Neural Machine Translation: Encoding and Attention Implementation and Comparisons

Millie Dwyer mmd378 **Eduardo Fierro** eff254 **Akash Kadel** ak6201 **Shasha Lin** sl4964

Abstract

In this paper we compare two methods of encoding for the Neural Machine Translation (NMT) task on the English to French Ted Talk transcript WIT^3 dataset (Cettolo et al., 2012). We further explore the impact of different attention scoring functions (dot-product and concatenation based) implementations on these results. We found that sub-word encoding works best on this dataset for both methods of attention, while dot-attention performed better within both encoding methods, supporting previous results found separately for sub-word encoding and attention methods on different datasets. We additionally found an error with a PyTorch tutorial implementation of concat-attention, which unfortunately impacts our results. This paper is our final project for the Fall 2017 DSGA-1011 course at NYU's Center for Data Science. [1]

1 Introduction

The encoder-decoder based neural network architecture has been proven effective in machine translation tasks, surpassing state-of-the-art conventional phrased-based machine translation systems (Sutskever et al., 2014) in performance. Many modifications have been made to the original word-based encode-decoder model, such as pervasive dropout (Sennrich et al., 2016a) and the attention mechanism (Bahdanau et al., 2014), enabling further performance boosts. In addition to improvements in model architecture, preprocessing techniques resulting in different language units (e.g. words vs subwords) for model input, have been shown to greatly impact model performances as well (Sennrich et al., 2016a).

Since the initial introduction of the attention mechanism in machine translation tasks (Bahdanau et al., 2014), variants of the scoring functions in the attention alignment have been shown to successfully improve translation of lengthy sentences (Luong et al., 2015). On the other hand, researchers have found that encoding the language data with methods such as byte-pair-encoding (Sennrich et al., 2016a) or simply using character as units of model training (Chung et al., 2016) improves the translation of rare words (Sennrich et al., 2016a) and enables the efficient representation of different words sharing the same morphemes or lexemes (Chung et al., 2016).

In this project, we evaluate different combinations of the attentions and language units on the 01-2016 edition of Web Inventory of Transcribed and Translated Talks (WIT^3) dataset (Cettolo et al., 2012), translating from English to French. We particularly look to build upon previous NMT work by combining the character-learning-unit method with the dot scoring function for attention (Luong et al., 2015), and comparing this with byte-pair-encoding (BPE) counterparts.

2 Literature Review

2.1 BPE vs Character Encoding

Although using words as the basic unit to train machine translation models is an intuitive method, it presents many problems that make learning difficult.

[1] All code used for this project may be found at: https://github.com/sl4964/MICA_NeuralMachineTranslation

Words often share smaller units and sub-sequences, such as morphemes and lexemes. To represent a large vocabulary efficiently on a word level, a superior word segmentation algorithm is required. One popular such method is BPE, which was first applied to natural language processing tasks in order to tackle the translation of rare words by (Sennrich et al., 2016a). Initially used for data compression that works on pairs of bytes, BPE iteratively merges units of language, starting from single characters to longer and longer character sequences (subwords), based on their frequencies in the corpus, until a set number of merge steps are completed. The use of BPE is a crucial component for many successful models, making several best-performers in the WMT 2016 shared news translation task (Sennrich et al., 2016a).

While certain segmentation techniques such as BPE have been shown to be more successful than others (Chung et al., 2016), word segmentation is a difficult task on its own, and using a segmentation method trained or designed separately from the translation task may result in suboptimal performance. Using characters as units of learning requires minimal preprocessing steps, and circumvents the need to use a separate segmentation algorithm (Lee et al., 2016). (Chung et al., 2016) have shown that models using characters as units for decoders outperform those with BPE-segmented subword units.

2.2 Attention Mechanism with different scoring functions

In an encoder-decoder machine translation model, such as (Cho, 2015), the encoder is responsible for learning a context vector, conditioned on which the decoder predicts each word in the target sentence. The context vector c is a function of the hidden states of the encoder $h_i, i \in 1, 2, ...T$:

$$c = q(h_1, h_2, .., h_{T_X})$$

The decoder learns the distribution of possible translations at time t conditioned on c and its previous hidden state z_{t-1}, and its previous translated word $\tilde{y}_{t-1}$:

$$z_t = f(z_{t-1}, \tilde{y}_{t-1}, c)$$

Contrary to vanilla encoder-decoders, the context vector c in attention-based models is not the encoder hidden state from its last time step h_{T_X}, but a weighted average of all hidden states $h_1, h_2, .., h_{T_X}$. For time t on the decoder side, a score $e_{j,t}$ is computed for the encoder hidden state at time j (h_j):

