Augmenting Data to Improve Incremental Japanese-English Sentence and Sentence-final Verb Translation

Ben Goldman

May 22, 2021
Version: Final Draft
Advisor: Alvin Grissom II
Abstract

Final verb prediction has been shown to help expedite simultaneous machine translation when translating between languages with different word orders. Prediction allows a system to begin translating earlier because it has access to information that it otherwise would need to wait for to begin translating. Specifically, if the system is able to predict the final verb in a SOV sentence, it allows the system to start translating much earlier. This thesis examines current prediction mechanisms in neural machine translation models to determine what factors improve predictions between SOV and SVO language. We first train a neural machine translation model to establish how well it can predict the English verb corresponding to the final Japanese verb. We then train new models, with modified data to see its impact on the model's ability to predict the English verb. We found that the model predicts the English verbs more accurately as more of the Japanese Sentence is revealed. Shuffling the preverb context as well adding subsentences to training data both improved the ability of the model to predict the English verb as well.
Acknowledgements

I would like to thank my advisor Alvin Grissom II for all his patience when helping me, even when I saved work to the last minute. I would also like to thank my second reader JD for his helpful comments.
1

Introduction

1.1 Introduction

Simultaneous machine translation is when a system translates in real time. The sentence is translated immediately as it’s inputted, similar to how a person would translate. Translating incrementally creates issues because the translator no longer has perfect information. In the situation where the translator can wait to see the full sentence, it has perfect information of the context it’s translating and thus can make as accurate of a translation as possible. However, waiting for the full context inevitability causes delay in the translation. To expedite simultaneous translation, the translator must predict future parts of the sentence. For example, consider when a system translates from Japanese to English. English sentences have the structure of subject-verb-object (SVO), whereas in Japanese the structure is subject object verb (SOV). Thus, in order to translate the sentence, the translator must wait until it sees the verb at the end of the sentence. This issue has been addressed in previous work by allowing the system to predict what words it expects to see later in the sentence.

This thesis aims to first give an overview of the current methods and technologies for simultaneous machine translation and then to evaluate current MT models if explicit final verb prediction is necessary and what factors affect prediction.
Background

2.1 Background

2.1.1 Machine Translation

Machine translation (MT) is the use of algorithms to translate text or speech from a source language to a target language. The idea of using computers for translation was initially proposed in 1946 by A. D. Booth and Warren Weaver at Rockefeller Foundation (Weaver 1955), and the first attempt of MT was done by Dostert (1955) a few years later. This thesis will be concerned with two types of MT; statistical machines translation (SMT) and neural machine translation (NMT). Statistical machine translation models create a translation for a given source sentence by using a probability distribution. The translation is calculated based by the equation $p(e|f)$ where $e$ is the word in the target language and $f$ is the sentence or word in the source language. In simpler terms, the formula is calculating the chances the string in the target language $e$ is the translation given we have seen some string $f$ in the source language. Neural machine translation leverages neural networks to encode the source sentence as a vector and then decode that vector into the target language. NMT will be explained in more detail later in this thesis.

2.1.2 Background On Machine Learning and Neural Nets

Machine learning aims to train models by learning from inputted data (training data) to make predictions about data it has never seen before. Machine learning can be divided into supervised and unsupervised learning. In supervised learning,
the training data also contains the labels we are trying to predict. This allows the system to adjust its predictions based on whether or not it predicted the same label that was inputted. In unsupervised learning, the training data does not contain the label we are trying to predict; instead the system has to learn by observing patterns in the data.

Neural networks are a class of models commonly used in machine learning. These architectures are typically composed of groups of neurons organized into layers. Each neuron is a mathematical operation that takes its input number $x_n$ in between 0 and 1, multiplies it by a weight and then passes it to the next layer. In order for the neurons in one layer to interact with the neurons in the subsequent layer, each neuron in the subsequent layer is fed a number from the output of the previous layer. The output of the previous layer is calculated by multiplying each neuron by a weight $w_i$, summing those values and then normalizing the value so that it is between 0 and 1. More succinctly, it is the normalized dot product between a vector of all the neuron values in the layer and a vector of weights. Thus the input for the first neuron in a subsequent layer is $f(\sum x_i w_i)$ where $f$ is the normalizing function. These layers propagate forward until an output layer is reached which determines the network’s prediction given the input values.

