9-2020

Does the Word "Chien" Bark? Representation Learning in Neural Machine Translation Encoders

Emily Campbell
The Graduate Center, City University of New York

Does the Word "Chien" Bark?

Representation Learning in Neural Machine Translation Encoders

by

Emily Campbell

A dissertation submitted to the Graduate Faculty in Linguistics in partial satisfaction of the thesis requirement for the degree of Master of Arts, The City University of New York

2020

This manuscript has been read and accepted by the Graduate Faculty in Linguistics in partial satisfaction of the thesis requirement for the degree of Master of Arts.

Professor Kyle Gorman

Date	Thesis Advisor

Gita Martohardjono

Date	Executive Officer

The City University of New York

ABSTRACT

Does the Word "Chien" Bark?

Representation Learning in Neural Machine Translation Encoders

by

Emily Campbell

Adviser: Professor Kyle Gorman

This thesis presents experiments with using representation learning to explore how neural networks learn. Neural networks which take text as input create internal representations of the text during their training. Recent work has found that these representations can be used to perform other downstream linguistic tasks, such as part-of-speech (POS) tagging. This demonstrates that the neural networks are learning linguistic information and storing this information in the representations. We focus on the representations created by neural machine translation (NMT) models and whether they can be used in POS tagging. We train 5 NMT models including an auto-encoder. We extract the encoder from each model and utilize the representations that the encoder produces to train a hand-crafted Encoder-Tagger (ET) model to do POS tagging. We explore the impact of various features including NMT target language, NMT BLEU score, encoder depth, sequence length, token frequency, and percentage of out-of-vocabulary (OOV) tokens in a sequence. We find that NMT encoder representations contain sufficient linguistic information to perform POS tagging and that there are correlations between several features, which helps us to better understand the inner workings of neural networks.

Contents

List of Tables

List of Figures

1 Introduction

Representation learning is a set of techniques which extract representations from neural networks and use these representations in downstream tasks. In natural language processing (NLP), these representations consist of numerical vectors that the neural network creates in order to "represent" words, phrases, or sentences as they pass through the model. Importantly, each element word in a sentence is represented not simply as that word, but as that word in the context of that particular sentence. Representation learning aims to explore and evaluate the representations in order to better understand how neural networks work and in order to determine if the use of these representations can supplement other neural networks.

Representations are generated as the by-product of training in many NLP tasks, such as sequence-to-sequence generation, sentence classification, and sequence tagging. Taking neural machine translation (NMT) as an example, models generally contain an encoder and a decoder. A sentence is fed into the encoder starting with the embedding layer, which creates a representation of each of the tokens. The sentence then continues through additional layers of the encoder. They may be linear layers, which map to from one dimensionality to another, or recurrent neural networks, where the nodes are able to capture and map the temporal sequence. The output from the encoder is the intermediate representation of the input sentence. This representation is then fed into the decoder, which maps the representation into an output dimension the size of the target language vocabulary. A similar series of events occurs with part-of-speech (POS) tagging models. Sequences are passed through an encoder and then the resulting representation is passed into a tagger. However, in either of these models, if the encoder output is extracted before it goes into the decoder or the tagger, it contains the model's representation of the sentence.

The representations contain a significant amount of information about the encoded sequence and each token within it. A NMT decoder, for example, is able to take a representations and generate the input sequence in a completely different language. A POS tagger can

take a representation and tag each of the tokens in the sequence with a part-of-speech tag. Given the effectiveness of these representations in their intended tasks, many researchers have explored alternative ways that these representations can be utilized. Some have looked at what these representations can tell us about neural networks while others have used these representations to augment and supplement other neural network models. The goal of this paper is to analyze whether an encoder trained in a NMT model can be used for a POS tagging task. Furthermore, we explore what information may be contained in these representations with the hope of further understanding how NMT models create representations of the source language. This will be achieved by creating a NMT model, extracting the encoder, and utilizing the representations of the encoder in a downstream task, POS tagging.

1.1 Theoretical Background

We will now detail studies on whether encoders can be shared between tasks and on how to analyze the linguistic information contained within encoder representations.

1.1.1 Generalizability of Encoders

Firat et al. (2016) trained a single multilingual NMT model on multiple languages by having all of the models share a single attention mechanism. They found that for translation tasks with low-resource languages, there were large improvements in the translation quality when using a shared encoder compared to a NMT model which had been trained on only a single language. This demonstrates that information from one model may be able to supplement and improve the performance of the second model.

While Firat et al. (2016) shared an encoder between NMT tasks, it is also possible to share encoders between different tasks. Subramanian et al. (2018) looked at whether a single model could be used for a variety of different tasks with different training objectives and different data sets. The tasks included a sentence classification task, natural language in-

ference (NLI), and three sequence-to-sequence generation tasks: neural machine translation, constituency parsing, and skip-thought vectors. Skip-thought vectors is a task where the model predicts the preceding and subsequent sentence from the current sentence. The goal is to build strong generalizable sentence representations. The model takes three sentences as input, encodes the middle sentence, and then uses one decoder to try to predict the previous sentence and the other to predict the following sentence. The decoders back-propagate to the shared encoder, so the encoder learns simultaneously from both. The researchers found that sharing a single encoder across a variety of tasks, even those that were only weakly related, led to consistent and, with some tasks, substantial improvements. These findings show that encoders may be able to transfer information between tasks and that additional training may strengthen, rather than weaken, the encoder representations.

