UPPSALA
UNIVERSITET

When and Why Universal Word Embeddings Are Not Useful in a Pure Zero-Shot Machine Translation System

Shifei Chen

Uppsala University
Department of Linguistics and Philology
Master Programme in Language Technology
Master's Thesis in Language Technology, 30 ECTS credits

August 22, 2020

Supervisors:
Ali Basirat, Uppsala University

Abstract

The concept of *palindromes* is introduced, and some method for finding palindromes is developed.

Contents

Preface

This thesis was finished under the supervision of Ali Basirat. I would like to thank him first for his continuous guidance and inspiration.

Thank you Mr. Anders Wall and everyone in the Anders Wall Scholarship Foundation for sponsoring my Master's study. This opportunity led me to meet everyone in the Master Programme in Language Technology, from whom I have learned a lot during the 2-years journey.

Last but not least, I would like to thank my parents for their unconditional love and support. And to my girlfriend, who has always been together with me when everything was unusual and chaotic.

1 Introduction

Palindromes are fun. I've tried to find some. In Chapter 2 previous work is reviewed, and Chapter ?? is about my results.

2 Previous work

2.1 Word Embeddings

2.1.1 Representing Words in Vectors

In Natural Language Processing, people need to convert the natural representation of words into forms that are more efficient for computers to process. The idea started with statistical language modeling introduced by Bengio et al., 2003. In 2013, Mikolov, Chen, et al., 2013 introduced Word2Vec, which encapsulates words and their latent information into vectors. Besides the benefit that it simplifies representation and storage of words for computers, it also enables the possibilities to calculate words and their semantic meanings just as vectors.

Take an example of vocabulary $V = \{\text{king}, \text{queen}, \text{man}, \text{woman}\}$, if we convert these words into vectors such as

$$\begin{aligned}\vec{k} &= \text{vec}(\text{king})\\ \vec{q} &= \text{vec}(\text{queen})\\ \vec{m} &= \text{vec}(\text{man})\\ \vec{w} &= \text{vec}(\text{woman})\end{aligned}$$

We could have an equation of

$$\vec{q} = \vec{k} - \vec{m} + \vec{w} \tag{2.1}$$

It is meaningful from both the mathematical perspective and the linguistic perspective. Figure 2.1 illustrates both perspectives in a vector space that contains these four vectors. Geometrically, the angle between $\vec{k}$ and $\vec{q}$ is small, together with the angle between $\vec{m}$ and $\vec{w}$. From the cosine similarity definition below

$$sim(x, y) = \cos(\theta) = \frac{x \cdot y}{||x||||y||}$$

It indicates that the cosine similarities of every two of them are high. Besides, Equation 2.1 is correct in this vector space too. We could get such an equation from basic vector calculation definitions.

Semantically, the word “king” and “queue” are closely related in the same category as well as the word “man” and “woman”. It is meaningful to say “queue” is a “king” being replaced its “man” part by a female.

To turn words into vectors, one could use a simple one-hot encoding. Like in the example above we could make $\vec{k} = [1, 0, 0, 0]$. However, these one-hot vectors can merely capture any latent semantic information between different words. Recent vectorized word representations, or word embeddings, were learned through neural networks, such as Word2Vec, which learns word embeddings through a Skip-gram model or a Continuous Bag of Words model (Mikolov, Sutskever, et al., 2013). Both models are shown in Figure 2.1.1.

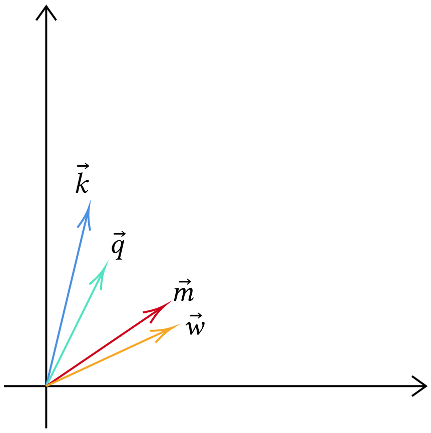

Figure 2.1: Illustraion of a vector space where Equation 2.1 exists.

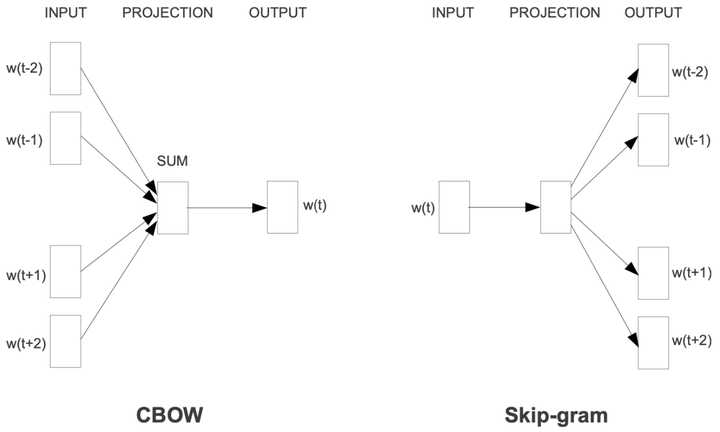

Figure 2.2: The Skip-gram and the CBOW model. (Mikolov, Le, et al., 2013)

