine Translation toolkits, as well as attempting to evaluate translation quality for Google Translate.

All training and running of models has been done on a mid-range machine running a GeForce GTX 1060 GPU, with 16GB of RAM and a Ryzen 5 1600 six-core processor. The Moses toolkit and Fast Align were both developed with Linux systems in mind, and therefore we have done our work with them on Ubuntu laptop lacking a GPU. There will be a limit set to how long any model will be given to train of 2 days, if a model exceeds this limit, hyperparameters and corpuses will be adjusted to reduce training time. This limit is to ensure enough time is given to retrain models if needed without taking several days.

2 Literature Review

In this section, relevant literature to our work is explored with an attempt to gain insight to our task prior to implementing and designing our translation systems. This section is divided into academic research into Statistical Translation, followed by Neural translation before we investigate the current state of machine translation in general for the Welsh language and then discuss research regarding the automatic evaluation of translated sentences.

2.1 Statistical Methods in Literature

Much of the contextual knowledge required to implement statistical machine translation systems has been gathered through the use of the book "Statistical Machine Translation" by Philipp Koehn [1], it has been an invaluable resource in learning much of the theory of statistical translation. This book details the theory behind statistical translation methods, the knowledge of which will allow me to implement these methods. Koehn has also been involved in the development of a statistical machine translation toolkit known as "Moses" [2], which could potentially be used to evaluate a high quality statistical model, ensuring that our implemented models are not producing flawed results.

Literature regarding Statistical Machine Translation can be found as early as 1949 by the cryptographer, Warren Weaver [3], with prior methods essentially amounting to rule-based approaches functioning as mechanical dictionaries [4]. This statistical method proposed in

1949, attempts to solve multiple meanings, by examining the context (the N words before and after) of a given word, a statistical approach can decide a given meaning. Naturally, it would be impractical to store every possible $2N + 1$ words for translation, hence taking a statistical approach. Weaver proposes that we use the probability that a given string from the target language is the correct translation of a string in a given text to predict potential translations.

A point to consider mentioned by Weaver in their 1949 paper, is that Technical writing, if not simple is often devoid of common contextually dependent meaning, the author uses Mathematics as an example, stating that; "one can very nearly say that each word, within the general context of a mathematical article, has one and only one meaning" [3]. This is an interesting point; will the selected Welsh Assembly corpus be an easier Machine Translation task or a harder one compared to other translation tasks? Does political language tend to be more or less concise than spoken language?

There are many ways that we can divide our sentences to generate a probability table, with the initial statistical models dividing their sentences into words to be translated (given a word, what is the most probable translation? How probable is it?) [1], but since has been shown that phrase-based approaches can significantly outperform word-based approaches [5], as one language will likely have words that consist of many other words in another language, or many words will translate to one (word-based approaches have great difficulty handling these occurrences). There are also syntax-based approaches currently being researched, which excel at producing syntactically "good" translations, if not better than phrase-based models at preserving meaning [6]. Combining syntax and phrase based translation methods has been proposed, leading to the method known as Hierarchical phrase-based machine translation, which divides phrases into further sub-phrases for translation [7]. For this project, we must think of what can be achieved in the time given to work, thus finding a good balance between ease of implementation and performance is necessary. For this reason we will focus on phrase-based models, as the improvements in more complex models are largely syntactic, and a word-based model would not perform well.

2.2 Neural Methods in Literature

Again, a large amount of the contextual and foundational knowledge was found in the book "Neural Machine Translation" by Philipp Koehn [8].

Neural methods of machine translation tend to outperform statistical methods. This can be seen in the performance benchmarks at the Conference on Machine Translation [9], with the most successful translation method being a neural one.

However, there has been research to show that statistical methods of translation (language models, phrase tables and reorder metrics) can be used in conjunction with neural models, providing models with additional translation recommendations [10]. These methods show promise but will not be explored in this paper.

A paper discussing the challenges of Neural Machine translation [11] discusses the areas that neural methods are likely to fall short. One point made is that neural translators have difficulty translating low-frequency words belonging to highly-inflected categories (word modification). This may lead Neural translation methods to have great difficulty with Welsh's concept of word mutation, in which both nouns and adjectives can have their initial letters change. The paper also discusses the issues that neural translators have with very long sentences, stating that neural methods do comparably better up to a sentence length of about 60 words, so some limit below 60 will be used for the training of our model. The paper also briefly mentions network architecture, stating that all recent work has been focused on encoder-decoder models of translation.

Recurrent Continuous network models [12] were the first to not require the usage of statistical techniques such as phrase alignment. Recurrence is a concept that, as its name suggests, performs translation using recurrence. Essentially, the hidden-state (weights etc) are re-used from the first word translation to translate the second word and so on. This allows for networks to have a limited amount of contextual knowledge about a sentence. A limitation with this kind of model is speed, performing any action recursively will make the problem exponentially tied to its input sentence length. These models still performed worse than state of the art alignment-based translation models, which have been surpassed more recently.

Seq2Seq [13] (or sequence "2" sequence models), are a type of encoder-decoder model, these models are made up of (as the name suggests) both an encoder network and a decoder network. The goal of these networks is to convert words to vector space and vice versa, with the encoder converting words from a source language into a vector representation and the decoder taking this vector, and attempting to convert it into a corresponding word in its target language. This method, when not optimized with other techniques, will result in heavy translation errors as sentences grow, this is because contextual knowledge is lost.

Neural translator's issues regarding limited sentence contextual knowledge have been addressed with the concept of attention [14]. Rather than keeping a limited knowledge (usually around 10 words), the network use an attention mechanism, which essentially aims to allow models to focus on contextually important parts of a sentence, no matter the length (although they will still suffer some loss of context as sentence size increases). While *seq2seq* models exist that apply Attention in their translation [15], this paper proposes a Transformer model, which while inspired by *seq2seq* Models, uses no recurrence or convolution. The proposed model is made exclusively from attention and feedforward operations, this lack of recurrence means that this model is incredibly fast to train and use while being considered one of the most powerful architectures for Machine Translation currently available. As detailed in the paper, a Transformer model provides an alternative to Recurrent Neural Networks, however they can process the input sentence in any order, rather than sourcing context from other words, the attention mechanism will provide the relevant context, identifying the meaning of each word. By not needing to translate and process other parts of a sentence, a Transformer model can translate all parts of the sentence in parallel.

A morphologically rich language is a language is a language heavy in word changes that indicate meaning or context. Morphologically rich languages can cause many problems, essentially inflating the size of vocabularies by filling them with synonyms or conjugations. Methods have been proposed to try and handle inflected word forms from these languages, from character based approaches [16] to probabilistic models [17]. This will be considered beyond the scope of this paper, but is a potential solution to some of the problems that may occur.

2.3 Welsh Language Translation in Literature

It seems that Neural methods of Welsh Machine translation is a rather unexplored area. We were able to find very few papers on neural implementations, of those found they were primarily focused on massively multilingual translation (a single model capable of translating to and from many languages) with welsh happening to be one of them [18], therefore no direct observations of Welsh translations are stated. The most notable multilingual implementation would be Google Translate, which uses Google Neural Machine Translation to translate to and from over one-hundred languages, including Welsh, however an article by the BBC discusses anecdotally how problematic Google Translate can be for Welsh language translation, with many non-welsh companies relying on Google Translate for their Welsh content translations [19], however this view of the quality of Welsh translation is done without considering the quality translations in other languages. It will be interesting to investigate if these Welsh translations are uniquely bad or if it's simply a failing of machine translation overall. It is also possible that the overall negative view of machine translation (primarily Google Translate) is founded upon selection bias, with poor translations seeming more significant than correct ones. The article does observe that these bad translations were making it into public documents, this demonstrates the importance of developing high quality Welsh translators. With Welsh being a highly used (required for all government documents and street signs) but rarely spoken (fewer than 30% of Welsh people can speak Welsh [20]), there is a possibility of translation software causing the quality of Welsh documents to decline, as using imperfect translation tools is both easier and cheaper than hiring a Welsh language translator.

One article details how useful Statistical Machine Translation can be in increasing the productivity of translators by using Statistical Translators to complement their translations [21]. This illustrates the importance of high quality translators in the translation industry, but has little else of note. Another paper [22] experimented with embedding statistical models in translation companies, to positive results. Stated in that paper, was the potential for improvement with the implementation of Neural translators. Experiments seem to be needed on how translators could improve assisted by state of the art neural translation systems.