Kim et al. (2019) also explored how different pre-training objectives affect the information contained in the sentence encoders. They focused specifically on acceptability judgement and NLI for sentences containing function words, definite articles, and coordinating conjunctions. They found that the model which had pre-trained as a language model performed the best. However, while each of the pre-trained models performed well on a few tasks, none of them were able to outperform a model which had no pre-training. In contrast to earlier findings, these results show that pre-training on a different task may not improve the performance of a model, which suggests that further exploration is warranted.

1.1.2 Linguistic Information Contained in Representations

Shi et al. (2016) investigated whether NMT models learn syntactic information by using NMT representations in syntax-focused downstream tasks. At the word level, they experimented on POS tagging and identifying phrase constituents. At the sentence level, they tagged sentences for voice, tense, and the highest-level syntactic structure in the sentence. They concluded that NMT encoders do learn syntactic information. In addition, they found

that different layers of the model tend to store different syntactic information. Word level information, for example, is stored higher up in the encoder near the embedding layer, while sentence level information stored is further down closer to the decoder.

Building off the idea that different information can be stored in different layers, Belinkov et al. (2017) explored which layers of the encoder and decoder in NMT models contain the most helpful representations. They utilized a variety of tasks, including POS tagging and morphological tagging. Based on their results, they posited that what they consider to be higher levels of the model — such as the embedding layer — focus more on word meaning and semantics whereas the lower levels include more information about word structure, consistent with the findings of Shi et al. (2016). They also explored which language pairs were more helpful in downstream tasks. They found that translating into a morphologically impoverished language, like English, often results in better encoder representations.

Similarly, Poliak et al. (2018) also used the encoder output from NMT models to train classifiers. However, they used the encoder output for natural language inference tasks such as anaphora resolution, mapping a paraphrase back to its source, and semantic role identification. They concluded that encoders were not able to capture enough information to consistently perform anaphora resolution or paraphrase mapping successfully. However, out of 16 semantic roles that the model was tasked with identifying, it was able to perform above-baseline on 6 of them. They noted that the model performs significantly better with English-Spanish when compared to English-Arabic, English-Chinese, and English-German, suggesting that the target language used in the NMT training played an important role.

In order to further test how much syntactic information information encoders learn, Gulordava et al. (2018) trained generic language models in four languages and tested whether these models could predict long-distance number agreement in a variety of sentence constructions. They additionally tested whether the models could predict the correct verb agreement even for subjects and verbs that usually did not co-occur, such as the subject *dreams* and

the verb *sleep*. They compared the predictions of the best model, determined by the highest accuracy in predicting whether the verb should match a singular or plural subject, to those of human annotators and found a difference in accuracy of only 2.4% on correctly predicting whether the verb should match a singular or plural subject.

It is our goal to further explore the issue of whether NMT encoders can be used in structured classification tasks such as POS tagging. Additionally, we will analyze which layers of the encoder produce the best representations for performing POS tagging, and the effect of the NMT target language on these representations.

1.2 The Present Work

Our primary goal is to explore whether NMT encoders learn enough information to be able to be trained to perform POS tagging. Previous research has suggested that encoders learn sufficient morphological and syntactic information. We utilize a custom-built model, which is able to extract the encoder from a NMT model and use it for downstream POS tagging. Our experiments focus as well on how using different target languages impacts the ability of the encoder to perform POS tagging. We draw from a number of diverse languages with different writing systems. In addition, we experiment with which layers of the encoder perform best at POS tagging. We extract representations from all three layers of the encoder and determine the utility of each at performing the tagging task.

The remainder of the paper is as follows. Chapter 2 details the data sets used in the experiments and the preprocessing techniques used in cleaning and preparing the data as well as the NMT models and the Encoder-Tagger (ET) models, and their modules and hyperparameters. Chapter 3 outlines the architecture and training of the models as well as the experiments. Chapter 4 analyzes the results of the experiments. Chapter 5 concludes the research and discusses avenues for future work.

2 Materials and Methods

This chapter presents information about the data sets used in the NMT and tagging experiments, as well as the methods used to clean, process, and tag the data. It also describes the final product of the thesis, the set-up of the NMT and Encoder-Tagger (ET) models, and the hyperparameters and other aspects of the model that can be tuned by the end-user.

2.1 Data

2.1.1 Translation Data

The NMT models were trained on the United Nations Parallel Corpus (Ziemski et al., 2016), henceforth the UN corpus. The corpus contains manually translated official records and other parliamentary documents produced between 1990 and 2014. We used a subcorpus, which is fully aligned at the sentence level, so that the models trained on each language pair would have the same amount of training data. Each language pair initially consisted of 11.3 million sentences and 334 million English tokens. During cleaning, sentences were limited to those between 5 and 55 tokens in both the source and target language. As a result, there were slight differences between the number of sentences for each language pair. Full statistics are given in Table 1. Five NMT models were trained: English-to-English, English-to-French, English-to-Mandarin, English-to-Russian, and English-to-Spanish. The English-to-English model is an auto-encoder, which translates each source sentence into the exact same sentence on the target side. This model serves as a control to see what information, if any, is learned when no transformations are required between the input and the output.

2.1.2 Tagging Data

We use the Wall Street Journal portion of the OntoNotes corpus. This consists of English news text re-annotated with new POS tags. It contains 71.29% of the original Penn Treebank data set from which it is derived (Hovy et al., 2006). There are 34,372 sentences and

Target Language	# of Sentence Pairs
English	9,457,741
Spanish	8,799,611
French	8,459,142
Russian	9,208,596
Mandarin	9,234,179

Table 1: Number of sentences per language pair.

901,643 tokens in the corpus. This data set provides gold tags for the POS tagging task.