$$e_{j,t} = f_{score}(z_{t-1}, h_j)$$

before a softmax is applied to normalize this score:

$$\alpha_{j,t} = \frac{exp(e_{j,t})}{\sum_k^{T_X} exp(e_{k,t})}$$

Finally, the context vector for predicting t^{th} word in the output sequence is calculated as

$$c_t = \sum_{j=1}^{T_X} \alpha_{j,t} h_j$$

Initially proposed in (Bahdanau et al., 2014), (Luong et al., 2015) generalized the scoring function f_{score} to include different formulations, and made the computation graph simpler by using the current decoder hidden state z_t in computing $e_{j,t}$ instead of the previous one z_{t-1}. (Luong et al., 2015) also showed that attention mechanisms in general greatly improve translation performance, with some formulations of the scoring function outperforming others. In the Model Architecture section, we explain the two scoring functions compared in our project.

3 Methods

3.1 Dataset

The WIT^3 dataset contains transcript translations for ted talks pulled from 2007 through 2016. The English to French translation of this dataset has been previously used by (Bentivogli et al., 2018) for evaluating NMT systems pre-trained on WMT 2015 dataset, comparing their improvements over canonical phrase based machine translation methods. We do not expect similar levels of performance as presented in (Bentivogli et al., 2018), given the time and resource constraints of this course. (Bentivogli et al., 2018) further take advantage of pre-training on separate corpora, which we did not replicate.

3.2 Pre-processing

Following (Lee et al., 2016), we pre-processed the text by first normalizing punctuation and then tokenizing the text[2]. This was the input for the BPE

[2]This uses the non-breaking prefixes corpus from http://www.loc.gov/standards/iso6392/php/code_list.php

algorithm. For the self-written character encoding function, we used only the normalized files. The 2016-01 version of the WIT^3 dataset is equipped with train, validation, development, and test sets. Table 4 of the Appendix shows the relative sizes of these datasets. For our experiments, we present results on the 2012, 2013 and 2014 test sets, while the original training set and validation set are used for model development and selection. For all of our experiments, the English input sentences are encoded using BPE. We vary the encoding of the French output sentences between BPE and character level encoding. The details of these encoding methods are described below.

3.2.1 BPE

Following (Sennrich et al., 2016b), we made sure that no pairs were considered that crossed boundaries. That is, no BPE's between words are learned.

```
It &apos;s a colon@@ ial animal .
C&apos; est un animal coloni@@ al
.
```

Example 1: Example on bpe2bpe from train set

As seen in Example 1, for both English and French sentences "colon i" is included as part of a split strategy into BPEs. In our implementation, for both English and French the algorithm was limited to the creation of 20,000 new symbols.

3.2.2 Character Decoder

The character encoding simply separates all characters in the reference translated sentence and splits all characters with an extra space. On the reference original sentences (in this case, english).

```
It &apos;s a colon@@ ial animal .
C ' e s t   u n   a n i m a l   c
o l o n i a l .
```

Example 2: Example on bpe2char from train set

3.3 Model Architecture

We used the classic encoder-decoder model used in (Sutskever et al., 2014) and (Chung et al., 2016), with the attention mechanism proposed in (Luong et al., 2015). We explored two different scoring functions in the attention weight calculation.

In (Bahdanau et al., 2014), the scoring function for current decoder hidden state t with encoder hidden state j uses the previous decoder hidden state z_{t-1} as input: and uses the the scoring function:

$$\begin{aligned} e_{j,t} &= f_{score}(z_{t-1}, h_j) \\ &= v^T tanh(W \cdot z_{t-1} + Uh_j) \end{aligned} \tag{1}$$

where v is a vector, and W, and U are matrices.

In (Luong et al., 2015), another version of scoring function is proposed where current decoder hidden state z_t is used instead of the previous one, and the tanh function is replaced by a simple dot product:

$$e_{j,t} = z_t \cdot h_j \tag{2}$$

(Luong et al., 2015) found that models using a tahn layer for the scoring function performs worse than using a simple dot product as in 2 in (Luong et al., 2015). However, their implementation of the tanh layer has a slightly different formulation than (Bahdanau et al., 2014):

$$\begin{aligned} e_{j,t} &= f_{score}(z_t, h_j) \\ &= tanh(W \cdot concat[z_{t-1} : Uh_j]) \end{aligned} \tag{3}$$

In our project, we compared scoring functions (3) and (2) in the same codebase, and hypothesize that we would have the same finding as (Luong et al., 2015) on our new dataset: the scoring function with simple dot product(2) would outperform the concatenation one (Equation 3).