It seems primarily, the limited research in Welsh Machine Translation, seems to be focused on how useful the technology can be, rather than implementing or evaluating it. And of that, we could find no articles having implemented Neural Translation for Welsh (other than massively multilingual ones). Neural Research in the area of Welsh Translation seems to need expanding.

2.4 Automatic Evaluation Metrics

Human judges are the best way to evaluate translation success. Evaluating thousands of sentences for languages that we do not speak is beyond the scope of this paper, and incredibly expensive and time-consuming. Fortunately, a number of automated methods of evaluating sentences with differing degrees of correlation with human judges exist. We will examine some of the more interesting and applicable here, and use them for evaluating our models later.

Reading through the previous citations, the common factor among them is that all of them have used BLEU for evaluation (along with other metrics). BLEU, or *Bilingual Evaluation Understudy*, initially proposed in 2002 [23] was one of the first metrics to show a high correlation with human evaluation, as well as being language independent. The paper states that BLEU provides a quick alternative to human judges. It works through counting the occurrences of n-grams and comparing them to the occurrences in the true source sentence, an n-gram being a set of n words in a sentence. It was found in the proposal paper that 4-grams were the optimal form of n-gram for correlation with human translators. A BLEU score will range from 0-1 where 1 is a perfect match with one of the translation candidates and 0 having no matches with the candidates. Because of this, it is unlikely to ever get a perfect BLEU score, as a correct translation still may not be structured exactly like the reference translation.

NIST scores attempt to build on BLEU, named for the National Institute of Standards and Technology (where it was developed) in 2002 [24], NIST weighs n-gram pairs based on how rare a given n-gram is. As you will see in the next section, it was found to outperform BLEU. Unlike other metrics, it does not return a score between 0 and 1, but rather an uncapped number, with higher signifying better.

METEOR, or *Metric for Evaluation of Translation with Explicit Ordering*, proposed in 2005 [25] as an alternative to BLEU, attempts to solve the weaknesses in BLEU to better correlate with human judgement. The paper concluded that METEOR scores correlated with human judgement at a rate of 0.964, compared to BLEU's correlation on this dataset being 0.817 or NIST, which achieved a correlation of 0.892. METEOR achieves this by attempting to match synonyms and reducing inflected words to their root form.

All of these metrics have been implemented in the nltk Python library, which we will be using to evaluate our models.

3 Hypothesis

Hypothesizing the outcome of this experiment is difficult as we have a limited understanding of other languages we will be evaluating. However, our understanding of the unique Welsh language conventions has lead to the following hypothesis:

Welsh is likely an inherently more difficult language to translate than French, due to complex language conventions such as mutation and its large use of "meaningless" words.

It is important to not that by "meaningless" we mean that Welsh contains many words that ultimately are unlikely to contribute to the true meaning of a sentence (we will investigate this further when providing examples of alignment in our statistical implementation). There are also examples of singular English words translating to long Welsh phrases; *"sorry"* translates to *"Mae'n ddrwg gen i*, as well as this sentence requiring a subject (the *"i"* in the sentence signifies the speaker), it will be interesting to see how a model can differentiate between subjects when it comes to translation.

4 Methodology

We will be implementing a wide range of models using both Toolkits and Python, with the aim of evaluating translation between languages rather than these models. We will take no steps to ensure that models are given an equal amount of resources and time to train, as we will not be comparing models directly, we will however ensure that each language for a given model is given the same amount of time and resources to train so that we may compare our findings between languages.

We will be implementing the following models:

Model Name	Train Size	Validation Size	Test Size	Languages
Google Translate	N/A	N/A	1,000	Welsh, English
OpenNMT	350,000	5,000	75,000	Welsh, English
Moses	350,000	75,000	75,000	Welsh, English
Neural	150,000	5,000	75,000	Welsh, English, Danish, Spanish
Statistical	350,000	500	1,000	Welsh, English

5 Corpuses

The selected corpuses will be filtered using the Moses toolkit, which will provide tools that can perform tokenization (insert spaces between punctuation), attempt to remove misaligned sentences, ensure consistent and appropriate capitalization and remove sentences over 80 words (an arbitrary limit to ensure there are no massive sentences). Both corpuses will be run through these processes, and split into datasets for training, validation and testing (although for some models reduced sets will be used for various reasons). The split will be 70/15/15, with 70% being used for training, although we are unlikely to use the full validation sets. Both datasets will contain 500k sentences, which will mean 350k are used for training and 150k are used for training and 75k for validation and testing. These splits can be achieved with a simple Python program that will split a corpus into these three parts.

It should be noted that our work here will be significantly restricted by the quality of our datasets. The primary issue being that we only have a corpus containing one valid translation rather than multiple possible translations. Having multiple candidates for translations available would allow for better automated evaluation, scores like BLEU work better with multiple translations to compare predictions to, as it is possible to translate a given sentence in a large variety of ways. Ideally, we would have a corpus containing multiple translation candidates, but we could not find an existing Welsh corpus containing multiple candidates. We could also create a corpus with this feature, but this would be expensive to create and is beyond the scope of this paper. We will still be using automated metrics to measure translation quality, but they will be unable to achieve their full accuracy while only given one example of a correct translation.

It should be also stated that every language we have chosen includes gendered nouns, a given noun can be either masculine or feminine and language rules depend on the gender of a given noun. This would primarily present an issue for translating in to that language from another, understanding the gender of a given noun would likely be difficult for many models to do. In two of the chosen languages (Danish and Welsh), the gendered nouns cause the following word to be inflected a certain way, thus increasing vocabulary size.

5.1 Welsh Corpus

As mentioned above, the corpus consists of translated transcripts from the national Welsh assembly meetings, this dataset was obtained from a paper attempting to implement phrase-based statistical machine translation [26]. The corpus is provided pre-aligned by sentence, but lacks phrase alignment data, which we will have to find means of generating if we intend to implement phrase based translations.

Examination of the dataset has revealed that it is missing accent characters, primarily the circumflex for vowels (â, ê, î, ô, û, ŵ, ŷ), which could lead to conflicting word definitions where there need not be. An example for this would be the word *"tŷ"* (meaning *"house"*) in its mutated form *"dŷ"* and the word *"dy"* (meaning *"your"*), in this instance a model would have to infer the meaning through contextual clues, which it will have to do in many cases anyway, but instances like this could be inferred without need for context if the circumflex character were to be preserved. There are other accent characters that occur in welsh, these however, occur far less frequently and would not cause conflicts when mutation is involved.

The mutation taking place in the example above will present a unique translation problem, essentially, the context that a noun is in can sometimes impact the first letter(s) of said word, this also depends on the gender of a noun (masculine or feminine). Fortunately, for our aim of translating from Welsh to English, and not the other way we can largely ignore this issue. It will however result in duplicates of words and phrases occurring in the translation tables.

Welsh mutation has Soft, Nasal and Aspirate candidates for letters, it is not important to understand why they occur only that they do. The candidates for mutation are the following:

Initial	Soft	Nasal	Aspirate
c	g	ngh	ch
p	b	mh	ph
t	d	ng	th
g	_	ng	
b	f	m	
d	dd	n	
ll	l		
m	f		
rh	r		

_ - *represents the dropping of a letter*

blank space indicates no mutation candidate (in which case no mutation occurs)

As you can see, each noun has potentially up to 4 different candidates for mutation, which could vastly inflate the number of "unique" words as understood by a model. It is unlikely that any model will apply mutation, they will simply acquire optimal translation candidates for each mutation. It will however mean that models will struggle heavily with rare mutation occurrences.

Here is an example of each form of mutation for the word cat:

Mutation	Welsh	English
Initial	**c**ath	cat
Soft	*ei* **g**ath	their cat
Nasal	*fy* **ngh**ath	my cat
Aspirate	ci *a* **ch**ath	dog and cat

Bold - mutated character

Italics - mutation cause

Noun gender can also impact the mutation type occurring, as well as the surrounding words. The Welsh word for ”two” is ”dau” in the context of masculine nouns and ”dwy” in the context of feminine nouns.