Neural networks often benefit from large training sets. However, most POS gold-tagged data sets are small due to the difficulty of hand-annotating data. To build a large POS data set, we instead POS tagged an existing data set using a state-of-the-art POS tagger, Flair. Flair is a neural network POS tagger which has achieved 97.64% accuracy on the Wall Street Journal POS tagging task, the best published result on this data set (Akbik et al., 2018). Although these tags cannot be considered gold-standard tags, having a larger data set may provide more opportunities for the ET model to learn and build better representations.

For the larger corpus, we chose the English section of the Workshop on Machine Translation (WMT) 2007 News Crawl data set, which is approximately 100 times larger than OntoNotes data set. It consists of text scraped from online news websites and split into individual sentences. Each sentence was cleaned and tokenized using language-specific tokenizers: spaCy tokenizers were used for English, French, Russian, and Spanish (Honnibal and Montani, 2017); the Jieba tokenizer was used for Mandarin Chinese (Sun, 2012). The training set comprised 70% of the data set while the validation and testing sets were made up of 15% of the data set each. The data was tokenized and cleaned using the same methods as the NMT data: unnecessary punctuation, such as "¿" in English texts, was removed, and sequences were limited to those longer than 5 tokens and shorter than 55 tokens. After cleaning, the data set contained 3,387,999 sentences and 69,576,868 tokens.

2.2 Models

The final product of this thesis is an end-to-end program that allows researchers to replicate this research project or to utilize the ET model to run their own experiments. The program cleans and tokenizes the data, limits the sequence length of the examples, builds vocabularies, trains NMT models, evaluates NMT models using cross-entropy loss and BLEU scores, decodes the output using either beam search or greedy decoding, extracts encoders from NMT models, loads the encoder into ET models, runs tagging data through ET models, and evaluates the accuracy of the ET models using categorical accuracy.

All of these programs can be run from the command line with a variety of options, including, but not limited to: the size of the vocabulary, the dimensions of the neural networks, whether to add pre-trained word embeddings, whether to freeze the embedding layer or the encoder, etc. This program is readily available on GitHub[1] for use by the general public with instructions and examples on how to use it. The code is licensed under the 3-Clause BSD license.[2]

2.2.1 Machine Translation

The NMT model is a sequence-to-sequence model that was built using the PyTorch library. It consists of an encoder, an attention mechanism, and a decoder. The NMT model is trained to minimize cross-entropy loss using Adam (Kingma and Ba, 2017).

Encoder

The encoder begins with an embedding layer which maps from the input vocabulary to the embedding dimension. The embedding layer has an option for weights from pre-trained word embeddings to be copied to it and this layer can be optionally frozen. Dropout can then be applied to the embedding layer. Next, the output from the embedding layer is passed

[1] `https://github.com/ebelle/encoder-tagger`
[2] `https://opensource.org/licenses/BSD-3-Clause`

through a Long Short Term Memory (LSTM) layer, and dropout is again applied. The LSTM can have 1 or 2 layers, and there is an option for it to be bidirectional.

In order to prevent the padding indices from the padded batches from passing through the LSTM, we use the PyTorch `pack_padded_sequence` function. The packed batch is fed through the LSTM, which maps from the embedding layer to the hidden layer. After the LSTM, we pad the encoder output back to its original size using the `pad_packed_sequence` function. Finally, if the encoder is bidirectional, the forward and backwards passes of the LSTM are concatenated, so that the encoder output is the same size as the hidden state from the LSTM. The encoder then returns the encoder output along with the hidden state from the LSTM.

Attention Mechanism

Attention weights are obtained using the attention model proposed by Luong et al. (2015). A mask is applied, so that any padding tokens are assigned zero weight, so that the model does not assign an attention score to the padding tokens. Finally, the weights are passed through a softmax function.

Decoder

The decoder consists of an embedding layer which maps the target vocabulary to the embedding dimension. Dropout is then applied. The embedded token is passed through a LSTM along with the hidden state from the previous time step. If the input is the first token in the sequence, then the hidden state from the encoder is used. For subsequent steps, the hidden state returned from the decoder LSTM is used. The attention weights are calculated by passing the hidden state from the LSTM, the encoder output, and a mask for the padding to the attention mechanism. Batch matrix-matrix multiplication is performed on the attention weights and encoder output in order to get a context vector. If the LSTM is bidirectional, we concatenate the encoder output for the forward and backward pass. Finally, the LSTM

output and the context vector are concatenated and passed through a fully connected layer, which maps them to a dimension the size of the output vocabulary. The output from this is then passed through a softmax layer to obtain the prediction at each time step.

Decoding

To obtain the predicted sequence, we use a greedy decoder. At each time step, the token with the highest predicted value is chosen as input for the next time step.

2.2.2 Encoder-Tagger

The ET model was also built using PyTorch. It consists of an encoder, which has been extracted from the saved NMT model, and a tagger. The tagger takes the representations from the encoder and predicts POS tags for each token in the sequence. It consists of one or two hidden layers, and a fully connected output layer, which maps the output from the hidden layer to the predicted tags. The ET model was once again trained to minimize cross-entropy using Adam.

Encoder

The encoder module is taken directly from the NMT model. The hyperparameters from the NMT model are saved and then loaded into the ET model. To build the encoder architecture, the user chooses which layer of the encoder they would like to extract the representations from. Optionally one can choose embedding representations, in which case only the embedding layer is built. For first LSTM layer representations, the embedding layer and the first LSTM layer are built, and to obtain representations from the whole encoder, the embedding layer and the first and second LSTM layers are built. The state dictionary from the pre-trained NMT encoder, which contains all the layers of the encoder and their learned parameters, is then loaded into the ET encoder. Once the state dictionary has been loaded, the encoder can either be frozen, or it can be fine-tuned with the tagger. It is important

to note here that the embedding layer for the encoder was created using the NMT source vocabulary. Each node of the input dimension represents a specific token. As a result, the source vocabulary for training the ET model must be the same source vocabulary that was used by the NMT model.