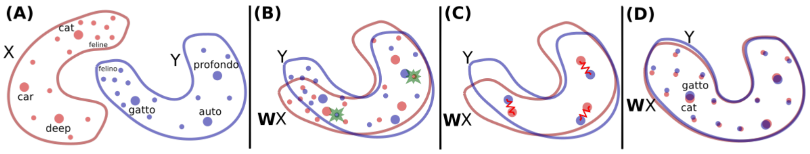

Figure 2.3: Aligning bilingual vector spaces. (Conneau et al., 2017)

The Skip-gram model

When given a target word w, the model can produce vector representations that are good at predicting the words surrounding w within the context size of C. The probability of a context word w_k given a target word w is:

$$P(w_k|w) = \frac{\exp({v'_{w_k}}^\top v_w)}{\sum_{i=1}^{|V|}\exp({v'_i}^\top v_w)} \tag{2.2}$$

Here $|V|$ means the size of the whole vocabulary from the corpus. Both v' and v stand for the input vector representation, and the output vector representation of a word (Mikolov, Chen, et al., 2013). The aforementioned one-hot vectors are used in the Skip-gram model too. They can initialize the input representation v'.

The Continuous Bag of Words model (CBOW)

The other model, CBOW, works just like the other side of the coin. It predicts the target word w based on a bunch of context words $w_{-C}, w_{-C+1}..., w_{C-1}, w_C$ within the window size C, as the formula below:

$$P(w|w_{-C}, w_{-C+1}..., w_{C-1}, w_C) = \frac{\exp({v'_w}^\top \bar{v}_{w_k})}{\sum_{i=1}^{|V|}\exp({v'_{w_i}}^\top \bar{v}_{w_k})} \tag{2.3}$$

Here $\bar{v}_{w_k}$ means the sum of the context word w_k's vectorized representation, while v'_w means the input vector representations of word w as in the Skip-gram model.

The difference between these two models is that the CBOW model predicts the target word from multiple given context words, while the Skip-gram model predicts the context words from one given center word. Hence the Skip-gram model is better at predicting rare words because all of the words are treated equally in the *word AND context* relationship. While in the CBOW model, frequent words have advantages over rare words as they will have higher probabilities in a given context. The Skip-gram model is arguably the most popular method to learn word embeddings as it is both fast and robust, especially with less frequent words in the corpus. (Levy et al., 2015)

2.1.2 Multilingual Word Embeddings

Learned from approaches like the Skip-gram model or the CBOW model, vectorized word representations tend to cluster words with similar semantics (Mikolov, Le, et al., 2013). It then becomes attractive to see whether we could fit two or more languages into the same vector space. Word embeddings that consist of more than one language are called multilingual word embeddings.

In the multilingual scenario, it is vital to align words in two different vector spaces to make word embeddings from different languages comparable. Figure 2.1.2 illustrated the alignment method from Conneau et al., 2017. Suppose there is a set of word pairs in their associated vectorized representation $\{x_i, y_i\}_{i\in\{1,...,n\}}$, the two vector spaces were aligned by a rotation matrix $W \in \mathbb{R}^{d\times d}$ as shown in process **(B)**, where we try to optimize the formula

$$\min_{W \in \mathbb{R}^{d \times d}} \frac{1}{n} \sum_{i=1}^{n} \ell(Wx_i, y_i)$$

Here ℓ is the loss function and it is usually the square loss function $\ell_2(x, y) = ||x-y||^2$. Then W is further refined in process **(C)**, where we choose frequent words as anchor points and minimize the distance between each correspondent anchor points by an energy function. After this, the refined W is then used to map all words in the dictionary during the inference process. We obtain the translation $t(i)$ of a given source word i in the formula

$$t(i) \in \operatorname*{arg\,min}_{j \in \{1,\dots,N\}} \ell(Wx_i, y_j)$$

Again, the loss function ℓ is typically the square loss function. However, using the square loss function could make the model suffer from the "hubness problem". Conneau et al., 2017 counter reacted to the "hubness" problem by introducing the cross-domain similarity local scaling (CSLS), whose detail is beyond this thesis's scope.

Despite the theoretical feasibility proven by mathematic formulas, we need data to drive the alignment process. Take bilingual word embeddings as the first step, their initial alignment data for adversarial learning of the rotation matrix W could come from a bilingual dictionary (Mikolov, Le, et al., 2013). Also, there are other kinds of alignment using aligned data from sentence level, or even document level, even though word-level information is most common. By using word-level information, we can start with a pivot language (usually English) and map each other monolingual word embeddings by looking up translation dictionaries. This mapping process could also start with vectors only, where we choose a bilingual word embedding that shares a language (also typically English) with other bilingual embeddings and choose the other bilingual word embeddings by aligning their shared language subspace. Sentence-level parallel data is similar to the corpus in Machine Translation (MT), which contains sentence-aligned texts (Hermann and Blunsom, 2013). Document-level information is more common in the form of topic-aligned or class-aligned, such as Wikipedia data (Vulić and Moens, 2013). The alignment process of multilingual word embeddings is roughly the same as bilingual word embeddings, using parallel data from either word-level, sentence-level, or document-level (Ruder et al., 2019).