5.2 French Corpus

The French Corpus we will be using is sourced from European Parliament transcripts (EuroParl) [27]. This corpus is much larger than the Welsh corpus, with about 2 million aligned sentences, therefore we will trim it to match the size of the Welsh Corpus (500k aligned phrases).

Although we lack understanding of the French language and cannot evaluate the quality of the dataset to the same standard as the Welsh language one, we have noticed some misalignment problems; primarily, it seems there are many translated sentences that have corresponding French sentences that are blank. Through processing the corpus through the Moses toolkit we can remove these misaligned sentences.

Not having a full understanding of the French language will prevent us from giving a full justification on why French models might behave the way it will, we are unfamiliar with French language conventions and sentence structure, and no amount of research will be able to give the comprehensive understanding that comes with language fluency (Welsh fluency will give us insight to why a Welsh model may behave a certain way).

5.3 Additional Corpuses

Alongside the previous two languages, we will be evaluating an additional 2 languages specifically in our Neural implementation of machine translation. Comparing the Welsh language exclusively to French will give us a poor perspective on how difficult it is to translate Welsh. Therefore, we will be comparing it to Spanish and Danish. We will be using the Europarl [27] corpuses for Danish and Spanish for their implementations.

Spanish has been chosen as it is considered relatively easy to learn for English speakers, thus it will be interesting to see how successfully a machine translation model can translate this language to English. It is also one of the most spoken languages in the world.

Danish has been chosen primarily as an alternative to German. We could not source the German to English corpus from the Europarl website, and it seems to be down. Danish was chosen as an alternative to German, primarily because it also includes the language convention of conjugation, where words can be combined to form new words. Pink in Danish is simply ”light red” (”lyserød”) as one word. Prefixes like ”the” are replaced by suffixes attached and modifying a word, it can be seen as a much simpler form of the Welsh Mutation system, but we believe that a machine translation model will have far more difficulty with this danish form as it is used far more regularly (every time ”the” would be used in English).

6 Implementing a Statistical Translator

Our statistical approach at machine translation will be phrase based. A phrase based model will be defined by 3 parts:

- The language model
- The phrase translation table
- A reorder metric

These parts will be used to calculate the probability of a translation. Combined in a variety of ways these three metrics can be used to quantify how good the current step in the translation process is.

We will be implementing this system in pure Python, as implementing it from scratch will grant a greater understanding of all the underlying processes behind Statistical translation, and we believe it to be a realistic goal to achieve in the given time.

Implementing the translator in python does come with some issues however, primarily being that Python is a slower language than its lower level counterparts [28], both Moses and PyTorch are largely written in C++ for its speed, and this will result in anything that we can create in python running slower than it would if it were written in C++. This sacrifice is likely worth it, as we are working under a time constraint, and python will allow us to implement a model far quicker than C++, due to higher level language features.

For the reason above, we will need to use optimization techniques to come close to implementing a system that can complete a testing set in a reasonable amount of time, as we will need to translate a large corpus of sentences to evaluate the models.

6.1 Word alignment

Word alignment is a complex topic that could have an entire paper written about it, and is not the focus of this paper. Essentially, it is a mapping between translations, showing the relationship between words in each language (for further reading, the problem is also known as Bi-text word alignment).

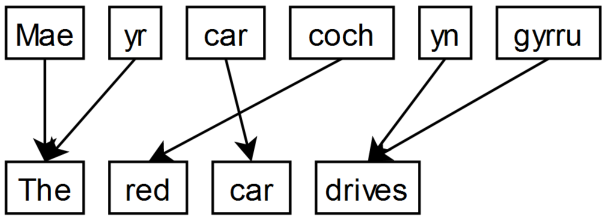

Figure 1: An example of a simple word alignment problem and its solution

As shown in the figure, multiple words can map into a single word (the opposite is also true). This is why word-based systems are inadequate at translating most languages; if every word directly equates to another, what would *"yr"* or *"yn"* translate to? Another problem is word order, the sentence structure of each language can be vastly different, in Welsh the adjective occurs after the subject of the sentence (notice that "red" and "car" are swapped from the Welsh source).

To solve these problems, we will use a word alignment tool that has already been written. In this case we will use *Fast Align* [29], a simple alignment tool that works quickly to generate Pharaoh format files, which can then be parsed to get the appropriate phrase-pairs. This will likely not result in a perfect phrase pair set, but across a large dataset, small anomalous errors will be of little consequence. The examples of usage are in Linux, with package requirements, making it difficult to set up on Windows systems. For this reason, alignment will be performed in Ubuntu, and then transferred to the Windows system to be used in model training.

We then must parse these word alignment files in Python. Fortunately, Philipp Koehn's book "Statistical Machine Translation" [1] outlines in pseudocode a function to do this. Unfortunately, the implementation Koehn outlines is incorrect. After trying and failing to implement the outlined pseudocode, we decided to research if an implementation of this pseudocode had been done before, and found a StackOverflow user who had encountered the same problem, with an answer recommending the optimal implementation of the pseudocode [30].

Having performed alignment, and then parsing the phrases into corresponding pairs, we can then create a phrase model in the following step.

6.2 Phrase Model

The phrase model is simply a count of how frequently one phrase translates to another. This involves counting the occurrences of a given translation for a phrase in our corpus, and then the probability of that translation given the input phrase can be calculated as:

$$p(e_i|w_j) = \frac{count(w_i, e_j)}{\sum_{k=1}^{N} count(w_i, e_k)}$$

Where the *count* function returns the number of occurrences the English phrase e occurs as a translation of the Welsh phrase w (or the French phrase). Essentially, this equation means that the probability of a given sentence being the translation of another sentence is the number of occurrences of the initial sentence divided by the total number of translations for the source sentence. This will result in the sum probability of all possible translations of a given phrase being 1.

We will store the phrase model as a Python dictionary, where a phrase can be given, and it will return a sorted list of translation outcomes from most probable to least probable along with said probability (which will be used for evaluation later).

6.3 Reordering Metric

There are many techniques that can be used to limit reordering, in this implementation we will go with one of the simpler approaches: distance-based reordering. Distance-based reordering uses the distance between a phrase's position in a source language to its new translated position to generate a score. This score is used to negatively weigh translations where the order of the sentence is drastically changed. It can be computed as:

$$dist = start_i - end_{i-1} - 1$$

Where $start_i$ represents the start position of the ith foreign phrase in its translation and end_i the end of the phrase.

6.4 Language Model

The Language Model is responsible for ensuring that the output of the phrase model and reorder model are grammatically correct. In this case, we have implemented a tri-gram model of English, as we only need a language model of the language we're translating to. The language model, as implemented in Python, takes the form of a 3-dimensional dictionary, with each "dimension" representing a word. Given three words, this dictionary returns the probability that they occur in the order they were given in. An illustration of this data-structure can be seen below.

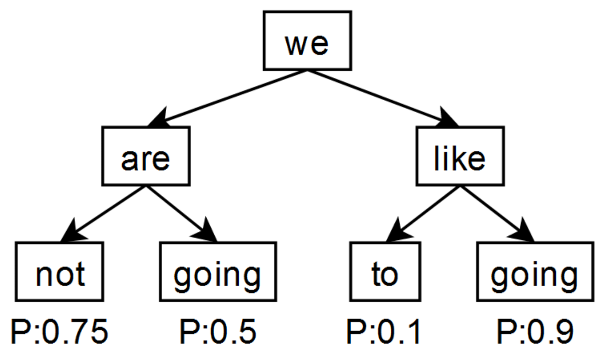

Figure 2: An illustration of tri-gram word alignment.

This data-structure is constructed by simply counting the frequency that a given word occurs after its two preceding words. We perform this across our English training dataset, which will mean our models understanding of correct grammar is purely based on the grammar used in this dataset. This will not result in perfect output grammar, this is a form of Markov chain, meaning that it is only concerned with the current state (in this case the current 2 words) to determine the probability of future words, meaning that it does not consider the sentence as a whole, but rather breaks the sentence into smaller, overlapping parts to evaluate. Evaluation of the start and end of a sentence is performed by inputting start and end tokens into the sentence, these tokens are never seen in output translations, and simply serve the tri-gram model by announcing to it that a given word is before the end of the sentence or the first word in a sentence. Below is an illustration.