Tagger

The tagger consists of one or two hidden layers and a fully connected output layer. The hidden layer or layers operate as tagging layers that take the representations from the NMT encoder as input. The fully connected output layer maps from the hidden layer to the number of possible tags. Available hyperparameters for the tagger include the number of hidden layers, the size of hidden dimension, and the percentage of dropout.

Predicted Tags

The predicted tags for each sequence are determined by obtaining the maximum-valued dimension for each token in the sequence. This is defined as:

$$\text{argmax}(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n) = \text{argmax}(\hat{y}_1), \text{argmax}(\hat{y}_2) \ldots \text{argmax}(\hat{y}_n))$$

where $\text{argmax}(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$ is the entire predicted sequence. Each prediction is independent and Viterbi decoding is not used. Instead, we rely entirely on the sentence representations from the NMT encoder to contain sufficient information to predict the correct token at each time step.

3 Experiments

This chapter provides details on the architecture and training of the NMT and ET models. In addition, we describe the experiments and their results.

3.1 Neural Machine Translation

For the NMT models, the architecture was similar to that used by Belinkov et al. (2017). The embedding layer had 300 dimensions and a dropout of 0.5. The LSTM had two bidirectional layers, 512 hidden units, and a dropout of 0.5. The decoder had a similar architecture with the addition of a final output layer: The embedding layer had 300 dimensions and a dropout of 0.5. The decoder LSTM consisted of two bidirectional layers, 512 hidden units, and a dropout of 0.5. The hidden layer in the decoder module had 512 hidden units. Batches contained 64 sequences. All the models were trained for 5 epochs. The model from the epoch with the best validation loss score was used for the ET model experiments.

In order to improve training, sequences were "bucketed", or sorted into sequences of equal length, before being batched. Without bucketing, sequences are batched either sequentially or randomly. For a data set where the maximum sequence length is 55 tokens, this results in the majority of the batches being around 50-55 tokens in length. As a result, shorter sequences may contain more padding than content and training takes a long time. After the sequences were sorted, the data loader created batches by pulling only sequences that were the exact same length. For example, with a batch size of 64, the data loader might pull 64 sequences that were all 14 tokens long. This resulted in none of the batches containing any padding. However, if there were not enough 14-token sequences, that batch would contain fewer than 64 sequences. While the NMT models may have benefited from a lack of padding, it is unclear what the impact of having the smaller batches might have been on the models.

The models were evaluated on their respective test sets using BLEU, which compares the predicted translation with a reference translation (Papineni et al., 2002). The goal of

Target Language	BLEU Score
English	100
Spanish	75
French	68
Russian	54
Mandarin	47

Table 2: BLEU scores for the NMT models.

these experiments was not to beat state-of-the-art performance on NMT tasks but rather to get reasonably good models on which to train the ET models. BLEU scores for these experiments are shown in Table 2. The auto-encoder model, which translated from English-to-English, obtained a BLEU score of 100, which indicates perfect translation. This is perhaps unsurprising given that each word is translated straight back into itself. However, it raises the question of whether the model has retained any information about the language, or if the model has only learned to directly copy from the source language to the target language. The Romance language models, which translated into Spanish and into French, had higher BLEU scores than the models which translated into Russian and Mandarin.

3.2 Part-of-Speech Tagging

Our ET architecture was also similar to that used by Belinkov et al. (2017). The ET encoder had the exact same architecture as in the NMT model. The embedding layer had 300 dimensions and the LSTM had two layers, had 512 hidden units, was bidirectional, and had a dropout of 0.5.

The tagger had one hidden layer with 1024 hidden units and a dropout of 0.3. The fully connected layer mapped from the hidden layer to the number of tags in the POS tagging set. Batches of 128 were used. The ET models were trained for 30 epochs, and a checkpoint was saved at each epoch. The model with the lowest validation accuracy was used for evaluation.

Train Set	Test Set	Accuracy
OntoNotes	OntoNotes	.9646
WMT	WMT	.9719
OntoNotes	WMT	.9570
WMT	OntoNotes	.9201

Table 3: Baseline accuracies for OntoNotes and WMT data sets.

3.3 Baseline

As a baseline, we trained HunPos taggers (Halácsy et al., 2007) on the OntoNotes and WMT data sets. We then tested each model on both the OntoNotes and the WMT test sets. HunPos is a hidden Markov model-based POS tagger, an open-source implementation of the TnT tagger (Brants, 2000). The model estimates the probability of a certain tag based on the likelihood of the token at that time step given that tag as well as the likelihood of the previous two tags and this tag occurring in a sequence. The results of the baseline experiments are shown in Table 3.

We do not anticipate that the ET models will be able to outperform the baseline because the encoders are not specifically trained for POS tagging and because the tagger is kept deliberately small. However, it is worth noting the differences in performance on each of the data sets. The models with the best accuracy were those that trained and tested on the same data sets. Conversely, the models that trained on one data set and evaluated on another performed less well. An additional finding was that the OntoNotes model had higher accuracy when evaluated on the WMT test set than the WMT model did when evaluated on the OntoNotes test set. This suggests that it may have greater generalizability despite being a significantly smaller data set.