**Recurrent Neural Networks**

An issue with traditional feed-forward neural networks is that they do not have persistence. Given sequential data, traditional neural networks are unable to use previously seen information to make future predictions. The solution is recurrent neural nets (RNN). An RNN works similarly to a traditional feed-forward neural net, but an RNN has cycles within the network. This allows RNNs to be good at processing sequence data for predictions. As shown previously, the output of a neuron in a hidden layer would be $h^t = f(\sum x_i w_i)$ where $x_i$ is the value of the neuron and $w_i$ is the associated weight. However, in the case of a RNN the value of the neuron would be $f(h^{t-1}, \sum x_i w_i)$. Note that in RNN activation functions take in the dot product of the neuron values and the weights, but it also takes in the input from the previous neuron. This change allows information to persist within the network and thus allow the network to retain a type of memory of what it’s seen as shown in Figure 2.1.
2.1.3 Background on Reinforcement Learning

Reinforcement learning (Kaelbling et al. 1996) is a type of machine learning used to teach models to make decisions in complex environments. In the standard case there is an environment which is represented by a set of states $S$, a set of actions $A$ for the agent to navigate those states, a reward function $R$ which tells the agent whether or not the action was desirable, and a policy $\pi$ which is what the agent follows to make its decisions. The goal of reinforcement learning is to train an agent to learn a policy $\pi$ such that it maximizes $R$. At every step $s \in S$ the agent takes an action based on $\pi$ and, based on the outcomes of that action, is either penalized or rewarded. The policy $\pi$ is adjusted accordingly. This process continues until the agent arrives at a termination state.

2.1.4 Neural Machine Translation

Neural machine translation (NMT) is an approach to machine translation that typically uses a type of neural network called an encoder-decoder (Bahdanau et al. 2015). These models consist of an encoder and a decoder. The encoder and often decoder are RNNs used to encode text to a vector and decode vectors to text, respectively. In the encoder-decoder framework the encoder extracts a fixed-length vector representation from a variable-length input sentence, and the decoder uses this vector representation to create the sentence in the target language. There have been many improvements upon this model for instance the addition of an attention mechanism, but this paper will not go into the specifics of them.
3.0.1 Prediction In Statistical Machine Translation

Consider the German sentence “Ich bin mit dem Zug nach Ulm Gefahren”, in English this sentence translates to “I traveled by train to Ulm”. This sentence cannot be accurately translated from German to English without knowing the verb at the end of the sentence is Gefahren, which means "to travel". However, waiting until the final verb is known will cause a delay in the translation. To generalize, waiting for the final verb in translations between subject object verb (SOV) sentences to subject verb object (SVO) sentences delays translations. To address this issue Grissom II et al. (2014) suggest a novel reinforcement learning framework that leverages predictions of the next word and predictions of the final verb. They also propose a system that predicts how the sentence will end and a metric to measure the accuracy of the translation against the speed of the translation (Grissom II et al. 2014). In other words the authors attempt to improve upon simultaneous SMT by training an algorithm how and when to predict future words and also present a method to quantify the improvements. Since the goal of SMT is to produce accurate translations while minimizing delay, the model has to balance optimizing accuracy and expeditiousness. In their study the authors translate from German to English because in English the sentence structure is often subject verb object (SVO), whereas in German the sentence structure is often subject object verb (SOV). Since a lot of a sentence’s meaning comes from the verb, in order to translate a German sentence into English the system must until wait until it knows the verb at the end of the sentence.

Since the system is translating in real time, the system needs to know when to translate, when to wait, and when to predict. They leverage reinforcement learning to create a policy to teach the system how to choose between these actions. In the
framework for the reinforcement learning the state \( s_t \) represents the world given that we have seen \( t \) words of the source language sentence. At every \( s_t \) the system has knowledge of the set of observed words \( x_{1:t} \) from the source sentence, predictions of the next word \( n_{t+1} \) and prediction of the final verb \( v^t \) along with the probabilities of those predictions. The set of actions the agent can choose from are wait, next words, commit, and verb. \text{Wait} \) tells the agent to wait until it has more input, \text{next word} \) tells the agent to guess what the next word in the sentence would be and apply it to the translation, \text{commit} \) tells the agents to translate based off the current information, and \text{verb} \) tells the agent to predict the final verb of the sentence. For example, given the partial observation “ichbin mit dem”, the state might contain a prediction at time \( t \) that the next word, \( n_{t+1} \), will be “Zug” and that the final verb \( v^t \) will be “gefahren” example taken from (Grissom II et al. 2014).

**Evaluating a simultaneous translation**

The agent could easily maximize either accuracy or speed in isolation. In order to maximize accuracy the agent could wait until the entire sentence has been inputted, which is called a batch policy (Pytlik and Yarowsky 2006). To maximize speed, the agent could translate after each word is inputted, which is called a monotone policy (Tillmann et al. 1997). The authors sought to train a policy such that their system could expedite translations when compared to a batch system but also maintain accuracy, by using predictions.

The authors evaluated their translations over time on the \( x \)-axis and BLEU score, an automated metric for assessing translation quality, (Papineni et al. 2002) on the \( y \)-axis. Every time the agent chooses an action that creates a translation, a score is plotted at that time step, with the goal to maximize the area under the curve. The authors mention the idea of a psychic translator, which is able to create complete translation after one word which would maximize area under the curve. In comparison a batch system would be a tall and narrow sliver to the right because it’s extremely accurate but only after it’s seen the full sentence (Grissom II et al. 2014).