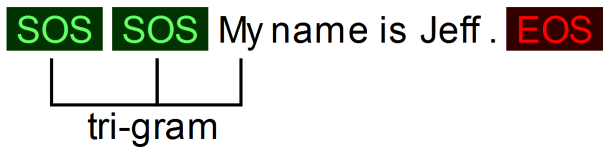

Figure 3: Illustration of the formatted sentence the tri-gram views. (SOS: start of sentence, EOS: end of sentence)

One thing perhaps unclear in this illustration, is how punctuation is taken into account. Through the processing of our dataset using the Moses toolkit, a space has been entered before every punctuation mark, allowing for the language model to view each punctuation mark as simply another word, which will teach it to identify good and bad punctuation.

To calculate the total tri-gram score of a sentence we iterate through each word and punctuation character (the start tokens are not iterated through, they are to evaluate the first 2 words) to calculate their sum.

6.5 Decoding

Decoding is the primary problem in Statistical Machine Translation. Somewhere in our statistical models, is the perfect translation to a large amount of sentences, the problem is only discovering this translation. Discovering the perfect translation is unfortunately, an incredibly exponential problem; balancing the language, reorder and phrase models to find the global maximum of a metric is likely to take far too long, therefore we must find a local maximum using a greedy algorithm. The definition of a greedy algorithm is contested, but in this instance we mean an algorithm that chooses the best local step (rather than the global best step that a brute-force algorithm will take).

We do need a metric to attempt to maximize, this metric essentially tries to quantify how good a translation is. It is impossible to create a perfect metric to quantify how objectively good a sentence is, therefore this metric is just an estimation. This metric is a combination of our phrase, reorder and language models. Lambda in this case will refer to the weights of each model:

$$score = \exp\left(\lambda_1 \log p_{phrase} + \lambda_2 \log p_{ngram} + \lambda_3 \log p_{reorder}\right)$$

Stack Decoding will be our method of decoding. Essentially it involves having stacks representing the number of translated words, starting with a stack of 0 words with an empty sentence in it. We iterate through the stacks one by one, finding possible phrase translations in the stack, the new sentences with their new phrases are then added to their corresponding future stack (depending on how many words they have translated). We then iterate through the next stack until a set of compete sentences are formed. We use our metric to evaluate how "good" a given translation is, this allows us to pick the best translation, as well as prune the stacks as will be explained later.

Having implemented stack decoding, we can see it's still very slow, which we cannot afford. Taking upwards of 20 minutes per translation task would mean to get a small dataset of 1000 words for evaluation would take 2 weeks, which is far too long considering we need to perform this step twice (and then implement the other methods). Ideally, we'd like to take advantage of multiple CPU cores in the training and tuning of our model, but this would significantly complicate our implementation, as python generally runs on one CPU core (without use of packages like "multiprocessing"). So we need a method to speed up the decoding process that doesn't involve multi-threading, we will do this using pruning.

Pruning essentially means removing low quality paths from our prediction process, thus removing them from all future searches too. There are a few pruning methods, some look for low quality predictions based on the average score of predictions, but in this case we will implement a number cap to our predictions. Essentially, if a prediction is added to a stack, and the number of predictions in that stack are then greater than N, remove the lowest quality prediction. We can also "prune" our phrase table for additional speed, setting an arbitrary limit to how low a probability of a phrase translation can be before it is ignored (we could do this after training, but then we couldn't tweak our limit without retraining).

6.6 Hyperparameter Tuning

Hyperparameter tuning for this model is difficult, as to tune a model, translations need to be generated quickly. The slow speed of translations from this model have required that we try to tune as efficiently as possible, with the minimum number of iterations. To find an approximate value for hyperparameters, an algorithm like Simplex can be used, this is ideally the way we'd like to tune the whole model, but performing a large amount of tuning this way would be incredibly slow.

We will use BLEU scores to measure how well the current hyperparameters are performing. We unfortunately cannot tune in the context of one sentence, this will overfit the model to that one sentence. Calculating a BLEU score for a random sentence from the validation set will result in vastly varying measures of success depending on ease of translation for that specific sentence. Therefore, we need to translate a small set of sentences and calculate the BLEU score across them, ideally we'd use a large set of sentences but increasing the size of this set will exponentially increase the tuning time. 3 sentences will be translated within each iteration of the tuning process, this is simply because anymore would make tuning unfeasible.

Prior to applying Simplex's algorithm, we will attempt to speed up tuning by finding initial weights for our hyperparameters using a random grid search. A random grid search takes a list for each weight and randomly picks a hyperparameter from each list N times, it then returns the best of the randomly selected sets of hyperparameters. This can be thought of a broader brush, attempting to find a vague area that we can then finely tune using Simplex's algorithm. We will try out 10 sets of random hyperparameters in our tuning process.

One last way to optimize the tuning process, or at the very least reduce the amount of time spent tuning, is to only select relatively short sentences as they are quicker to translate. The maximum sentence length was 25 words long. This is likely to lead to worse performance on longer sentences, but will allow for more tuning.

For Simplex's algorithm we will use an implementation included in the Scipy python library. We pass the tuning algorithm the function we want to tune and the weights to start from, it then tunes N times, returning the new hyperparameters after completion. In this case, 100 iterations of this algorithm are performed, with each doing 3 translations, taking about 5 hours.

7 Implementing a Neural Translator

Many outlines exist on implementing Neural Translators in the PyTorch library, we will be using an outline [31] based on the paper proposing Transformer models using Attention [14]. The difficulty of following this implementation comes with inputting the datasets, through researching Machine Translation systems in PyTorch, it seems that implementations generally use a library like Multi30k from the PyTorch TorchText module, these are preformatted for PyTorch, allowing for models to easily be trained using them. Attempting to format and prepare an external dataset for usage in PyTorch presented a challenge.

Preparing the corpus involved tokenizing the text, sorting the text based on length, adding sentence start and end tokens, converting text to lowercase and then converting the corpora to PyTorch's dataset format to put them into TorchText fields.

Complete corpora could not be used as it would exceed the memory available on the GPU, a solution to this would be some form of data streaming from a hard-drive or SSD to the VRAM on the GPU. For this reason, each language will have a training set of 150k.

The Model's architecture will take this form:

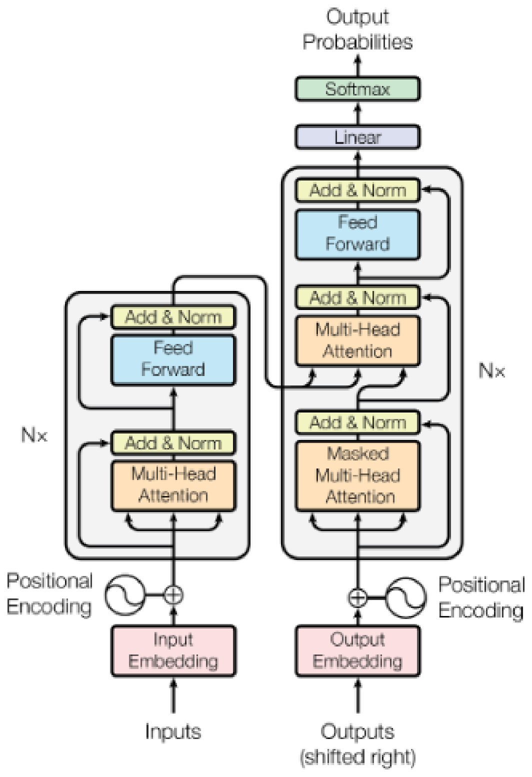

Figure 4: Diagram from ”Attention Is All You Need”[14]

As you can see, after every layer we perform a normalization step (”Add & Norm”). This is done to reduce the training time and uses a technique known as bath normalization, which takes a batch of training cases and computes the mean and variance within that batch, it is then used to normalize the input on each training case [32].

7.1 Encoder

The encoder model first will embed the sentence words into an embedding space, mapping a word into a vector. A word can have multiple different meanings, and thus we use a ”Positional Encoding” to give the vector the context of the word based on the position of the word in a given sentence.

The Multi-Headed attention is used to determine how relevant a given word is to other words in the sentence. As stated in the literature review, this allows for context to be given for words when translating it.

The generated attention vector is then passed to a Feed-Forward network, it is applied to the attention vectors to translate it into a form that the decoder can better understand. The N in the diagram represents the number of layers the Encoder has, as it will go through multiple Attention units before being passed to the decoder (N Multi-Head Attention units to be precise).