2.1.3 fastText

In this work, I have chosen fastText aligned word vectors[1] (Joulin et al., 2018) as my vectorized word representation. They are based on the pre-trained vectors computed from the Wikipedia corpus using fastText (Bojanowski et al., 2016). Not only because of the soon to be talked good performance, but fastText aligned word embeddings also has a large selection of languages available to use out of the box. It is ideal for implementing the cross-lingual experiments on aligned word embeddings.

fastText is an extension of the original Word2Vec methods, which uses sub-words to augment low-frequency and unseen words. Take word `low-key` as an example. As a whole word, its possibility in a given document would be much lower than its components, `low` and `key`. fastText learns its vectorized representation from a smaller n-gram sub-word level. It divides the whole word into sub-words units as below if we assume $n = 3$

[1] https://fasttext.cc/docs/en/aligned-vectors.html

<lo, low, ow-, w-k, -ke, key, ey>

Each sub-word has its own vectorized representation learned through a CBOW or Skip-gram model as in Word2Vec. The word vector for the whole word unit <low-key> is then the sum of all of its sub-word units' vectors. Hence its rareness would be compensated by two more frequent subwords low and key, even if it might not appear in the training document at all.

In terms of multilingual alignment, fastText improves the standard solution to the hubness problem by directly including the Relaxed CSLS (RCSLS) criterion into the model during both the learning and the inference phase. Before the work of Joulin et al., 2018, inverted softmax (ISF) Smith et al., 2017 or CSLS (Conneau et al., 2017) was only used in the inference time to address the hubness problem while square loss is still the loss function used in the training time. However, since both the ISF and the CSLS are not consistent with the square loss function in the training time, they will create a discrepancy between the learning of the translation model and the inference. According to the authors, fastText outperforms other alignment approaches by 3 to 4% on average.

2.2 Multilingual Neural Machine Translation (MNMT) Systems

2.2.1 Multilingual Neural Machine Translation

Neural Machine Translation (NMT) uses neural networks to learn the translation relationship between a source and a target language. Its power has gone over the traditional Statistical Machine Translation (SMT) and enabled things that were impossible in the past. One of that is Multilingual Machine Translation, as shown in the Figure 2.2.1, it uses the same attentional encoder-decoder model as bilingual NMT but trains it on a multilingual corpus (M. Johnson et al., 2016). The benefit of such a multilingual system does not necessarily stop at higher translation performance between common languages like English, French, or Spanish; it also leverages additional information from high resource languages to low resource languages in the same semantic space (Ha et al., 2016). People identify this information leverage as a special form of transfer learning (Zoph et al., 2016), which could happen both in the horizontal or vertical direction (Lakew et al., 2019): In the horizontal direction, knowledge transfers from pre-trained data (such as word embeddings or language models) to the raw test data; in the vertical direction, knowledge transfers from languages to languages, which could either transfer from high resource to low resource languages or from seen languages to unseen languages. The latter is called zero-shot translation.

2.2.2 Zero-shot Machine Translation Systems

Zero-shot translation stands for translation between language pairs invisible to the MNMT system during the training time. E.g., we build an MNMT system with training language pairs of German-English and French-English while test its performance on a German-French scenario. In 2016, M. Johnson et al., 2016 first published their result on a zero-shot MT system. Their multilingual MT system, including an encoder, decoder, and attention module, requires no change to a standard NMT system. The only modification is in the training corpus, where they had introduced an artificial token at the beginning of each source sentence to denote the translation target language. Ha et al., 2016 also showed that their universal encoder and decoder model is capable of zero-shot MT. The concept of translation between unseen language pairs is attractive, especially for

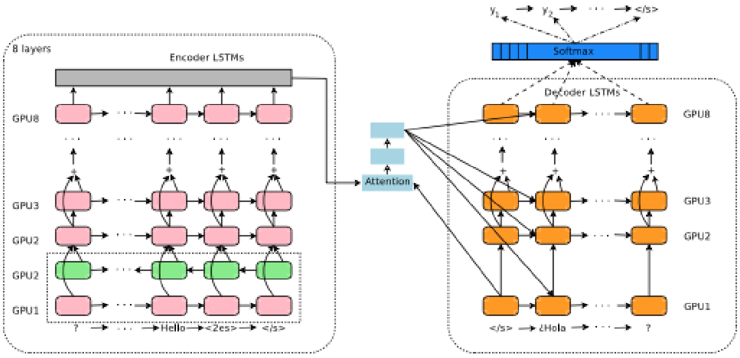

Figure 2.4: Google's MNMT Architecturef from (M. Johnson et al., 2016)

.

low-resource language pairs, though these two models both underperformed than a pivot based system.

There are two reasons that could explain the gap between a zero-shot system and a pivot based system, language bias (Arivazhagan et al., 2019; Ha et al., 2016, 2017) and poor generalization (Arivazhagan et al., 2019). Language bias means that during inference, the MT system tends to decode the target sentence into the wrong language, usually copying the source language or the bridging language Ha et al., 2016. It could be the consequence of always translating all source languages into the bridging language, hence make the model difficult to learn to translate the desired target language (Arivazhagan et al., 2019).