Formally the authors defined \( Q \) as the score function for a partial translation, \( x \) as the sequentially revealed words \( x_1, x_2, ..., x_T \) from 1 to \( T \) and \( y \) as the partial translation \( y_1, y_2, ..., y_t \) with respect to a reference \( r \) where \( T \) is the length of the inputted sentence. Every \( y_t \) has a BLEU-n score. They calculate the total score
latency BLEU (LBLEU) of the translation by summing the scores of all the individual segments and then multiplying the final score by $T$.

$$Q(x, y) = \frac{1}{T} \sum_t \text{BLEU}(y_t, r) + T \cdot \text{BLEU}(y_T, r)$$ (3.1)

**Prediction**

The prediction for the next word is created using a bigram language model: they measure the probability of a word by using the conditional probability given the preceding word. For the final verb prediction, they used the prior probability of a particular verb $v_t$, with the likelihood of the source context $x_{1:t}$ at a time $t$ given that verb $p(x_{1:t}|v)$. The prior was calculated with relative frequency estimation. Thus

$$v^{(t)} \equiv \arg \max_v \prod_{i=1}^t p(x_i|v, x_{i-n+1:i-1}).$$ (3.2)

where $x_{i-n+1:i-1}$ is the $n-1$-gram context. In order to save computation time the system only considered the 100 most frequent final verbs.

**Policy**

After creating the framework and the translations, the authors also train a policy that made the agent maximize the LBLEU reward. The authors use imitation learning (Abbeel and Ng 2004; Syed et al. 2008). In imitation learning the agent is given an optimal sequence of actions, and learns a generalized policy that maps states to actions. The agent was given the optimal policy $\pi^*$ and learned a generalized policy. This policy learning can be viewed as a cost-sensitive classification, which means not all errors have the same cost. The state given to the classifier was represented as a feature vector $\phi(x_{1:t}, n_{t+1}, v^t)$, the loss was the quality of the translation measure by LBLEU and the output is the action that should be taken in that state. In order to help the classifier generalize beyond the training examples they included features such as the length of the source sentence and the identity of the predicted verb and next word as well as their respective probabilities.
Results

They show that their learned policy was able to outperform the monotone policy towards the end of a sentence and outperform the batch policy throughout the sentence as seen in Figure 3.1. Furthermore, the authors noticed in some cases the translation quality suffers, because the translation model they use favors more general verbs. As a result, some of the sentences with incorrect translation were still comprehensible, but the predicted verb was a more general verb instead of the intended specific one. As seen in Figure 3.2 where the model predicted “gezeigt” (showed) instead of “vorgestellt” (presented).

Fig. 3.1: The final reward of policies on German data taken from Grissom II 2014

Fig. 3.2: The system predicted an incorrect verb but still created a comprehensible translation taken from Grissom II et al. (2014)
3.0.2 Verb Prediction In Humans vs Machines

Grissom II et al. (2016) study humans’ ability to perform verb prediction in order to create a benchmark to compare against computational incremental verb prediction. They measured humans’ ability to predict verbs by having them predict final verbs in Japanese, an SOV language, given a fragment of the sentence. They then examine a computational approach to incremental verb prediction and compare its performance.

Methods

For the human experiments, the authors first selected 200 sentences from the Kyoto Free Translation Task (KFT) corpus (Neubig 2011b) a collection of sentences with both the Japanese and English translation. The participants were then asked to choose a verb chunk when shown a fragment of the sentence. A verb chunk is more formally known as a *bunsetsu* in Japanese and can be thought of in English as a phrase that contains the verb and information about that verb, such as negation and tense. Rather than have the participants choose any random *bunsetsu* they were presented four options which they had to select from. The authors created two data sets from these 200 sentences, one where only the final *bunsetsu* was removed and one where the sentence was cut at a random, predetermined length.

Human vs. Machine Results

The authors found that on average when the participants were presented sentences that were only missing the final *bunsetsu* they had an average accuracy of 81.1%, and when participants had to guess the verb phrase for the random length sentences the average accuracy was 54%. Since previous psycholinguistics experiments have shown that Japanese speakers start syntactic processing by using case, the authors examined the correlation between the number of case markers and accuracy (Yamashita 2000). Case markers mark the role of words in sentences. In English, case is not explicit, but in a language like Japanese, it is an explicit part of the language. For example, there would be a word after the subject to signify that word is the subject of the sentence. To measure the number of case markers in relation to the
length of the whole sentence, the authors defined case density. The number of case makers divided by the number of *bunsetsu*.

Predictability increased when more of the sentence was shown. In addition, the authors found that case density was a significant factor in predictive accuracy on the random length set, which caused the authors to suspect that case may be important information for the computer if there aren’t enough context words. Additionally, the participants faced an issue where their verb choice was incorrect, but was still reasonable for the sentence, and thus in these situations all the participants guessed the wrong verb.