Model	Data	Target Language	Total Acc.	In-vocab	OOV
OntoNotes	OntoNotes	English	.6947	.7301	.3259
		French	.8512	.8905	.4465
		Mandarin	.7960	.8282	.4612
		Russian	.8128	.8456	.4737
		Spanish	.8603	.8971	.4776
WMT	WMT	English	.6572	.6988	.2079
		French	.8202	.8735	.2496
		Mandarin	.7630	.8069	.2892
		Russian	.7766	.8231	.2776
		Spanish	.8291	.8783	.3019
OntoNotes	WMT	English	.4227	.4472	.2569
		French	.4991	.5247	.2230
		Mandarin	.4770	.4986	.2415
		Russian	.4807	.5004	.2688
		Spanish	.5057	.5296	.2457
WMT	OntoNotes	English	.4343	.4669	.0952
		French	.5027	.5394	.1247
		Mandarin	.4938	.5259	.1589
		Russian	.4813	.5147	.1349
		Spanish	.5151	.5503	.1507

Table 4: Categorical accuracy for POS tagging by target language.

3.4 Encoder Representations

Initial experiments explored whether the output from the encoder contained enough information about each token to perform POS tagging. We trained ET models on the gold-tagged OntoNotes data and the WMT data, which used predicted tags. Then, we evaluated each model on their own test set and the test set of the other data set. The results are shown in Table 4.

We looked at overall accuracy as well as accuracy on in-vocabulary and out-of-vocabulary (OOV) tokens. OOV tokens are tokens in the test set that did not appear in the training set. Since the tokens did not appear during training, the model does not have a word embedding for them and they are all mapped instead to the same "unknown" word embedding and have

Target Language	Layer	In-Vocab	OOV
English	Embedding	.7530	.0000
	1st LSTM	.7943	.5043
	2nd LSTM	.7301	.3259
French	Embedding	.8857	.0018
	1st LSTM	.9370	.6081
	2nd LSTM	.8905	.4465
Mandarin	Embedding	.8404	.0001
	1st LSTM	.8947	.5905
	2nd LSTM	.8282	.4612
Russian	Embedding	.8667	.0003
	1st LSTM	.9142	.5726
	2nd LSTM	.8456	.4737
Spanish	Embedding	.8819	.0092
	1st LSTM	.9377	.6042
	2nd LSTM	.8971	.4776

Table 5: Accuracy for POS by language and layer for OntoNotes data set.

the same representation. Uniquely, the embedding layer in the ET model was trained in the NMT model, so each of the embeddings are from the NMT vocabulary. As a result, OOV tokens here are tokens which were not in the NMT vocabulary. The NMT vocabulary was relatively large consisting of 50,000 vocabulary items. Nevertheless, 8.2% of the WMT data set and 8.8% of the OntoNotes data set consisted of OOV tokens. Importantly, this was true regardless of which data set the ET model trained on; the vocabulary had already been set by the NMT model and was not impacted by any information from the ET training sets.

3.5 Encoder Depth

Our next experiment explored the information captured by each layer of the encoder. The results are shown in Table 5. There are three layers in the encoder: the embedding layer, the first LSTM layer, and the second LSTM layer. The second LSTM layer is the last and final layer of the encoder and there are no additional layers, so the representations from this layer are the same as the representations from the entire encoder, shown in Table 4.

4 Discussion

In this chapter, we analyze the results of the experiments and explore the effects of various features on the results.

4.1 POS Tagging Results

As shown in Table 4, the ET models performed reasonably well on the POS tagging task. The model with the highest overall accuracy, the English-to-Spanish OntoNotes model, had an accuracy of 86.03%, which is only 10% below the accuracy of the HunPos model on the same dataset and 6% below the HunPos model with the lowest accuracy. This shows that the representations contain adequate linguistic information for performing POS tagging. In fact, the ET models were also able to accurately predict OOV tokens demonstrating that the representations also contained syntactic information about the positions of each of the tokens in the sequence.

4.2 Target NMT Language

We now look at the relationship with the encoder representations and the target NMT language. Belinkov et al. (2017) found a partial correlation between tagging performance and BLEU scores. We also found a correlation between BLEU scores and tagging accuracy in all models except for the auto-encoder. For the models that translated from English into another language, if the model had a higher BLEU score than another model, it also consistently had a higher tagging accuracy. This was true across all of the models and datasets. Yet, the auto-encoder, which had had a perfect BLEU score, had the worst tagging accuracy both for overall accuracy and for in-vocabulary and OOV tokens showing that it built weaker word and sentence representations. This is perhaps due to the fact that the auto-encoder copies the text directly rather than performing transformations like the translation models. Nevertheless, the auto-encoder still learned linguistic information; it was able to correctly

predict OOV tokens, which requires syntactic and contextual understanding.

In addition, Belinkov et al. (2017) determined that better representations were created when translating into morphologically impoverished languages. To explore this further, we experimented with the opposite set up. We translated instead from a morphologically impoverished language, English. The target languages were three morphologically rich languages: French, Spanish, and Russian and two morphologically impoverished languages: English and Mandarin. We found no correlation between the morphological richness of a language and the performance of the ET model. This suggests again that the morphological richness of the target language may not be an important feature when translating from a morphologically impoverished language.