The other potential reason for the worse performance of a zero-shot system is poor generalization (Arivazhagan et al., 2019). When a zero-shot system is trained purely on the end-to-end translation objective, the model prefers to overfit the supervised translation direction features than learn more transferable language features.

To fix these two problems, there has been work on improving the preprocessing process (Lakew et al., 2018), parameter sharing (Blackwood et al., 2018; Firat et al., 2016), additional loss penalty functions (Arivazhagan et al., 2019) and pre-training modules using external information (Baziotis et al., 2020). In some cases, zero-shot system could achieve better performance than pivot based systems.

2.2.3 MNMT Systems Based on Word Embeddings

Like in Section 2.2.2, in cases where people need to translate from or into a low-resource language, they usually find it challenging to locate enough parallel data that consists of such kind of less common language. In addition to shared encoder and decoders between languages, if we could build up a vector space with word embeddings from different languages aligned, we could further leverage the similarity of word embeddings to compensate for the lack of parallel data (Zou et al., 2013). We could find unseen words in the training data buy looking for their neighbors in the vector space.

There are successful applications of pre-trained word embeddings in a MT system, such as the embedding layer in an MT system (Artetxe et al., 2017; Neishi et al., 2017), the substitution of a supervised dictionary (Conneau et al., 2017), or an external supplementary extension Di Gangi and Federico, 2017. There are even cases where successfully trained a machine translation system using very little or none parallel data (Conneau et al., 2017). Nevertheless, in most MT systems, using pre-trained word embeddings purely as the embedding layer will not outperform other models such as Transformers (Vaswani et al., 2017) and its other evolutions, largely because the training data for an MT system is usually several orders of magnitude larger than the monolingual pre-trained word embeddings. Typically pre-trained word embeddings are mainly introduced in MT systems dealing with low-resource languages.

For NMT systems focused on low resource language, Qi et al., 2018 looked into the question of when and why are pre-trained word embeddings useful. They found that pre-trained word embeddings are consistently useful for all languages. The gains would be more visible if the source and target language are similar, such as languages within the same family. Also, pre-trained word embeddings work well only on MT systems with moderate performance. Pre-trained word embeddings can not work when there is not enough data to train a basic MT system. Finally, aligned word embeddings are useful in a multilingual MT system. For bilingual MT systems, pre-trained word embeddings do not necessarily need to be aligned.

Moreover, aligned word embeddings do not work well for morphologically rich languages such as Russian and Belarusian. Qi et al., 2018 argues that this may be mainly due to the sparsity in the word embeddings files. Plus, most of the previous works target zero-shot language pairs, not on completely unseen languages. For language pairs $A \longrightarrow \text{EN}$ and $\text{EN} \longrightarrow B$, they are all interested in the unseen language pair $A \longrightarrow B$. For language pairs that include an unseen language C, whether it is on the source side, or the target side, it remains to see how universal word embeddings could help translate in this scenario.

3 Error Analysis in a Word Embedding Based MNMT System

In this chapter I will perform experiments in a universal word embedding based MNMT system. Then I will analysis its results to show why such kind of system failed to translation compeletly unseen languages depite its theoratical feasibility.

3.1 Theoratical Feasibility

As mentioned in Chapter 2, Mikolov, Le, et al., 2013 showed that there is a linear relationship between similar word embeddings in different languages. For each word pairs, assume their vector representations are $\{x_i, y_i\}_{i=1}^{n}$, we could calcualte a transformation matrix W such that Wx_i approximates to y_i. Mikolov, Le, et al., 2013 also showed their result in the word/phrase translation task for suck kind of approximated word embedding mappings. For some subsets of words, around 70% of word embeddings are exactly matched with each other by calcluating the Precision@5 score. If the threshold for the cosine similarity $\max \cos(Wx, y_i)$ being loosed to 0.6, the Precision@5 score would be as high as 90%.

In order to convert words into vectors to be calculated in the neural network, NMT systems should treat each word as word embeddings. The value of these word embedding could be learned directly during translation, but then the initialization is a crucial step as poor initialization could lead to slow converge or worse local mimima (Glorot and Bengio, 2010). The situation could be even more challenging when transltion with very few parallel corpora, since there is no data to help the embedding layer to converge to its ideal state. Hence the aforementioned word embedding mapping technique becomes appealing.

Qi et al., 2018 explored how effective it is by using aligned pre-trained word embeddings in a NMT system. They found that regardless of languages, alignment is useful as long as it's applied in a multilingual setting. They believe that since both the source and the target side vector spaces are already aligned, the NMT system learns how to transform the simialr fashion from the source langauge to the target language.

Therefore, translating a compeletely unseen language can be viewed as the question below — Given a vector space Z that consists of aligned word embeddings $\{a_i, b_i, c_i, ...\}$, how much does the NMT system knows about an unseen language A if it was only trained on the remaining languages? In theory, since the word embeddings are clustered by their semantic meanings in the vector space Z, we should be able to build loose mappings between each of the semantic centers from both the source side and the target side. The generalization ability of the system is the key to answer this question. Hence I conducted some prelimentary experiments below.