**Verb Classification with a Linear Classifier**

After computing a baseline to compare against, the authors introduced incremental verb classification using logistic regression, a supervised machine learning classifier. In their experiment, the authors used German and Japanese because they both have sentences with SOV structure. The motivation behind using logistic regression as opposed to the prediction scheme used in Grissom II et al. (2014) is that the latter has low accuracy and tends to over predict more common verbs. Its tendency to over predict common verbs stems from the way the verb prediction is calculated. As mentioned earlier, the verb is selected using the formula

$$v(t) \equiv \arg \max_v \prod_{c \in x_t} p(c|v)p(v).$$

(3.3)

This formula chooses the verb that maximizes the probability of the observed context, scaled by the prior probability of the verb in the overall corpus. However, since the distribution of verbs in the corpus is uneven the calculation favors more common verbs.

In order to train the model, it was given 463,716 verb final sentences, along with context features and final verb features. The context features are unigrams and bigrams for the words preceding the final verb, unigrams and bigrams for case markers in the order they appear and the last observed case marker. These features have binary values 1 if they are present and 0 otherwise. The verb features are the verb’s tokens given by the morphological analyzer, which typically include tense and aspect information. In addition, the model takes into account the interactions
between the verb features and the context features, so the model is given the Cartesian product of the context and verb features.

To predict verbs, the classifier determines the probability of all of the four options, by adding the verb feature to the feature vector. The highest probability of +1 or the lowest of -1 is chosen.

The authors found that their model, similar to humans, improves accuracy as more of the sentence is revealed (Figure 3.3), but also lagged behind the human participants.

![Fig. 3.3: Graph of verb classification results taken from Grissom II et al. (2016)](image)

**Verb Multi class Classification with a linear Classifier**

Finally, in order to compare their classifier with the work done by Grissom II et al. (2014) focused on how well they could classify the most frequent verbs instead of predicting all verbs with a multi-class logistic regression classifier. In this experiment, classifiers can only classify the verbs as one of the fifty most common verbs in the Japanese KFT and the German Wortschatz corpus, a collection of sentences with both English and German translations. In order to train this classifier, the features are once again case marker unigrams, case markers bigrams, and the last observed case marker.
Results

The logistic regression classifier did better than the $n$-gram classifier because the $n$-gram classifier has the tendency to over-predict frequent verbs. The classifier had an accuracy reaching 39.9% for German and 29.9% for Japanese. The results show that bigram features helped in both languages, but anything higher than trigrams cause the model to overfit the training data. Furthermore, they found that bigrams improved translation with Japanese more than with German, and they hypothesised this was due to the fact that the way some words interact in Japanese provide a lot of information about the sentence.

3.0.3 Prediction in Neural Machine Translation

Alinejad et al. (2018) study the performance of prediction in simultaneous NMT. This simultaneous NMT system uses an AGENT to control the encoder-decoder NMT model. Previously the AGENT only had the choice between the actions READ and WRITE. READ would add more information to the encoder RNN and WRITE would produce output using the decoder RNN. This proposal added a third action PREDICT, that would allow the AGENT to predict the next word in the sentence. This is inspired by Grissom II et al. (2014), but this prediction is not necessarily for the final verb and is being applied in a NMT model instead of a SMT model. Their hope was that by adding this extra action it would minimise delay and improve the quality of the translation (Alinejad et al. 2018).

This agent-based framework was first suggested by Satija and Pineau (2016), but theirs did not have the prediction feature. This framework is composed of two major components, the ENVIRONMENT and the AGENT.

The ENVIRONMENT the authors used for their system is an attention-based encoder-decoder MT system (Bahdanau et al. 2015) adapted for simultaneous translation. The encoder RNN converts the inputted words and predicted words into a context vector $H^p_n = h_1, h_2...h_n$ where $n$ is the number of inputted so far and $p$ is the number of predicted words since the last read. Whenever the AGENT chooses to READ $h_n = f_{ENC}(h_{n-1}, x_n)$ where $x_n$ is the next input word. If the AGENT instead chooses to PREDICT $h_{n+p} = f_{ENC}(h_{n+p-1}, x_p')$ is added to the context vector. At every time step $t$ the decoder uses the current context vectors to generate a predicted output $y_t$. If the AGENT decides to READ or PREDICT the current prediction $y_t$ will be
discarded, but if the action is WRITE $y_t$ will be produced as the output and the system will be updated accordingly.

The AGENT determines which action to take at every time step $a_t$. By the end of the program the AGENT has created a sequence of actions to tell the framework what to do with the ENVIRONMENT $A = a_1, a_2...a_T$. The AGENT was trained using reinforcement leaning with the policy gradient algorithm (Williams 1992). A policy gradient algorithm means that it uses gradient ascent or descent to alter the parameters to maximize the expected reward function for the model. In order to expedite training the authors prevented the AGENT from choosing redundant actions as shown in figure 3.3. For example if the AGENT has chosen WRITE at the previous time step it cannot predict next because it has no context.