Transformers are powerful because they can run in parallel, we can pass every word in a sentence simultaneously to our encoder network to generate a vector for each word and then pass each of these vectors to our decoder simultaneously.

7.2 Decoder

While in training, we must also encode the target sentence into vector space. This will allow us to compare the output to the true result and modify the network accordingly to attempt to output results more aligned with the true result. The embedding, positional encoding and masked multi-head attention are much like the encoder, and can be skipped when testing the network as they are only used in its training. It is important that we mask the future words from the true translation, as if the network could see the words occurring next, no training would take place.

The attention from the encoder and decoder (if training) is then passed to another attention block, which is responsible for computing how related each word is across languages, this allows the model to generate a ”map” to determine the relation between words in each language are. It is then passed to a feed forward unit to become easier for the next processor block to use (this process is repeated N times for each layer much like the encoder) or to pass to the Linear block, which expands the dimensions to fit the vocabulary size of our target language, and Softmax block to convert our vectors to a probability distribution and to present output probabilities.

8 Google Translate

For evaluating Google Translate the python library ”google_trans_new” [33] can be used. There seem to be some issues with many of the python libraries for translation, including the more popular library ”googletrans” due to API changes at Google. For the selected library, we needed to download it from GitHub rather than using pythons package manager, as it seems the pip version of this library has broken recently. Due to these issues, it should be noted that the code intended to fetch these translations is prone to stop working in the future if there are further changes to Google's API. This library allows us to fetch translations from Google Translate, allowing us to automate the translation of sample text for evaluation.

It should be noted that for evaluation we will be working with smaller sets of testing data. This is because there is a limit to the amount of translation requests we can perform to the Google servers, this limit seems to range, sometimes allowing for up to 2k translations, but usually we received 1200 translations before being temporarily timed out by Google. Therefore, the quantity of translations to evaluate will only be 1000 sentences.

9 Moses

The Moses website describes a baseline system for translation and the steps needed to implement this system [34]. The baseline system uses GIZA++ [35], which we will use rather than fast align. This will not cause significant issues, as we are not directly comparing the individual systems of machine translation, but rather how each system translates the two chosen languages.

Although statistical translation does not generally take place on the GPU, the Moses toolkit can take advantage of multiple CPU cores in its training and tuning, vastly speeding up processing time compared to a single-threaded program. Along with a variety of techniques that the software takes advantage of to speed up the training, validation and testing steps the toolkit is bound to run far faster than any single-threaded statistical system we can implement in 3 months, thus making it a good control for statistical translation as it is quick to set up and get high-quality translations.

When performing tuning, my models couldn't achieve the full 8 iterations of tuning across the dataset due to errors we were unable to resolve, therefore we will settle for 5 iterations of tuning. Although it will likely underperform compared to a more finely-tuned system, as long as both models are given the same number of iterations, it will be completely fair to compare the two.

10 OpenNMT

OpenNMT is an open source ”ecosystem” (toolkit providing all tools needed) for Machine Translation [36]. It is available in both PyTorch and Tensorflow implementations, each with their own advantages and disadvantages but ultimately the results should be similar, with the OpenNMT website stating that the PyTorch implementation eases use while the Tensorflow implementation is modular and stable. We will be following a listed implementation, so the modularity is inconsequential, and as we already have a working CUDA PyTorch environment set up, the PyTorch implementation is likely the better choice. The GitHub page for OpenNMT-py [37] includes details for a basic implementation of a translation model, which we will be following much like for Moses.

The training time of this model was long, and this was using a relatively small vocabulary size of 10k words. The models could have definitely benefited from an increased vocabulary size, but this only increased the amount of time required for training the model significantly.

11 Preparing to Demonstrate

As it is, neither our Statistical nor Neural implementation can easily be demonstrated. To demonstrate both, we will construct a simple User Interface that allows for translation using the Python Tkinter library. We will simply have two text boxes and a ”translate” button that translates the text from one text box and places it in the other. Additionally, we implement a ComboBox that allows the user to select the language. For additional demonstration purposes, the Statistical implementation will have fields to modify the weights for the score metric (weights for n-gram, reorder and phrase table).

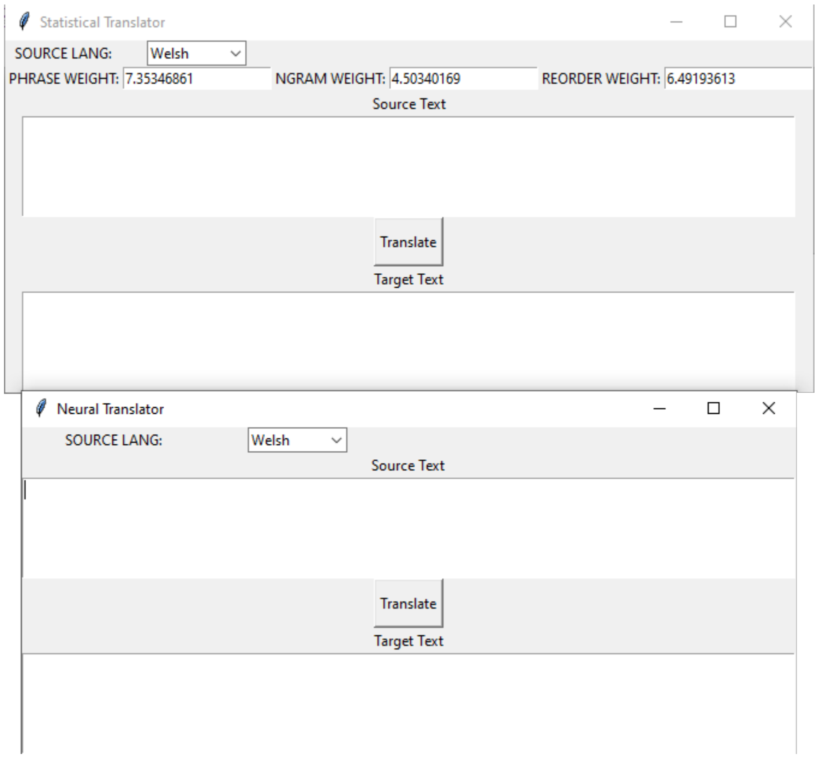

Figure 5: User Interface

This allows for simple demonstration of translation for both models. It will also allow us to test specific sentences for evaluation later.

12 Evaluation

Unfortunately, evaluating these models to a high level of accuracy will be impossible. High quality datasets including multiple possible translation candidates could not be found for the Welsh language. For the sake of fairness, we will not be sourcing a multiple candidate corpus for the other languages either. This is likely to decrease our overall scores for each of the models, so it should be kept in mind that each of these models likely perform much better than their resulting scores in this paper. It should however, give an idea about how well each of the models perform compared to each other. However, higher vocabulary languages may be prone to performing even lower due to the lack of multiple candidates in translation.

We have also been limited in how many translations we can generate in both the Statistical model and Google Translate. The Statistical model is unable to translate 75k sentences in any reasonable amount of time, and the Google Translate model, as stated earlier, is unable to translate large quantities without being timed out. Due to this, we will use a reduced testing set for each of these models. Both models will translate only 1000 sentences. We will be able to compare this against other models to see if there is an impact in how successfully the languages perform relative to one another, but we suspect that this has little to no impact on the results.

PyTorch-ignite is a high level library intended to make model training and evaluation easier. This library includes a BLEU metric, that we will use for evaluation. Both NIST and METEOR metrics will use the NLTK (natural language toolkit) library for their metrics respectively. NLTK has a BLEU metric too, but by this point we had already experimented and implemented the BLEU metric in PyTorch-ignite, and it should matter very little which implementation we use as they should perform the same (perhaps one could be faster?). The total score of a model is the mean of all its scores for each sentence.

13 Results

13.1 Training Observations

Whilst training each model, one observation made was that Welsh was by far the fastest model to train on each of the models. The reason for this is likely due to the vocabulary size of each of the languages. Welsh, from our observations has a far smaller vocabulary size compared to other languages. This observation seems to contradict our initial hypothesis somewhat, as we believed that Welsh's concept of Mutation would lead to a vastly inflated vocabulary size, but this is simply not the case. The vocabulary size for each of the languages for the "Neural Implementation" training set (150,000 sentences) were:

Language	Vocabulary Size
Welsh	22,181
French	30,415
Spanish	35,811
Danish	40,828

We had anticipated that the Danish language would have a massive vocabulary, the suffix system ("the" being added at the end of the word) inflated the number of words massively. We had expected for the Welsh results to be along the same lines, but we may only conclude that Welsh mutation occurs far less frequently than Danish suffixes and conjugation. Because of this, it is likely that the model will have difficulty translating mutated sentences.