3.2 Experiment Settings

To get a basic multilingual MT system running, I chose English (EN), Germen (De) and French (FR) to be my training languages. Let C donate the final corpus, l donates the langauge specific corpus fragment and Z is the set of correspoding candidate langauges, $Z_{TRAIN} = l_{EN}, l_{DE}, l_{FR}$. I picked up Swedish (SV), Hungarian (HU) and Hebrew (HE) being my test languages, therefore $Z_{TEST} = l_{SV}, l_{HU}, l_{HE}$.

For each experiment, a basic MNMT system is trained using a training corpus C_{TEST} with all three training languages , including all six directions from the cartesian product without duplicates

$$C_{\text{TRAIN}} = \{x \times y \mid x, y \in Z_{\text{TRAIN}} \text{ and } x \neq y\} \tag{3.1}$$

It is tested on the test corpus with all three training languages and one of the test language, consist of bidirections of three different training language to the only test language.

$$C_{\text{TEST}} = \{(x, y) \cup (y, x) \mid x \in Z_{\text{TRAIN}} \text{ and } y \in Z_{\text{TEST}}\} \tag{3.2}$$

I designed the experiments and picked up the training and target languages based on the following aspects.

Language Similarity

In the work Qi et al., 2018 the authors mentioned their observation that pre-trained word embeddings are useful for languages from the same language family, the closer their relationship is the higher the performance improve is. For aligned word embeddings, if it is applied in a MNMT system consist of langauges from the same language family, it will also be benificial.

Shared Alphabets

In a typical word embedding based NMT system without subword encoding, it uses a word to index mapping to look up a correspoding word embedding for each word in the text, and a reverse index to word mappding to reconstruct the human readible text from its inferrencens. During the whole process, every word is treated as a whole, no subword segment is available. This is different than a Transformer system which subword encodings like BPE or sentencepiece are commonly used. Although fastText word embeddings were learned by using subword information (Bojanowski et al., 2016), in the final representation form all of the subword informatino is no longer available. Hence the NMT system could learn a rough mapping from semantics in the source language to the target language in a common vector space, it might see a big drop for distant languages that don't share a common alphabets.

Word Order

For a word embedding based NMT system, syntactic information lies compleetely in its hidden layer. This again differs from a Transformer system. Transformer systems learn both the lexicon, syntactic and semantic information all together in their hidden layers. Hence it remains to see how word embedding based MNMT performs on languages with different word orders, e.g. SVO versus SOV languages.

3.2.1 Corpus and Preprocessing

I used the TED talk subtitile corpus from Qi et al., 2018 [1] to train my NMT. The whole corpus has roughly 270000 sentences was splited into three parts, train, dev, test at the ratio of 0.95 : 0.025 : 0.025.

[1] https://github.com/neulab/word-embeddings-for-nmt

To build up the corpus for each experiment, I have modified the original script from Qi et al., 2018 and added a few customized features. In short, the script will extract the common sentence from each part of the splited corpus to form up a common intersection used in training, developing and testing. Since our experiments consists of languages that are relatively common in the TED project, this fine tuned corpus isn't too much different from the original corpus, hence the size for the train, dev and test split are still kept afterwards.

For preprocessing, since the orginal TED corpus is already tokenized by Moses, I then turned all of the text into lowercases and applied a sentence lengh filter to remove any long sentence that have more than 60 word to prevent bad perfromance in training. After that, when building the i2w and w2i index for the pre-trained embeddings, I have also remove any words that are less frequent than 2 times to stop the system from overfitting with too much low-frequency words. All of the preprocess function are built upon the built-in XNMT preprocess features (Neubig et al., 2018).

3.2.2 Neural Network

For the neural network I used a modified version of the neural network from Qi et al., 2018, which is built with XNMT Neubig et al., 2018. I have doubled the encoding layer to a 2-layer-bidirectional LSTM network and added the accuracy score as a evaluation metric alongside the BLEU score (Papineni et al., 2002). Everything else are kept from the orignal experiment settings, including a encoder-decoder model with attention (Bahdanau et al., 2014) and a beam size of 5, trained using batch of 32 and the Adam optimizer (Kingma and Ba, 2014). The initial learing starts at 0.0002 and decays by 0.5 when development BLEU score decreases (Denkowski and Neubig, 2017).

3.2.3 Embeddings

The embeddings I used are fastText aligned word embeddings[2]. They are based on the pre-trained vectors on Wikipedia[3] using fastText (Bojanowski et al., 2016). The alignment is performed using RCSLS as in Joulin et al., 2018.

Each of the fastText word embedding file is language specific and contains word embeddings in 300 dimensions. I concatnated different language files To build up multilingual word embedding files for the MNMT system. If there is a shared word w with two different vector values $\vec{v_a}$ and $\vec{v_b}$ in different embedding files, I will add those two vectors up and pick the average value $v_{\vec{mean}}$ as the new vector.

$$v_{\vec{mean}} = \vec{v_a} + \vec{v_b}/2 \tag{3.3}$$

In this way, there are possibilities that both of the unique semantic values in the two words w_a and w_b could lost, as there are cases that word with distant meaning share the same spelling in different languages. But it could also be argued that many word with the same spelling do have similar meaning. For example the word `café` means the same thing in both English and French, as English borrowed that word from French. Later in the experiment I also tried a different approach where I treat each word as a unique word despite they might share the same spelling, both of the results are shown below.