![Fig. 3.4: Transition graph of actions taken from Alinejad et al. (2018)](image)

The reward function for the agent was defined as the cumulative sum of rewards for previous actions, and the reward for each individual action was defined as the combination of quality $r_Q^t$, delay $r_D^t$, and prediction $r_p^t$ rewards. Thus the formula for the final reward at each step was defined as $r_t = \alpha r_{t}^Q + \beta t_{t}^D + \lambda r_{t}^p$ where $\alpha$, $\beta$, and $\lambda$ are adjustable parameters to maximize focus on quality, delay or prediction respectfully.

To measure the quality the authors used a modified version of the BLEU score multiplied by the Brevity Penalty to measure how each action impacted the quality of the translation. The Brevity Penalty prevents very short sentences from receiving too high of a score. $\Delta BLEU(t)$ is the difference between BLEU score of the translated sentence at the previous time step and the current time step $\Delta BLEU(t) = BLEU(W^t, W^*) - BLEU(W^{t-1}, W^*)$. Where $W^t$ is the prefix of the translated sentence at time $t$ and $W^*$ is the reference.

$$r_Q^t = \begin{cases} 
\Delta BLEU(t) & t < T \\
BLEU(W, W^*) & t = T 
\end{cases}$$
The delay reward is what encourages the AGENT to minimize delay. They used Average Proportion (AP) the average number of source words needed when translating each word. (Gu et al. 2017) Given the source words $X$ and the translated words $W$, AP is

$$d(X, W) = \frac{1}{|X||W|} \sum_t s(t)$$

(3.4)

$$r_t^D = \begin{cases} 
0 & t < T \\
 d(X, W) & t = T 
\end{cases}$$

where $s(t)$ is the number of source words.

The authors of this paper found that in order to encourage the AGENT to choose the PREDICT option the authors had to create prediction quality reward which rewarded the AGENT for accurate predictions. This is defined as

$$r_t^p = \begin{cases} 
\Delta BLEU(t) & a_t = WRITE, a_{t-1} = PREDICT \\
 r_{t-1}^p & a_t = WRITE, a_{t-1} \neq PREDICT \\
0 & \text{otherwise}
\end{cases}$$

**Results**

They train their model on English-German in both directions and compare their model against the previously establish simultaneous neural MT model. The authors found that as the sentence length increased, prediction helped translation quality, but in the case of shorter sentences the prediction actions do not improve translation quality. From their results they also find the highest quality scores when the prediction was about 20 percent of the actions for both EN $\rightarrow$ DE and DE $\rightarrow$ EN. If there is no reward for prediction the number of predictions quickly goes to 0, but if the prediction reward is too highly the model predicts too much and the quality of the translation decreases. Overall Alinejad et al. (2018) show the power of prediction in NMT systems but do not focus on cases where the languages have different sentence structure.
4

Statement of The Problem

4.0.1 Motivation and Problem Statement

Previous work has shown that it’s possible to predict the final verb with linear and neural models when simultaneously translating from SOV languages to SVO languages to expedite translations (Grissom II et al. 2016; Li et al. 2020). Furthermore, it has been claimed neural simultaneous machine translation models also predict the final verb (Gu et al. 2017). We want to investigate that claim by evaluating how well a standard neural machine translation model can translate the first English verb (corresponding to the final Japanese verb) given a partial Japanese SOV sentence.

Grissom II et al. (2014) used final verb prediction in statistical machine translation, and Alinejad et al. (2018) implemented verb prediction in neural machine translation. We want to first train a neural machine translation model to evaluate whether it is predicting the final verb implicitly. After training this model we to compare it to three other models that have been all trained on the same sentences, but each with a variation to the training data to highlight what other factors affect a model’s verb prediction ability. The first variation we test is the removal of the final verb from the Japanese training data. If it is the case the model is still able to predict the verb, the model is able to learn what the verb should be from other context. Next, we shuffle the preverb context to help the model build stronger associations between the words and the final verb. Finally, we mimic the work done in Li et al. (2020) to see how our model does in comparison to theirs. From these experiments we hope to be able to determine if these types of models able to predict the English verb corresponding to the final Japanese verb, and if so, what are some of the conditions are that affect their ability to do so.
5

Methods

5.0.1 Verb Prediction with Neural Machine Translation

On standard machine translation tasks, machine translation models tend to improve with more examples. However, our task of predicting verbs is not a standard machine translation task. The following experiments are designed to clarify the role that the amount of context plays in the verb prediction task. To this end, we train our models on increasing larger sets of the sentences and observe the model's ability to predict the first English verb.

Following Li et al. (2020), who used BiGRUs with attention to predict verbs directly, we train our model with transformers. Both architectures use attention mechanisms but the transformers do not use any recurrent networks like the BiGRUs used by Li et al. (2020). Transformers are a relatively new architecture used for translations task, but Devlin et al. (2018) showed that the transformers were able to exceed state-of-the-art results on several natural language processing tasks.