We had not expected that the Spanish vocabulary would be so large, upon further investigation it does seem that there are situations in which a word will change. Some pronouns can be added as a suffix to verbs, for example the word *"amar"* (*love*) can have a suffix when used in the context "to love you" and become *amarte*. Certain adjectives can be added as suffixes too, with *"interesante"* (*interesting*) can gain a suffix and become *"interesantísimo"* for *"very interesting"*. This explains its inflated vocabulary.

French also seems to have multiple forms of inflection, verbs mutating heavily depending on their context. For example the French verb *"savoir"* (*to know*), becomes *"je sais"* (*I know*) or *"nous savons"* (*we know*).

These vocabulary sizes are a surprising result, clearly mutation has had a far smaller impact on vocabulary size. Differences in root word vocabulary size likely has some impact here, there are likely differences in how many words are in each language, but it is unlikely that this would have an impact to this extent across a fixed set corpus of similar language. We will conclude that Welsh mutation occurs far less than similar grammatical structures in other languages. The political nature of the corpuses could have also impacted how frequently Welsh mutation occurs, as the use of words that can trigger mutation could occur more frequently in casual conversation. Our best hypothesis is that the other languages have some form of word modification that occurs more generally; Spanish and Danish conjoining adjectives etc.

It will be interesting to see the correlation between vocabulary size and BLEU scores. It is likely that a larger vocabulary results in lower BLEU scores for a few reasons, one being that a system will be less confident in any given translation if given more options. A larger vocabulary will also make measuring model accuracy difficult, more words will mean more "correct" translations which will mean more opportunities for the model to present results that vary slightly from the true translation in the training set. This problem is made worse by the fact that we only have one possible translation for each sentence in our testing set, which will likely make automated evaluation metrics correspond less with human judgement. There is very little we can do about this, as stated earlier, we were unable to find good corpora for the Welsh language that also included multiple translation candidates. This will likely impact Google Translate the heaviest, as it was trained on a general corpus rather than a politically worded corpus, therefore it is more likely to propose more casual language which will be considered "wrong" by automated metrics.

We chose the Neural implementation to evaluate Danish and Spanish because it was relatively fast to train, and although it likely could have performed better with more iterations, layers, and a larger training set, this would have resulted in more training time, which isn't easily afforded when training 4 separate models. All other models took significantly longer to train (other than the statistical, which takes too long to translate), but, hopefully result in higher scores on the automated metrics.

13.2 Model Scores

	BLEU		METEOR		NIST	
Model	CY	FR	CY	FR	CY	FR
Google Translate*	0.4363	0.2272	0.7429	0.5764	9.3915	7.1803
OpenNMT	0.4806	0.2487	0.7608	0.5739	12.0009	8.3804
Moses	0.4128	0.2373	0.7451	0.5935	11.1515	8.0743
Neural	0.3553	0.1672	0.6666	0.4986	10.1124	6.7237
Statistical*	0.1133	0.0572	0.4495	0.3447	5.2797	3.9262

** - indicates a reduced training set (1000 sentences)*

As you can see, universally, across all metrics and models, Welsh vastly outperforms French. This contradicts our initial hypothesis but would align with the vocabulary sizes of each language. Our statistical implementation performed far worse than expected, likely due to the poor reorder metric (distance based reordering), a technique like lexical reordering would likely improve this score, more tuning time is also likely needed to find better hyperparameters.

Google Translate performed far better than expected, we hypothesized that single translation candidates and the fact that it is intended for all-purpose language translation would result in significantly reduced BLEU scores, this is clearly not the case and while it is likely that it has been under-scored compared to other metrics, it has still managed to beat many of the specifically trained models and achieve a relatively good score.

Using MatPlotLib we can graph our calculated BLEU scores for better illustration:

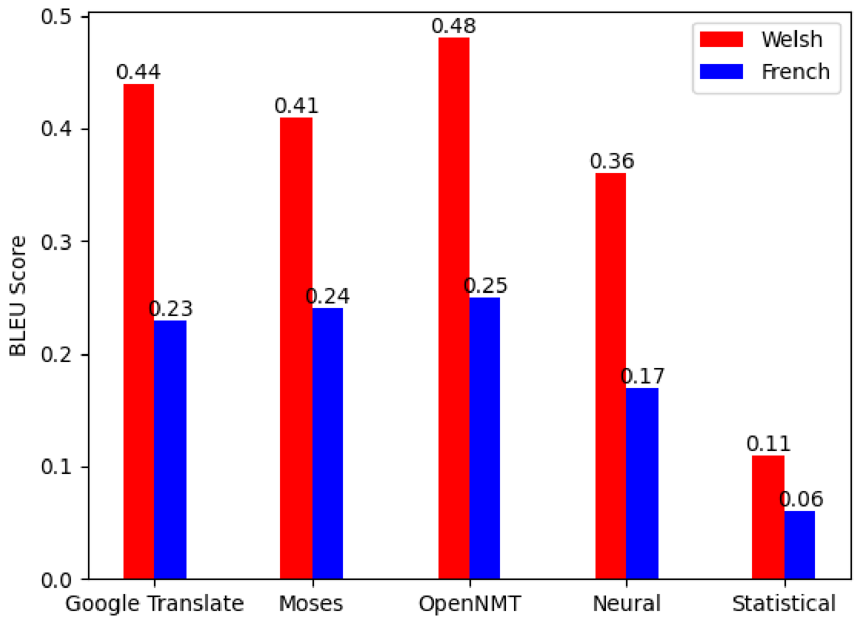

Figure 6: Graphed BLEU scores of each model

The models have performed in line with general academic consensus, with Neural implementations performing better than their statistical counterparts. While this isn't a fair comparison as each have different corpus sizes and training time, it does suggest that our models are reliable. Naturally, the models we have implemented have performed worse than the Toolkits and Google Translate, this is to be expected as they are state-of-the-art models developed over years. We are confident however that given more time to train and a larger corpus (VRAM data streaming needed), our neural implementation could have performed far stronger.

The variation between the performance of the Welsh model and the French model stayed consistent, implying no significant difference in ease of translation and model type. It is interesting that the French Moses model performed better than its Google Translate counterpart, but this can easily be dismissed as being caused by the larger French vocabulary causing reduced scores as well as the all-purpose nature of Google Translate.

It should be stated again, that a lack of French language understanding will prevent us from fully justifying why a French model has performed a certain way. Likely, it is primarily to do with vocabulary size, with French having a vocabulary 37% larger than Welsh'. Evaluating the performance of French and Welsh compared to the additional two languages will allow us to determine if vocabulary size is the sole reason for poor performance or some other convention or grammatical structure is impacting the performance.

To examine performance across different word counts, we have graphed our BLEU score for different sentence lengths, this will allow us to look for differences in the handling of longer sentences.