[2] https://fasttext.cc/docs/en/aligned-vectors.html

[3] https://www.wikipedia.org/

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

Table 3.1: Initial results for SV, HU and HE on the baseline system (Target language annotation only, dropout=0.3, trained on mixed language branch corpus.)

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

Table 3.2: Results for different language annotations (Target only, source only and full annotation))

3.3 Results and Analysis

Before I got the results, I anticipated that among all three test languages, Swedish will get the best result and it could be on the same level as the baseline MNMT consists of EN, DE and FR only. The other two languages will not get any close to the performance of Swedish, and they could go down to less than 10 BLEU scores.

However, in the results shown in Table 3.1, all of the three languages got very low BLEU scores. Swedish, though it indeed was the best of three, only achieved about 3.5 BLEU score. The other two language, Hungarian and Hebrew, didnt even go abobe 1 BLEU score. The surprisingly low results made me curious and would like to check what could be improved.

3.3.1 Altering the Source/Target Language Annotation

The first improvement I tried is to alter the way the target langauge annotation in the source sentences. In the corpus building script I add a custom `__{lang_id}__` token at the front of each source sentence as suggested in M. Johnson et al., 2016. A sentence in the processed source text that needs to be translated into German would look like

```
__de__ And we struggle with how to deal with them .
```

I added two other annotation — the source token and source token together with the target token, into the experiments. Hence a sentence in English would look like this

```
__en__ And we struggle with how to deal with them .
```

When it needs to be translated into German, the annotation would then become

```
__en__ __de__ And we struggle with how to deal with them .
```

The results are shown in Table 3.2. As it shows, source token didn't change the overall result a lot but removing the target token would drastically lower the final BLEU score. This indicated that in a MNMT system, the suggestion that the source language will be learned automatically by the system during training is correct. People should only annotate the target language into the final result only.

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

Table 3.3: Results for langauge similarity. Three other Germanic languages DA, NL and NO were added one by one into the training corpus)

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

Table 3.4: Results for source word token.)

3.3.2 Analysising on the Effect of Language Similarity

It is anticipated that Swedish will perform better the Hungarian and Hebrew in this experiment settings. To deeper undetstand the question, I have further designed three more experiments to analysis the effect of language similarity.

The additional experiments will still use Swedish as the test langauge while remove French as the training language, since it is the only training language that doesn't below to the same language branch as Swedish. (English German and Swedish are all Germanic languages. French is Romance.) I have also included three more Germanic languages as the training language, Danish (DA), Dutch (NL) and Norwegian (NO). Everything else is the same. The results are shown in Table 3.3.

As the results shows, the system gained most improvements when Danish and Norwegian were added. Desipte Dutch and Swedish are both Germanic languages, it doesn't help a lot for the MNMT system to learn how to translate from Swedish or into Swedish. This confirms that closer languages would benefit each other even more by using pre-trained word embeddings (Qi et al., 2018).

To study how much of such kind of benefit were brought by shared vocabluaries, I modified the Danish expriment and tagged each word with its source language id in additional to every sentence. Punctuations are not distinguished among languages, which means they don't receive a language-specific tag. The word embeddings also tagged to point it to the correct source word. An English sentence that needs to be translated into German is then

```
__de__ <<en>>And <<en>>we <<en>>struggle <<en>>with <<en>>them .
```

The result in Table 3.4 showed that the BLEU score would dramatically decrease if each word is no longer allowed to be shared between languages. Hence most of the improvements were brought by the fact that Swedish, Danish and Norwegian have a large amount of common vocabluaries. On the other side, it also indicates that the system didn't learn too much syntactic information during training. Even though these languages have similar grammar structures, the system didn't catch it very well, otherwise we would see smaller BLEU score gap between the results as the close grammar relationship will be preserved in the embedding layer.

4 Target Filtering

4.1 Methodalogy

From the observation in last section, we found that the majority error in the translated text is that they are translated into the wrong language. Hence in this section we would like to see if it is possible to counter-attack this negative effect by replacing the words in the correct language manually.

The whole experiment is based on the hypothesis that our NMT system have already learned the genrally mapping between words in the source vector space and the ones in the target vector space, even though the correct word in the target word space hasn't been seen by the system during training. However since every aligned word embeddings are grouped by their semantics, the correct target word should also be around the wrong output word. In other words, our system is roughly on the right track, it just need to be fine-tuned to find the correct answer.

More specifically, in a vector space S that contains both the source and target aligned word embeddings W_s and W_t, for each $w_s \in W_s$ the system would have already learn at least one mapping to a target word $w_t \in W_t$. We are looking for other $w'_t \in W_t$ that are with in a specific radius of the original w_t. The distance should still be relatively small that that w_t and w'_t are both considered to be a effective translation of the source word w_s.

In theory to determine the nearyby neighbour w'_t we can use different kinds of metrics. Here I have chosen to use the Euclidean distance where determines the distance between w_t and w'_t as

$$d(w_t, w'_t) = \sqrt{\sum_{i=1}^{n} (w_{t_i} - w'_{t_i})^2} \tag{4.1}$$

4.2 Experiment Settings

To find the output word in the correct language, we would apply cosine similarity to compare the distance between two vector words. The distance d is a variable here and its value needs to be determined as well. Hence I have chosen to test the distance argument d by different experiments, ranging from $d = 0.25$ to $d = 5$.