5.0.2 Neural Machine Translation

We first train a neural machine translation model to translate Japanese sentences to English sentences. We use this model to establish a baseline to compare other models that have been trained on slightly altered data. After training, the model will be measured on it's ability to properly predict the English verb.
5.0.3 Dataset

We train the model with Japanese and English sentences from the Kyoto Free Translation Corpus (KFT). The KFT is composed of Japanese sentences and those sentences’ English translations. For every English sentence, the first English verb is extracted and stored to compare against the translated verb. We use the official KFT split where the KFT is split into train (440,000), dev (1,166) and test (1,160) sets. To extract the verb from the English sentences and determine if and when the verb appears in the translated English sentences, the English sentences are tagged using a part-of-speech (POS) tagger. A POS tagger, when given a sentence will output the same sentence, but with every word tagged as a noun, verb, adjective, etc. Below is an example of English sentence that has been tagged. In this example, NNP stands for proper noun singular and VBD stands for verb past tense. The POS tagger used in this thesis is Flair, a NLP python package.

George Washington went to Washington .

George Washington went to Washington.

| George | Washington | went | to | Washington |
| NNP | NNP | VBD | IN | NNP |

5.0.4 Model

The model used in this experiment is a transformer (Devlin et al. 2018). The transformer is a network architecture based solely on attention mechanisms. This model has been shown to be equal if not superior to previous models on large data and trains much quicker. We train the model on the 440,000 parallel Japanese and English sentences. The validation data was the 1,166 sentences from the corpus. The sentences are tokenized by character for both the Japanese and English sentences. We use a batch size of 256, and the stopping criteria is when there has been no improvement for the last 50 checkpoints. After training the model’s final BLEU score was .35, which is an improvement over the reference BLEU score of .1935 for Japanese to English on this dataset (Neubig 2011a). The reference BLEU score for this dataset is from the KFT’s website where they track statistical models that have the best BLEU scores on the dataset.¹

¹http://www.phontron.com/kftt/
The model was trained using Sockeye, an open-source sequence-to-sequence toolkit for NMT (Domhan et al. 2020). Sockeye implements the state-of-the-art transformer encoder-decoder architecture. During testing, each Japanese sentence is split into every possible subsentence as shown in the example below. These subsets are inputted to the trained model. If the first verb in the translated sentence is an auxiliary verb such as *is* we check if the next word in the sentence is a content verb. If so we consider the content verb the predicted verb of the sentence. Otherwise we consider the first verb the predicted verb. After the first verb is located the verb is stored to compare against the actual verb of the sentence. The phrase 曹洞宗 の 開祖 is broken down into the following subsentences:

曹洞宗の開祖

Fig. 5.1: Accuracy when predicting the English verb at different intervals in the sentence. As more of the Japanese Sentence is revealed the accuracy of verb prediction increases. When comparing lemmas the accuracy is significantly higher than when we are comparing just the content verbs.
5.0.5 Results

Prediction of All Verbs

We first analyze the performance of the model by measuring accuracy of all the predicted English verbs as a function of the percentage of the Japanese sentence that was revealed when the English verb was predicted. Accuracy is computed by measuring the amount of times the model matched the first verb or matched any of the first contiguous verbs. Fig 5.1 shows that as more of the sentence is revealed,
accuracy of verb prediction increases. However, the overall accuracy of all verbs is relatively high when at most only 10% is revealed. Further investigation shows that the dataset is dominated by common verbs. These verbs are mostly auxiliary verbs as opposed to content verbs. Some examples of auxiliary verbs are *is*, *were* and some examples of content verbs are *buy*, *want*, *swim*.

**Prediction of Main Verbs**

Since the dataset is dominated by only a few auxiliary verbs, as shown in fig 5.3, the accuracy when comparing the prediction of all verbs largely measures whether one of those common verbs is present in the sentence. To better evaluate the model’s ability to predict content verbs, we only consider examples where the verb is a content verb.

As expected, after excluding auxiliary verbs, the accuracy significantly decreases, but there is a more significant trend of improvement as the percentage of the Japanese sentence is revealed. This is mostly likely due to that fact that content verbs require more context than auxiliary verbs like *is* and *was*.

**Prediction of Verbs Without Considering Tense**

Analyzing the predictions where the model incorrectly predicts the verb shows that in many cases it was predicting the correct verb, but in the wrong tense. For example, it would predict the verb *had* but the actual verb is *have*. To determine whether the model predicted the correct verb regardless of whether model also predicted the correct tense, we compute the accuracy of the lemma of the verbs instead of the verb. A lemma is the base form of the verb. For example, from "produced", the lemma is "produce". By comparing lemmas instead of the inflected verb, the predicted verb and the actual verb will match even if the tense does not. The lemmas of the verbs were computed using NLTK’s lemmatizer.