4.3 Impact of Data Set

In keeping with the findings of the HunPos baseline experiments, models which trained and tested on the same data set had the best tagging accuracy scores. However, based on the HunPos experiment, we had found that the OntoNotes model had been able to generalize better. This was uniquely interesting given that WMT is a significantly larger dataset, which provides more data for the model to learn. In the ET experiments, however, we find that the WMT model performs slightly better when tested on the OntoNotes testing set. However, when we look at the performance on in-vocabulary and OOV tokens in Table 4, we can see important differences between the performances of the models. While the WMT model has higher accuracy on in-vocabulary tokens than the OntoNotes, it has noticeably lower accuracy on OOV tokens. Thus, the WMT model seems to have built better representations of individual tokens perhaps due to the larger corpus size. Conversely, the OntoNotes model seems to have built stronger syntactic representations, which supports the findings of the HunPos experiments. This may reflect the use of automatically generated tags on the WMT data set compared to the hand-annotated tags for the OntoNotes data set; small mistakes

in the tagging for the WMT data set may have led to confusion at the syntactic level.

4.4 Effect of Token Frequency

Belinkov et al. (2017) found that POS taggers perform better on tokens that occur more frequently in the corpus. We also saw that higher frequency tags tend to have higher accuracy rates. We found a strong correlation between word frequency occurs and the POS tagging accuracy ($\rho = .1531$). It is likely that the representations for more frequent tokens are more informative due to increased training opportunities. This supports the idea that the WMT model had better representations due to being trained on a larger data set with more tokens.

4.5 Effect of Tag Frequency

While our models performed well on the tagging tasks overall, for several, less frequently occurring tags, the models were not predicting the tags accurately or at all. We compared tag frequency with prediction accuracy and found a strong correlation between tag frequency and accuracy. This holds both for predictions for the first LSTM layer ($\rho = .5677$), the second LSTM layer ($\rho = .5559$), and to a lesser degree, the embedding layer ($\rho = .4502$). This shows that tag frequency is an important factor in the data set, but that the impact may be stronger at the lower levels of the encoder.

We also wanted to separate the effect of word accuracy from token accuracy and focus more on the syntactic information. We looked at sequences of two (bigram) and three (trigram) tags and contrasted how frequently that sequence occurred with how likely the model was to predict the last tag in the sequence. The results are shown in Figures 1 and 2. We found a strong positive correlation between the sequence frequency and a correct tag prediction for the last tag in the sequence. This indicates that the models relied on sequencing information in addition to knowledge of individual works to tag tokens.

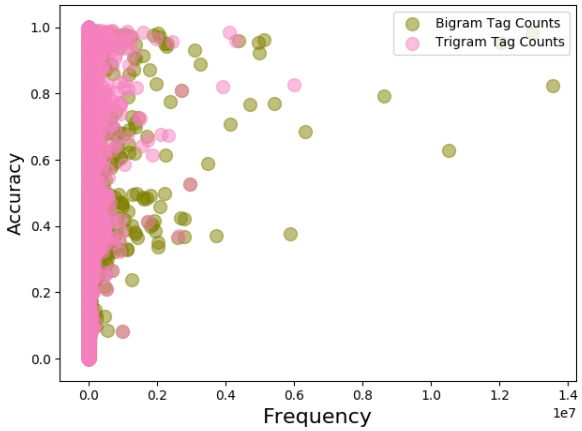

Figure 1: Bigram and trigram tag frequency and accuracy ($\rho = .3215$).

4.6 Encoder Depth

Belinkov et al. (2017) had the same encoder architecture as our NMT encoder. They found that the best representation for POS tagging came from the first layer of the LSTM. They posited that the higher embedding layer of the encoder focuses more on word meaning while the lower LSTM layers learn word and sentence structure. As can be seen in Table 5, we found a similar pattern. The second layer of the encoder consistently had the best results for POS tagging. We investigate why by looking at the accuracy for in-vocabulary and OOV items from each of the layers.

The first layer of the encoder is the embedding layer. It takes each token as input and projects the input into an n-dimensional vector. Notably, no sequential information is available at this layer and, therefore, the layer focuses entirely on individual words. For example, the sequence "man bites dog" and "dog bites man" are represented in the same way except for having a different word order. As a result, we would not expect any syntactic information to be available at this level. Thus, we can see that at the embedding level, the in-vocabulary predictions are reasonably good, but the OOV predictions are low across all the models and languages. Since no syntactic information is included in that layer, the

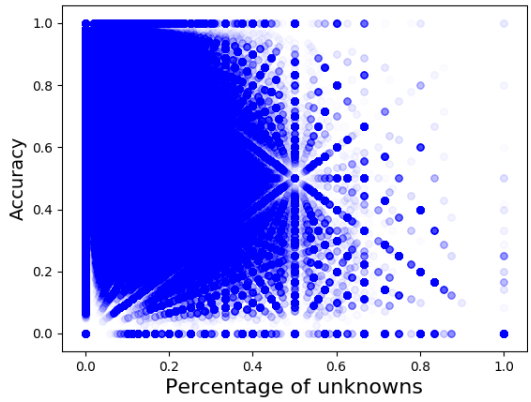

Figure 2: Percent of OOV items and accuracy ($\rho = -.1873$).

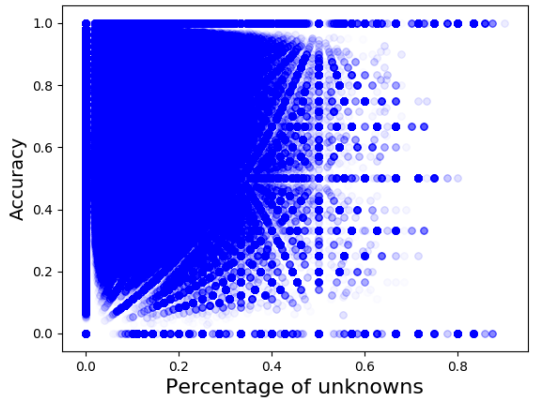

Figure 3: Percent of OOV items and in-vocabulary accuracy ($\rho = .0080$).

model has difficulty predicting tags for OOV tokens.