The algorithm is described in Algorithm 1

```
Input: hypothesis $H$, source language embeddings $E_s$, target language
       embeddings $E_t$, distance threshold $D$
Result: Updated hypothesis $H'$ with words being replaced by their neighbors in
        the desired language
Build kd-tree $T$ from $E_s$ for $l \in H$ do each line $l$ in the source hypothesis $H$
    for $w \in l$ do each word $w$ in line $l$
        if $w$ is a punctuation then
            skip $w$;
        else if $w$ is an unknown word then
            skip $w$;
        else
            query distance $d(w, w')$ for $w$ in $T$;
            if $d < D$ then
                replace $w$ with the correspoding $w'$
            end
        end
    end
end
```

Algorithm 1: Pesudo code for output hypothesis word subsitution. Each word in the NMT output hypothesis that are not in the desired language will be replaced by its cloeset neighbour in that language.

Performing a distance query on a vector space that has more than 3×10^6 vectors is slow, especially when all these vectors are considered to be high dimensional vectors. I have implemented my code by using SciPy (Virtanen et al., 2019). There are algothrims like KD-tree (Maneewongvatana and Mount, n.d.) that could reduce the calcualtion time for low-dimensional vectors, but for vectors that are higher than 20 dimentions it is not necessarily faster than brutal force.[1] On the other hand, based on the Johnson–Lindenstrauss theorem (W. B. Johnson and Lindenstrauss, 1984), a vector space should have at least more than 300 dimensions to distinguish 1×10^6 vectors in it. As the aligned vector space in fastText contains more than 3×10^6 words, the dimensions could not be compressed any more or you are at risk of not be able to distinguish each word. All in all, the script is slow at subsitute every word in the output hypothesis into the correspoding one in the desired language.

4.3 Results and Analysis

[1] As descibed on the API document, "High-dimensional nearest-neighbor queries are a substantial open problem in computer science.", https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html

5 Conclusion and Future Work

Bibliography

Arivazhagan, Naveen, Ankur Bapna, Orhan Firat, Roee Aharoni, Melvin Johnson, and Wolfgang Macherey (2019). “The Missing Ingredient in Zero-Shot Neural Machine Translation” (Mar. 2019). eprint: 1903.07091. URL: https://arxiv.org/pdf/1903.07091.pdf.

Artetxe, Mikel, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho (2017). “Unsupervised Neural Machine Translation” (Oct. 2017). eprint: 1710.11041. URL: https://arxiv.org/pdf/1710.11041.pdf.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural Machine Translation by Jointly Learning to Align and Translate” (Sept. 2014). eprint: 1409.0473. URL: https://arxiv.org/pdf/1409.0473.pdf.

Baziotis, Christos, Barry Haddow, and Alexandra Birch (2020). “Language Model Prior for Low-Resource Neural Machine Translation” (Apr. 2020). eprint: 2004.14928. URL: https://arxiv.org/pdf/2004.14928.pdf.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin (2003). “A Neural Probabilistic Language Model”. *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435.

Blackwood, Graeme, Miguel Ballesteros, and Todd Ward (2018). “Multilingual Neural Machine Translation with Task-Specific Attention” (June 2018). eprint: 1806.03280. URL: https://arxiv.org/pdf/1806.03280.pdf.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2016). “Enriching Word Vectors with Subword Information” (July 2016). eprint: 1607.04606. URL: https://arxiv.org/pdf/1607.04606.pdf.

Conneau, Alexis, Guillaume Lample, Marc’Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou (2017). “Word Translation Without Parallel Data” (Oct. 2017). eprint: 1710.04087. URL: https://arxiv.org/pdf/1710.04087.pdf.

Denkowski, Michael and Graham Neubig (2017). “Stronger Baselines for Trustable Results in Neural Machine Translation” (June 2017). eprint: 1706.09733. URL: https://arxiv.org/pdf/1706.09733.pdf.

Di Gangi, Mattia and Marcello Federico (2017). “Monolingual Embeddings for Low Resourced Neural Machine Translation”. In: Dec. 2017.

Firat, Orhan, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho (2016). “Zero-Resource Translation with Multi-Lingual Neural Machine Translation” (June 2016). eprint: 1606.04164. URL: https://arxiv.org/pdf/1606.04164.pdf.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Ha, Thanh-Le, Jan Niehues, and Alexander Waibel (2016). “Toward Multilingual Neural Machine Translation with Universal Encoder and Decoder” (Nov. 2016). eprint: 1611.04798. URL: https://arxiv.org/pdf/1611.04798.pdf.

Ha, Thanh-Le, Jan Niehues, and Alexander Waibel (2017). “Effective Strategies in Zero-Shot Neural Machine Translation” (Nov. 2017). eprint: 1711.07893. URL: https://arxiv.org/pdf/1711.07893.pdf.

Hermann, Karl Moritz and Phil Blunsom (2013). "Multilingual Distributed Representations without Word Alignment" (Dec. 2013). eprint: 1312.6173. URL: https://arxiv.org/pdf/1312.6173.pdf.