Fig 5.1 shows that when comparing the lemma of the verb the accuracy improves in comparison to when only the inflected verbs are compared. By the definition of a lemma, the accuracy when comparing lemmas instead of verbs must be equal or greater than the accuracy when comparing the verbs, because if the verb matches then the lemmas must match as well. Again, there is a trend of improvement as
more of the sentence is revealed. In all three instances we see that the model is able to more accurately predict the verb as it gains more context.

**Impact of Length**

It has been previously shown that as sentence length increases, the accuracy of sentence final verb prediction decrease (Grissom II et al. 2016). We test whether this holds true for our model by measuring the accuracy of verb prediction as a function of the total length of the sentence. Fig 5.2 shows that as the length of the sentence increases the accuracy of our model decreases, as in Grissom II et al. (2016).

**5.0.6 Prediction Without Training with the Final Verb**

After establishing a baseline model, we consider what other factors are important in English verb prediction. Specifically, how important is it for the model to observe the final Japanese verb during training? To test this, we train our model with the same dataset used in the first experiment, but the final Japanese verb has been removed from the training data. We remove the final Japanese verb from the training data, by locating the verb using a POS tagger. We then observe the effect this has on the accuracy of verb prediction. Since we have established a baseline of how well the model performs when the data is not altered we can see how this removal will affect the scores in comparison. The results will further highlight what parts of the sentence impact the model's ability to predict the verb.

**Data Preparation and Model for Verb Removal**

In order to have an accurate comparison, the training, validation and test sets for this experiment are the same sentences used in the previous model. The only difference between these datasets and the previous datasets is that the Japanese sentences have been tagged using a POS tagger to identify the Japanese verb and all the final verbs have been removed from the training and validation data. The model parameters are identical to the model in the previous experiment. The data was tokenized by character, the model used transformers and was trained using Sockeye. The tagger used for this experiment was python package called Fugashi because the
Flair tokenizer cannot tokenize Japanese sentences (McCann 2020). The final BLEU score for this model after training is .34 which is slightly lower than the BLEU score for our baseline model. Therefore, it would appear that verbs are crucial for the model to accurately learn how to predict the verb.

**Fig. 5.4:** Accuracy of verb prediction when the model was trained without the verb as a function of the percentage of the Japanese sentence that was revealed. Accuracy was much lower in comparison to the other models.

**Removal of Verb Results**

To compare the model that was trained without the final Japanese verb to the baseline model we compare the accuracy of the model when evaluating all verbs, only content verbs, and the lemmas of the verbs.

**Accuracy on All Verbs**

When computing the accuracy of English verb prediction on all verbs the model preforms better than the model’s performance on the other sets. However, this is again due to the fact of the uneven distribution of verbs in the dataset. Of the 15,916 total sentences, the model predicts the verb to be was 7,724 times even though the actual number of instances of was is 4423. From this and the extremely low BLEU score, we can hypothesize the model is not correctly translating the sentence and over-predicting the verb was. Thus, the model’s high accuracy on all verbs is not an accurate metric of the model’s ability to predict content verbs.
5.0.7 Verb less Model on Main Verbs/Lemma

Evaluating the accuracy of the model when only looking at the content verbs and the lemmas of the verbs, we see it is slightly worse than when the model is trained with the verb. Both this model and the baseline model were able to predict the correct verb over 10% of the time when enough of the Japanese sentence was revealed.

5.0.8 Verb Prediction with Training on Preverb Context Shuffled

In Japanese, word order is not as strict as it is in languages like English, it allows scrambling. However, since we are scrambling the words rather than the bunsetsu chunks the English meaning is not preserved. We test if the model will learn additional relationships between the preverb context and the verb by presenting the preverb context in a different order with the same English translation. This experiment aims to measure how shuffling the preverb context will affect the verb prediction accuracy of the model. We have previously shown how a model without shuffling performs and how a model without the final verb performs. We now train a model with shuffling and compare it to those previously trained models to see to what extent the model needs well formed sentence to correctly translate the first English verb.

Data Preparation and Model for Shuffling

The training, validation, and test sets are the same sentences as the models trained without shuffling and without the verb for accurate comparisons, but the training and validation data have the pre verb words shuffled. We use the same POS tagger as used in the verb removal experiment to isolate the final verb. After isolating the final verb, we randomly shuffle all of the preceding context. We keep the model the same for fair comparison with the baseline model. After training, the final BLEU score of this model is .28, a slight improvement of .03 in comparison to the baseline model.
Fig. 5.5: Accuracy of verb prediction when the model was trained on shuffled pre verb context as a function of the percentage of the Japanese sentence that was revealed. Verb prediction accuracy increases as more of the sentence is revealed. The Final BLEU score of this model is .34

Preverb Context Shuffle Results Analysis

We measure the accuracy with all verbs, content verbs, and the lemmas of those verbs. Fig 5.5 shows a similar pattern to the baseline results. When analyzing the verb prediction accuracy of all verbs we see relatively high performance when compared to the accuracy of content verb prediction. Fig 5.5 shows that the accuracy of content verb prediction and lemmas slightly increases as the amount of context increases. When comparing the lemmas of the verbs the accuracy rises from 3% to 10%.