3.4 Loss Function and Evaluation

To train our encoder-decoder systems, we use a masked cross entropy loss function as seen in the PyTorch tutorial (Robertson, 2017).

The loss is computed after processing every batch. It is similar to cross entropy on a per sample, per step basis. For every sample sentence, the calculated loss is normalized by dividing by the length of the sentence. The loss formulation is shown below:

$$Loss = -\sum_{b=1}^{B} \frac{\sum_{i=1}^{max_L} \sum_{p=1}^{V} 1.\mu_{p==y_{b,i}}}{l_b},$$

where,
B = batch size;
max_L = maximum length in the batch
V = vocabulary size
l_b= length of the b^{th} input sequence
$\mu_{p==y_{b,i}}$ = softmax probability value for predicting p^{th} word

3.5 Training Details

We focus on the tuning of few parameters, leaving the rest heuristically fixed according to recent literature on the topic (Luong et al., 2015). All the models tested had only two layers for the encoder and the decoder. We did a random search with hyper-parameters including learning rate, (sampled from uniform distribution between 0.0001 and 0.001), the hidden/embedding size (uniformly sampled between 512 and 1024), and beam-search width (uniformly sampled between 5 and 18). We also varied other parameters such as dropout (.1 vs .2), batch size (16, 32, 64, 80), for each of the models we ran.

We alternatively explored using Adam and SGD optimizers, as well as some learning rate decay in the late-stage of trials where loss seemed to be stabilizing. The impact of this variation is discussed below with respect to the character-level dot-attention decoder.

Following (Bengio et al., 2015), we used a inverse sigmoid decay schedule:

$$\epsilon_i = k/(k + exp(i/k))$$

where ϵ_i is the probability of using teacher forcing for each batch, i is total number of training iterations, and k is chosen based on expected speed of model convergence. We expected our models to converge after 50,000 iterations, and made the decision to set k at 3000, which makes teacher forcing ratio equal .41 after training for half the expected converging time. These values were fixed for all trials.

BPE tokenization of the data files led the average English training sentence to be composed of 105 ± 78 characters, while the French BPE tokenized training sentences are 120 ± 92 characters long [See Table 4 in the Appendix]. Because of these statistics, we trained our model on pairs that had a maximum of 200 characters per sentence, trimming in order to avoid the computational cost of using longer outlier pairs. We similarly limited the data to sentences longer than 5 characters, although this trimming had less of an impact on the overall data size. [Table 5 in Appendix].

4 Results and Analysis

We initially aimed to compare BPE and character level encoding across both dot- and concat-attention methods. We found that our dot-attention implementation was more stable and ultimately gave better results. We implemented the dot-attention model later on in our development process, and only realized some critical issues with our concat-attention (Bahdanau et al., 2014) implementation through comparison with dot-attention (Luong et al., 2015). The results below are focused on comparing BPE and character level encoding for dot-attention, but results for concat-attention are included as well. More explanation of this issue may be found in the appendix.

4.1 Model Comparison

The best performing model, as seen in Table 2 and 3, was reached using the dot-attention mechanism and byte-pair encoding to byte-pair encoding. This model was reached with a constant learning rate of 0.0002315, a hidden size of 738 and a beam search parameter of 15. This was reached after 7,200 batches of size 80, which took 144 hours on NYU's Prince[3].

To analyze the attention on all the models, we selected from one of the test sets 2 examples[4]: one that may look relatively easy to translate and a more challenging example. For the dot-attention bpe2bpe example, this can be seen in 1 and 2 respectively.

For the easy example in Figure 1, the model has no problem recognizing the relevant words for translating. The translation generated by the model only diverged with the original translation by "est", which originally was "c'est". But perhaps the more interesting case is the difficult example, as seen in

[3]As for the day of the submission of this project, there were 244 GPU cards available on Prince, each with different performance, from NVIDIA Tesla K80 to NVIDIA GTX 1080 to others. This project didn't run in one single card, which may strictly vary time comparisons. For more information visit https://wikis.nyu.edu/display/NYUHPC/High+Performance+Computing+at+NYU