Johnson, Melvin, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2016). "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation" (Nov. 2016). eprint: 1611.04558. URL: https://arxiv.org/pdf/1611.04558.pdf.

Johnson, William B and Joram Lindenstrauss (1984). "Extensions of Lipschitz mappings into a Hilbert space". *Contemporary mathematics* 26.189-206, p. 1.

Joulin, Armand, Piotr Bojanowski, Tomas Mikolov, Herve Jegou, and Edouard Grave (2018). "Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion" (Apr. 2018). eprint: 1804.07745. URL: https://arxiv.org/pdf/1804.07745.pdf.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization" (Dec. 2014). eprint: 1412.6980. URL: https://arxiv.org/pdf/1412.6980.pdf.

Lakew, Surafel M., Alina Karakanta, Marcello Federico, Matteo Negri, and Marco Turchi (2019). "Adapting Multilingual Neural Machine Translation to Unseen Languages" (Oct. 2019). eprint: 1910.13998. URL: https://arxiv.org/pdf/1910.13998.pdf.

Lakew, Surafel M., Quintino F. Lotito, Matteo Negri, Marco Turchi, and Marcello Federico (2018). "Improving Zero-Shot Translation of Low-Resource Languages" (Nov. 2018). eprint: 1811.01389. URL: https://arxiv.org/pdf/1811.01389.pdf.

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015). "Improving Distributional Similarity with Lessons Learned from Word Embeddings". *Transactions of the Association for Computational Linguistics* 3, pp. 211–225. DOI: 10.1162/tacl_a_00134. URL: https://www.aclweb.org/anthology/Q15-1016.pdf.

Maneewongvatana, Songrit and David M. Mount (n.d.). "Analysis of approximate nearest neighbor searching with clustered point sets" (). eprint: cs/9901013. URL: https://arxiv.org/pdf/cs/9901013.pdf.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space" (Jan. 2013). eprint: 1301.3781. URL: https://arxiv.org/pdf/1301.3781.pdf.

Mikolov, Tomas, Quoc V. Le, and Ilya Sutskever (2013). "Exploiting Similarities among Languages for Machine Translation" (Sept. 2013). eprint: 1309.4168. URL: https://arxiv.org/pdf/1309.4168.pdf.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality" (Oct. 2013). eprint: 1310.4546. URL: https://arxiv.org/pdf/1310.4546.pdf.

Neishi, Masato, Jin Sakuma, Satoshi Tohda, Shonosuke Ishiwatari, Naoki Yoshinaga, and Masashi Toyoda (2017). "A Bag of Useful Tricks for Practical Neural Machine Translation: Embedding Layer Initialization and Large Batch Size". In: *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 99–109. URL: https://www.aclweb.org/anthology/W17-5708.pdf.

Neubig, Graham, Matthias Sperber, Xinyi Wang, Matthieu Felix, Austin Matthews, Sarguna Padmanabhan, Ye Qi, Devendra Singh Sachan, Philip Arthur, Pierre Godard, John Hewitt, Rachid Riad, and Liming Wang (2018). "XNMT: The eXtensible Neural Machine Translation Toolkit" (Mar. 2018). eprint: 1803.00188. URL: https://arxiv.org/pdf/1803.00188.pdf.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://www.aclweb.org/anthology/P02-1040.pdf.

Qi, Ye, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig (2018). "When and Why are Pre-trained Word Embeddings Useful for Neural Machine Translation?" (Apr. 2018). eprint: 1804.06323. URL: https://arxiv.org/pdf/1804.06323.pdf.

Ruder, Sebastian, Ivan Vulić, and Anders Søgaard (2019). "A Survey Of Cross-lingual Word Embedding Models". *JAIR* 65, pp. 569–631. DOI: 10.1613/jair.1.11640. eprint: 1706.04902. URL: https://arxiv.org/pdf/1706.04902.pdf.

Smith, Samuel L., David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla (2017). "Offline bilingual word vectors, orthogonal transformations and the inverted softmax" (Feb. 2017). eprint: 1702.03859. URL: https://arxiv.org/pdf/1702.03859.pdf.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention Is All You Need" (June 2017). eprint: 1706.03762. URL: https://arxiv.org/pdf/1706.03762.pdf.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, and Denis Laxalde (2019). "SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python" (July 2019). DOI: 10.1038/s41592-019-0686-2. eprint: 1907.10121. URL: https://arxiv.org/pdf/1907.10121.pdf.

Vulić, Ivan and Marie-Francine Moens (2013). "A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else)". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1613–1624. URL: https://www.aclweb.org/anthology/D13-1168.pdf.

Zoph, Barret, Deniz Yuret, Jonathan May, and Kevin Knight (2016). "Transfer Learning for Low-Resource Neural Machine Translation" (Apr. 2016). eprint: 1604.02201. URL: https://arxiv.org/pdf/1604.02201.pdf.

Zou, Will Y., Richard Socher, Daniel Cer, and Christopher D. Manning (2013). "Bilingual Word Embeddings for Phrase-Based Machine Translation". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1393–1398. URL: https://www.aclweb.org/anthology/D13-1141.pdf.