5.0.9 Verb Prediction with progressively longer subsentences

We now train the model on progressively longer subsentences, in line with the work done by Li et al. (2020). In their study, they train on progressively longer subsentences at intervals of 30%, 50%, 70% 90% and 100% of the sentence. They use a BiGRU model, but in our experiments we use transformers. Since we attempt to predict the verb at every subsentence we hope to see if the addition of these subsentences to the training data will improve the model’s accuracy. In addition the
additional subsentences significantly increases the size of the training data, as each sentence is split into five different subsentences.

### Adding Subsentences to Data

We use the same sentences from the KFT as used other experiments for consistency. We filter for the sentences that end in a verb. The only difference is that the training and validation data are truncated into five progressively longer subsentences: 30%, 50%, 70%, 90% and 100%. These subsentences are then appended to the training data. After training, the final BLEU score for the model is .34.

![Graph showing accuracy of verb prediction](image)

**Fig. 5.6:** Accuracy of verb prediction when the model was trained with additional subsentences measured as a function of the percentage of the Japanese sentence that was revealed. Overall the model has much higher accuracy than the original model.

#### 5.0.10 Additional Subsentences Results

Fig 5.6 shows a sharp increase in accuracy of verb prediction when predicting content verbs and comparing the lemmas as more of the percentage of the Japanese sentence is revealed, whereas the accuracy when comparing the prediction of all verbs remains the same over time. This indicates again that measuring the model's ability to predict all verbs is largely measuring if one of the common verbs occur in the sentence. When comparing the lemmas almost no context revealed the accuracy is around 6%, but as the full sentence is revealed accuracy improves to 13%. The
model significantly improves with more context and performs better at every point than the baseline model.

Fig. 5.7: Comparison of accuracy of verb prediction on all verbs as a function of the percentage of Japanese sentence revealed across the baseline model, the model that was trained on the shuffled preverb context and the model that was trained on additional subsentences. The model that was trained on the shuffled preverb context peaks when 30% of the sentence is revealed. The baseline model peaks initially and then continues to trend back towards that peak as more of the sentence is revealed. The model that was trained on additional subsentences had the highest accuracy, followed by the baseline and finally the model that was trained on the shuffled preverb context.

Fig. 5.8: Comparison of accuracy of lemmas as a function of the percentage of Japanese sentence revealed across the baseline model, the model that was trained on the shuffled preverb context and the model that was trained on additional subsentences. Accuracy increases as more context is revealed. The baseline model had the highest accuracy.
Fig. 5.9: Comparison of accuracy of verb prediction on content verbs as a function the percentage of Japanese sentence revealed across the baseline model, the model that was trained on the shuffled preverb context and the model that was trained on additional subsentences model. Both models show similar results to the baseline. Accuracy increases as more context is revealed. The model with additional subsentences performs better when there is less context, but with full context the baseline model performs best.

5.0.11 Comparison of Models

Fig 5.7 shows every model performs well when predicting all verbs in comparison to when the model is predict only content verbs and their lemmas. The accuracy on all verbs of these models may be due to the unbalanced nature of the dataset. In addition, the data suggests that BLEU score is positively correlated with accuracy. The models with higher BLEU scores had higher accuracy on verb prediction. This claim is supported by Fig 5.8 and Fig 5.9 because by the time the full Japanese sentence is revealed the model that had the highest BLEU score, performs the best. Furthermore, the shuffled model and the model with additional subsentences did not show significant improvement over the baseline which would suggest that shuffling the sentence did not impact the model’s performance. Interestingly, the shuffled model performed similarly to the baseline model despite the fact that in many cases shuffling the sentences made the sentences nonsensical.
Conclusion and Future Work

From previous literature it has been established that prediction is a powerful mechanism to expedite both simultaneous SMT and NMT. Currently most of work with prediction in NMT focused on only the prediction the next word or next few words (Alinejad et al. 2018). However it has been shown in SMT that prediction of the final verb can improve and expedite translations (Grissom II et al. 2014).

From our experiments we found that all the models performed better when they were given more context. Furthermore, we found that both shuffling the data and adding subsentences did not improve the model's performance. Similarly training the model without the final Japanese verb hindered the model's ability to learn.

Future work should analyze if the prediction mechanisms in NMT are implicitly predicting the final verb in SOV sentences. If these systems are predicting the final verb, further work can characterise and analyze the circumstances in which the system predicts the final verbs. With the intent to see if the model can be tweaked to see if it can focus on prediction on the final verb and if this would improve translations. If these systems are not implicitly predicting the final verb, future work should be done to implement final verb prediction into current systems.
References