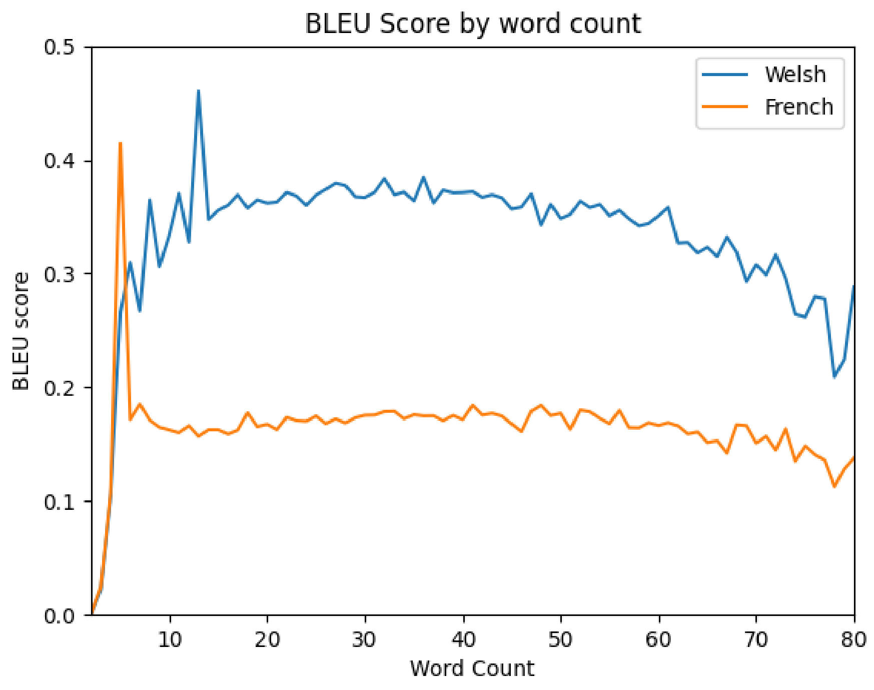

Figure 7: Neural BLEU score for different sentence word lengths

As you can see, there is relatively little loss in BLEU score, this is due to the explained concept of Attention, providing important context to our Neural Model. Some BLEU score drop is to be expected, this is because as a sentence gets longer, the more possible permutations of a "correct" translations there are. One translation candidate for evaluating BLEU scores increase the effect of this problem.

There are a few large spikes on this graph, through investigating the corpus we have concluded that these are repeating phrases that the model learns well and then can simply translate directly such as *"I propose amendment 3"*. The BLEU score is also initially very low, as there are very few sentences with less than 5 words.

It should be mentioned that the corpora do not have even numbers of words across every sentence length, it is an uneven distribution. This is also likely a reason for the initial data spikes for the lower word lengths.

13.3 Language Scores

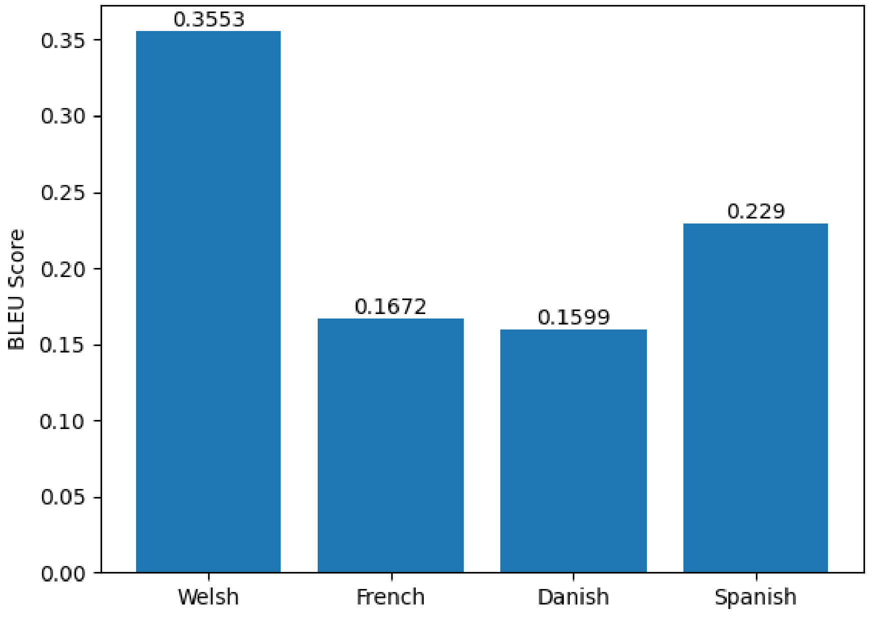

Figure 8: Graphed BLEU scores of each language for the Neural implementation

The results presented here have completely contradicted our initial hypothesis. All other languages have performed worse than Welsh. Relative to Welsh this performance is to be expected when looking at our vocabulary size, although it is unusual that the Spanish language performed so well compared the French and Danish when considering that its vocabulary size is between that of the two languages.

The performance of Spanish in this context is difficult to explain, as the vocabulary size is greater than French and yet it performs worse. As we are not linguists, we cannot easily justify why this might be. We would hypothesize that it is likely due to a grammatical structure more in line with English. It is frequently stated that Spanish is one of the easiest languages for English speakers to learn as it is a Romance language derived from Latin [38], similar word roots would not result in easier translation, but it is possible that the similar grammatical structure could result in easier translations.

13.4 Examples

In this section we will investigate how our models handle specific Welsh examples and where they may have gone wrong. We shall start with a simple example:

	1	2	3	4	5
Source	mae	’n	ddrwg	gen	i
Direct	it	is	bad	from	me
Translation	i	am	sorry		
Neural	i	am	sorry		
Statistical	it	i	am	sorry	

As you can see, in this simple example our Neural implementation performs perfectly. We believe there is a small issue with our statistical translator, in that it prioritizes sentence length over ”correctness”, it will attempt to get a translation for every input token, as more words will result in a higher score. This is a flaw in the model and would need to be fixed.

Looking at longer sentences, it is clear that the statistical method has issues with reordering:

Source:	*mae traws-gydymffurfio yn uchel ar agenda ffermwyr Cymru, gyda’r mwyafrif helaeth o blaid datgysylltu llawn sy’n ei gwneud yn ofynnol i brosesau a gweithdrefnau gael eu symleiddio a fyddai, gobeithio, yn arwain at lai o fiwrocratiaeth*
Translation:	*cross compliance is high on Welsh farmers’ agenda, with the vast majority favouring full decoupling and requiring processes and procedures to be simplified, which would, hopefully, lead to less bureaucracy*
Statistical:	*agenda along with the vast majority of plenary a requirement for a minimum and be would hopefully result in a decreased number of party farmers high is*
Neural:	*it is high on the welsh farmers’ agenda, with the vast majority of parties who require control and procedures to be simplified, leading to less ⟨unk⟩*

As you can see, the components of the correct sentence are roughly there, just in a bad order. This is due to the reorder metric, which upon experimentation with its weight, seems to not help. A new form of reordering is needed to improve the performance of statistical translation. If the reorder metric were to improve, then the language model would be able to function better, which would result in more of the sentence being translated as it would make grammatical sense.

The neural method performed quite well here, the meaning of the sentence can be understood from this sentence, even if it is quite rough. You can see the ”*⟨unk⟩*” token, which indicates an unknown word (in this case *”bureaucracy”* from *”fiwrocratiaeth”*). This is interesting, we predicted this would occur in our training observations; that it is likely that niche mutation cases can cause problems, in this case the mutation of *"biwrocratiaeth"* to *"fiwrocratiaeth"* has resulted in an unknown translation, as the model has not seen the mutated word before. Attempting to translate the un-mutated form of this word results in the correct translation.

14 Conclusion

We have, in this paper, proven our initial hypothesis wrong. While not a realistic measurement of the efficiency of each model, our measurements show Welsh scores higher in automated metrics (BLEU, NIST, METEOR), and despite only having one candidate for translation, the strong lead that the Welsh language has across these automated metrics indicate that it would likely still have a strong lead with more translation candidates. The vocabulary size also helps quantify the complexity of a translation task, where Welsh' vocabulary was found to be far smaller than that of the other languages.

Welsh Mutation, while often more complex than other forms of inflection, occurs far less often, resulting in manageable vocabulary sizes compared to other morphologically rich languages. This results in Welsh being an easier translation task than all the selected languages for this experiment.

The infrequency and variability of Welsh mutation does act in its detriment, although it does result in smaller vocabularies, it does result in issues for the hundreds of thousands of niche mutation cases. If a certain mutation does not appear in the training corpus, then it will result in an unknown word. When training a model to translate to Welsh with a high degree of accuracy, this must be kept in mind. A way to handle mutated synonyms is essential to translate from Welsh to any other language, and a model must understand mutation to translate to Welsh.

Perhaps the complexity of mutated words can be remedied with character-based approaches [16], a paper proposes a Neural Machine Translation system that can handle languages with massive vocabularies or ones that are morphologically rich. This allows for the model to handle affixes with ease. A simpler approach may be to process sentences to remove mutation prior to them being translated, mutation conveys no meaning and therefore no information is lost when translating to English (although a model would need an understanding of mutation to translate to Welsh).

The data would also suggest that systems trained to translate a specific type of text (political language etc) will perform better on that text than general systems. We would suggest that companies and individuals that use general translation models like Google Translate look into training their own model if good quality corpora are available.