In contrast, the next layer of the encoder is the first layer of the LSTM. We see that most of the models have the best accuracy rates at this layer. This is most likely because this layer contains a combination of individual word representations, produced by the embedding layers, and contextual information, gained from the LSTM. For all of the models, both the in-vocabulary and the OOV token prediction accuracies improve from the embedding layer to the first LSTM layer.

Finally, the last layer of the encoder is the second layer of the LSTM. While all the models have a lower overall accuracy, they improve on OOV token accuracy. This suggests that the final encoder layer contains a significant amount of syntactic information which it can utilize to predict OOV tokens more accurately.

4.7 Effect of Out-of-Vocabulary Tokens

If the models were relying heavily on syntactic information, a high percentage of OOV tokens in the sequence would make tagging more difficult. As shown in Figure 2, we find that there is a significant decrease for overall accuracy as the percentage of OOV tokens decrease.

However, it is unclear whether this is due to a decrease in the quality of the syntactic representations or if the model has a lower accuracy at predicting OOV tokens and having more of them in a sequence leads to lower accuracy for that sequence. We investigate by looking at in-vocabulary and OOV tokens. As shown in Figure 3, we find that the accuracy for the in-vocabulary tokens remains the same regardless of the percentage of OOV tokens in the sequence. We can conclude from this that the models are relying more heavily on their knowledge of individual words in order to tag in-vocabulary items.

Conversely, from looking at the encoder depth, we have found OOV tokens are predicted mostly based on syntactic information. Thus, we would anticipate that having a higher percentage of OOV tokens would make it harder to predict the tags for OOV tokens especially at the lower encoder levels. As shown in Figure 4, the percentage of OOV tokens has only a small impact on the embedding layer representations. This is especially interesting because based on our earlier analysis, we had not anticipated any correlation at all. This suggests that some syntactic information may be available in the embedding layer after all. Additionally, as seen in Figures 5 and 6, a higher percentage of OOV tokens decreases the tagging accuracy when for representations taken from the LSTM layers. This shows that OOV tokens are impacted by a decrease in syntactic information when there are a higher percentage of OOV tokens in the sequence.

4.8 Effect of Sequence Length

We also looked at the effect of sequence length on accuracy. Due to the fact that neural networks often have difficulties with long sequences (Pouget-Abadie et al., 2014), it is possible that our ET models would have decreased accuracy as sentences got longer. Gorman and Bedrick (2019) also found that POS tagging models have difficulty with very short sequences. They posit that because short sequences are often titles, the capitalization leads to the over-prediction of proper nouns. As shown in Figure 7, we found that both very shorter and very

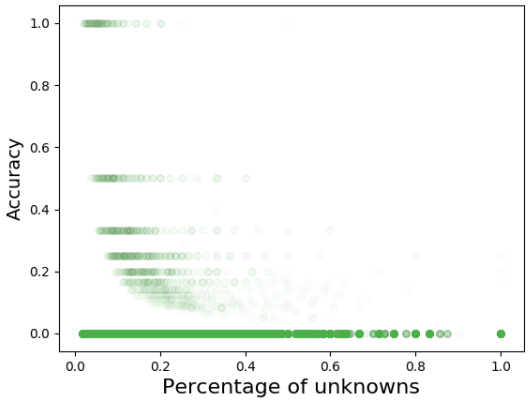

Figure 4: Percent of OOV items and accuracy embedding layer ($\rho = .0157$).

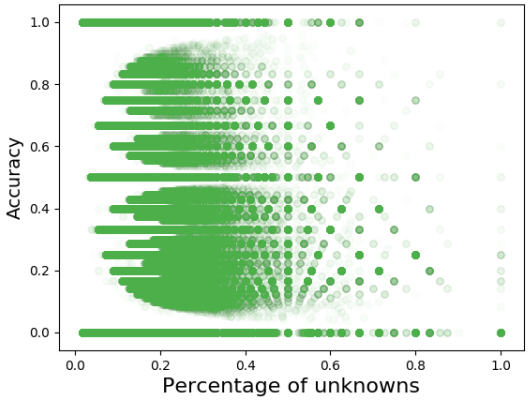

Figure 5: Percent of OOV items and accuracy 1st LSTM layer ($\rho = -.0313$).

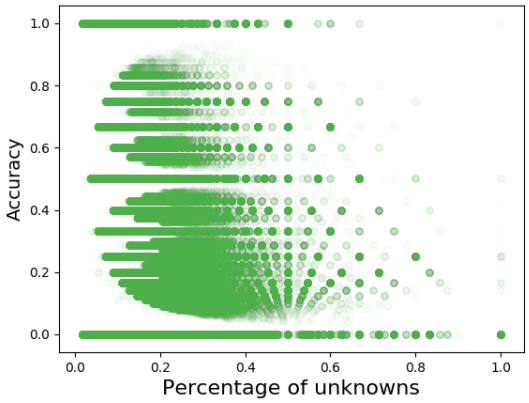

Figure 6: Percent of OOV items and accuracy 2nd LSTM layer ($\rho = .0742$).