We will be considering this project a success. We have set out to prove or disprove a specific hypothesis, which we have disproved successfully. Furthermore, we have successfully implemented a Neural and Statistical machine translation model, as well as trained two

Toolkits for evaluation.

15 Further Work Needed

The primary area we have found needs more work, is the compilation of high quality Welsh corpora. There are very few aligned corpuses available for the Welsh language, and we could not find any with multiple translation candidates. Having multiple translation candidates is essential to properly evaluating models using automated metrics. Without multiple translation candidates, an automated metric can only evaluate translation success against its one translation, when any given translation likely has hundreds of potential translations. This would be a very expensive task both financially and in work, as a great deal of care would need to be taken to compile a high quality corpus.

Examination of findings like these require a fluent knowledge of the used languages that we were unable to perform. Attempting to justify findings without a full knowledge of how a language works has proven to be very difficult.

Our statistical translator needed more work, but this work would have likely been wasted implementing it in Python. A complete rewrite in a low-level language like C++ would likely be needed, along with implementing multi-threaded parallel computation to speed up translation. Performing tuning and evaluation cannot be done in a reasonable time with a model as slow as our implementation. Lexical reordering is likely needed to sufficiently reorder sentences to make sense.

A Machine with better hardware would result in better model performance. It would be interesting to see if there is any variability between higher quality model performance and lower quality model performance. Perhaps as models improve, the difference between BLEU scores across languages will reduce?

Attempting to train a model to handle mutation to a high degree of accuracy in English to Welsh translation would be an interesting future project. Mutation and gendered noun conventions would present very difficult problems for models to solve.

16 References

[1] Philipp Koehn. *Statistical Machine Translation*. Cambridge: Cambridge University Press, 2009, p. 447. ISBN: 1139483307, 9781139483308. URL: https://books.google.co.uk/books?id=kKYgAwAAQBAJ&dq=Machine+Translation&lr=&source=gbs_navlinks_s.

[2] Philipp Koehn et al. “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*. 2007, pp. 177–180.

[3] Warren Weaver. “Translation”. In: *James Joyce in Context* (1949), pp. 125–136. DOI: 10.1017/CBO9780511576072.012.

[4] W John Hutchins. “Machine translation: A brief history”. In: *Concise history of the language sciences*. Elsevier, 1995, pp. 431–445.

[5] Richard Zens, Franz Josef Och, and Hermann Ney. “Phrase-Based Statistical Machine Translation Phrase-Based Statistical Machine Translation”. In: September 2002 (2014). DOI: 10.1007/3-540-45751-8.

[6] Holger Schwenk. “Syntax-based Language Models for Statistical Machine Translation”. In: *The Prague Bulletin of Mathematical Linguistics* 93 (2010), pp. 137–146. DOI: 10.2478/v10108-010-0014-6.Unauthenticated.

[7] David Chiang. “Hierarchical Phrase-Based Translation”. In: May 2006 (2007).

[8] Philipp Koehn. “Neural machine translation”. In: *arXiv preprint arXiv:1709.07809* (2017).

[9] Ondřej Bojar et al. “Findings of the 2016 conference on machine translation”. In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. 2016, pp. 131–198.

[10] Xing Wang et al. “Neural machine translation advised by statistical machine translation”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[11] Philipp Koehn and Rebecca Knowles. “Six challenges for neural machine translation”. In: *arXiv preprint arXiv:1706.03872* (2017).

[12] Nal Kalchbrenner and Phil Blunsom. “Recurrent continuous translation models”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1700–1709.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[14] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[15] Xiang Yu, Ngoc Thang Vu, and Jonas Kuhn. “Learning the Dyck language with attention-based Seq2Seq models”. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 138–146.

[16] Marta R. Costa-Jussà and José A. R. Fonollosa. *Character-based Neural Machine Translation*. 2016. arXiv: 1603.00810 [cs.CL].

[17] Einat Minkov, Kristina Toutanova, and Hisami Suzuki. “Generating complex morphology for machine translation”. In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 128–135.

[18] Roee Aharoni, Melvin Johnson, and Orhan Firat. "Massively multilingual neural machine translation". In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1 (2019), pp. 3874–3884. DOI: 10.18653/v1/n19-1388. arXiv: 1903.00089.

[19] Rory Cellan-Jones. *Google Translate serves up 'scummy Welsh' translations.* July 2018. URL: https://www.bbc.co.uk/news/technology-45012357.

[20] *Welsh language data from the Annual Population Survey: October 2019 to September 2020.* Jan. 2021. URL: https://gov.wales/welsh-language-data-annual-population-survey-october-2019-september-2020.

[21] Benjamin Screen. "Productivity and quality when editing machine translation and translation memory outputs: an empirical analysis of English to Welsh translation". In: *Studia Celtica Posnaniensia* 2.1 (2017), pp. 113–136.

[22] Myfyr Prys and Dewi Bryn Jones. "Embedding English to Welsh MT in a Private Company". In: *Proceedings of the Celtic Language Technology Workshop.* 2019, pp. 41–47.

[23] Kishore Papineni et al. "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics.* 2002, pp. 311–318.

[24] George Doddington. "Automatic evaluation of machine translation quality using n-gram co-occurrence statistics". In: *Proceedings of the second international conference on Human Language Technology Research.* 2002, pp. 138–145.

[25] Satanjeev Banerjee and Alon Lavie. "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.* 2005, pp. 65–72.

[26] Andreas Jones, Dafydd and Eisele. "Phrase-based Statistical Machine Translation between English and Welsh". In: 2006 (2006). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.3429&rep=rep1&type=pdf#page=71.

[27] Philipp Koehn et al. "Europarl: A parallel corpus for statistical machine translation". In: *MT summit.* Vol. 5. Citeseer. 2005, pp. 79–86.

[28] Farzeen Zehra et al. "Comparative analysis of C++ and Python in terms of memory and time". In: (2020).

[29] Chris Dyer, Victor Chahuneau, and Noah A. Smith. "A simple, fast, and effective reparameterization of IBM model 2". In: *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference* June (2013), pp. 644–648.

[30] Fredrik Hedman. *Phrase extraction algorithm for statistical machine translation.* Aug. 2014. URL: https://stackoverflow.com/a/25128809/2921254.

[31] bentrevett. *PyTorch Seq2Seq: Attention is all you need.* https://github.com/bentrevett/pytorch-seq2seq/. 2019.

[32] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[33] lushan88a. *google_trans_new python library*. 2020. URL: https://github.com/lushan88a/google_trans_new.

[34] *Moses Baseline System*. URL: http://www.statmt.org/moses/?n=Moses.Baseline.

[35] Franz Josef Och and Hermann Ney. "A Systematic Comparison of Various Statistical Alignment Models". In: *Computational Linguistics* 29.1 (2003), pp. 19–51.

[36] Guillaume Klein et al. "OpenNMT: Open-Source Toolkit for Neural Machine Translation". In: *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 67–72. URL: https://www.aclweb.org/anthology/P17-4012.

[37] Guillaume Klein et al. *OpenNMT-py*. https://github.com/OpenNMT/OpenNMT-py. 2017.

[38] Dylan Lyons. *9 Easiest Languages For English Speakers To Learn*. URL: https://www.babbel.com/en/magazine/easiest-languages-for-english-speakers-to-learn.

17 Appendix

Ensure that **all** python files are run from the directory they are contained in (rather than root directory).

17.1 Git Repository

The git repository can be found here:

https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2020/arm034

17.2 Statistical Translator

Running the **train_statistical.py** file will train the model.

Running **evaluate_statistical.py** to tune and generate output translation file.

Running **stat_demo_ui.py** will present a UI to perform translations. This is the recommended way to test specific translations.

17.3 Neural Translator

Running **neural.py** will train, tune and test the model. The variable **train_model** can be set to false to skip the training and load the model from the hard drive to generate translations. The **lang** variable is used to determine the language to translate from.

Running **neural_demo_ui.py** will present a UI to perform translations. This is the recommended way to test specific translations.

17.4 Graphing & scoring

Run **output/graph_output.py** (ensure it is run from the **/output/** directory) to calculate NIST, METEOR and BLEU scores and generate all the graphs used. You must close the graph to generate the next one.