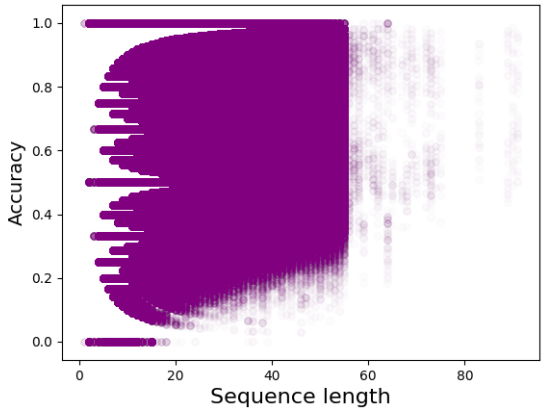

Figure 7: Sequence length and accuracy (ρ =.0692) .

long sequences have less consistent accuracy rates, which is in line with previous research. However, it also is important to note that the NMT models were trained on source and target sequences between 5 and 55 tokens in length. As a result, the encoder may have had less syntactic understanding of sequences above and below that length, which may have had an impact on the accuracy.

5 Conclusion

Within this chapter, we analyze and evaluate the results of our experiments and make recommendations for future work.

5.1 Summary

Our first experiment addressed the question of whether the representations from NMT encoders can be used in POS tagging. We determined that the representations contained enough information to perform the POS tagging task with our best model obtaining an accuracy that was only 10% below a reasonable baseline created using the HunPos POS tagger. In addition, the models were able to accurately predict POS tags for a significant portion of the OOV tokens showing that syntactic and contextual information is contained in the representations. Moreover, we found that a smaller data set may be effective as a larger one, especially when building syntactic representations.

Next, we explored the information stored in each layer of the encoder. In line with previous research, we determined that the most precise word representations occurred at the embedding layer, while the most informative syntactic representations occurred at the second LSTM layer, and a combination of both occurred at the first LSTM layer. As a result, all the models had the best overall accuracy at the first LSTM layer.

Then, our experiments focused on the impact of target language on encoder representations. We found a correlation between BLEU scores and tagging performance showing that higher BLEU scores resulted in better accuracy. However, we discovered that while auto-encoders had high BLEU scores, they do not seem to learn as much linguistic information as translation models suggesting that the act of translating is important for learning syntactic and morphological information. In addition, no relationship between the morphological richness of the target NMT language and accuracy in POS tagging was found.

In addition, we analyzed the results and found correlations between various features of

the training data and the model accuracy. We found that there is a strong relationship between word frequency and accuracy suggesting that for the best results, a very large training data set may be needed. Tagging accuracy also increases in direct relation to tag frequency in the data set. In addition, we looked at bigram and trigram tag sequences and found that the more frequently a tag sequence occurred, the more likely a model was to correctly predict the last tag in the series. We also explored how the percentage of OOV tokens in a sequence impacted the tag predictions for that sequence. While in-vocabulary tokens were not impacted, tagging for OOV tokens had lower overall accuracy rates, which led to lower accuracy rates for the sequence as a whole. This suggested that for tagging in-vocabulary tokens, the models rely on knowledge of the individual words while for OOV tokens, contextual information is more important. Lastly, we saw the models predicted best on average length sequences, which were not overly short or long, which is keeping with what previous studies have found.

5.2 Future Work

In our experiments, we focused on POS tagging. There are many additional tagging or classification tasks that could also be explored such as named entity recognition or chunking, which involves extracting individual phrases from sentences. In addition, all of the languages we studied are subject-verb-object (SVO) sentence structure languages. Exploring languages with more diverse syntactic structures may provide more information about how syntactic information is learned. Moreover, the addition of more linear layers might allow for exploration into how much syntactic information a linear layer can learn compared to an LSTM. Additionally, we did not use any decoding algorithm for decoding the ET output instead relying on the representations to contain this information. If a decoding algorithm was used, the accuracy may be able to reach that of the baseline HunPos experiments. It would also be interesting to take a closer look at why some data sets generalize better and what features

of those data sets allow them to generalize. We leave these experiments for future work.

Bibliography

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., and Glass, J. (2017). What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Brants, T. (2000). TnT – a statistical part-of-speech tagger. In *Sixth Applied Natural Language Processing Conference*, pages 224–231, Seattle, Washington, USA. Association for Computational Linguistics.

Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv:1601.01073 [cs, stat]*. arXiv: 1601.01073.

Gorman, K. and Bedrick, S. (2019). We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2786–2791, Florence, Italy. Association for Computational Linguistics.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. *arXiv:1803.11138 [cs]*. arXiv: 1803.11138.

Halácsy, P., Kornai, A., and Oravecz, C. (2007). HunPos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 209–212, USA. Association for Computational Linguistics.

Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Kim, N., Patel, R., Poliak, A., Xia, P., Wang, A., McCoy, T., Tenney, I., Ross, A., Linzen, T., Van Durme, B., Bowman, S. R., and Pavlick, E. (2019). Probing what different NLPtasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*. arXiv: 1412.6980.

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

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Poliak, A., Belinkov, Y., Glass, J., and Van Durme, B. (2018). On the evaluation of semantic phenomena in neural machine translation using natural language inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 513–523, New Orleans, Louisiana. Association for Computational Linguistics.

Pouget-Abadie, J., Bahdanau, D., van Merrienboer, B., Cho, K., and Bengio, Y. (2014). Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *arXiv:1409.1257 [cs, stat]*. arXiv: 1409.1257.

Shi, X., Padhi, I., and Knight, K. (2016). Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Subramanian, S., Trischler, A., Bengio, Y., and Pal, C. J. (2018). Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv:1804.00079 [cs]*. arXiv: 1804.00079.

Sun, J. (2012). *Jieba chinese word segmentation tool*. https://github. com/fxsjy/jieba.