[4]This was done after computing the final BLEU scores to ensure their validity

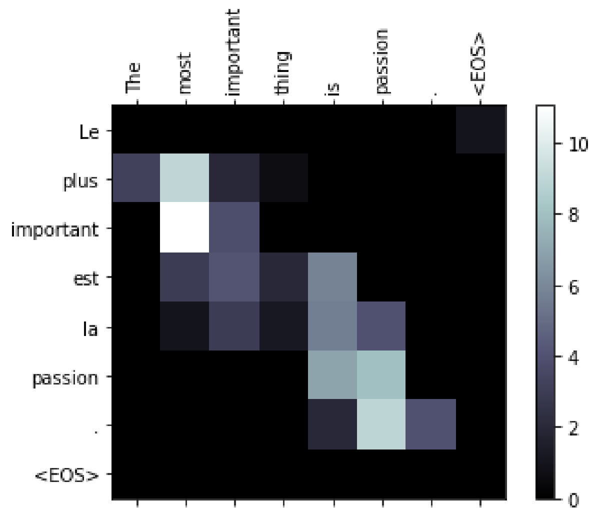

Figure 1: Luong Dot Attention For bpe2bpe model Example 1

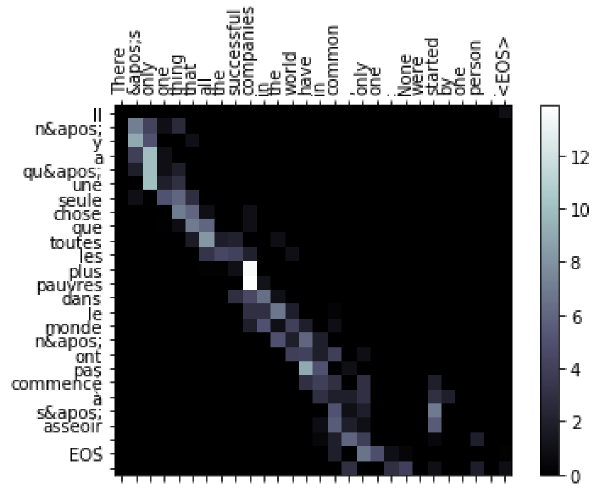

Figure 2: Luong Dot Attention For bpe2bpe model Example 2

Figure 2. The model clearly starts with a right foot the translation, with “Il n’y a qu’une seule chose que toutes les plus pouvres dans le monde” (which means “There is only one thing that all the most powerful”). But then it suddenly diverges with “n’ont pas commence”. This last word “commence” (which means “start”) is driven by “common” in English, which again is translated to french as “commun”.[5]. After this point, the model completely diverges to “sit down”, which although is grammatically correct in French, does not correspond with the source sentence in meaning.

Our best model with dot-attention scoring and character decoder did not perform as well as the BPE decoder model. Looking at the same examples clearly shows the suboptimal performance of this model (see figure 6 in Appendix B): In the easy example, the model is able to translate the phrase ”the most important thing” correctly, but fails to translate the subject of the sentence, ”passion”. In the hard example, the model is able to correctly align and translate easy words such as ”in” and ”thing”, and aligns ”the world” with a semantically close phrase, ”la mondalisation (globalization)”. Based on the learning curve [Figure 5 in Appendix B], we think the inaccuracies in translation are a result of stopping training before converging, and suspect that if we train the model for longer time the translation result could improve. Our result do not necessarily show that character decoder perform less well than BPE decoder, but simply that they may take longer to train, given that sentence lengths in characters can be an order of magnitude longer than sentences in words.

Contrary to the dot-attention example, the concat-attention model for bpe2bpe as seen in Figures 7 and 8 in Appendix D, fixes all attention on a single word. In the first case, it comes from “thing” and in the more difficult case from “one”. Strangely, for the first example the model is still capable of returning a moderately good translation, which re-translated to English means “The trick is the important thing”. That is, the model diverged from an almost perfect translation for one word: trick to passion. For the second example, the translation makes no sense. Its attention relies heavily in the word “one”, and consequently has three times the word “seul” and two times “seulement” (both meaning“only”), and two times “une” (both meaning“one”). This model was trained with a learning rate of 0.0006165, hidden size of 786 and beam search of 9. Nonetheless, it only trained for 3000 batches of 80 pairs. In a similar fashion, our best model using concat-attention and the character level decoder does not achieve the same performance as the best dot-attention model.

[5]The original translation in the test set for this example is “Toutes les socits qui russissent dans le monde n’ont qu’une seule chose en commun , une seule chose : Aucune n’ait t lance par une seule personne .”

[6]Due to time constraint and the relatively big size of our test

Model	Attention	Learning Rate	Hidden Size	Beam Search
bpe2bpe	dot	2.3e-4	738	15
bpe2bpe	concat	6.2e-4	786	9
bpe2char[6]	dot	2.3e-4	1024	15
bpe2char	concat	3.2e-4	512	8

Table 1: Hyper Parameters for best models.

Model	Test 2012	Test 2013	Test 2014
bpe2bpe	16.87	15.02	14.12
bpe2char	10.33	9.30	10.61

Table 2: Test Bleu scores for Luong Dot-Attention.

Model	Test 2012	Test 2013	Test 2014
bpe2bpe	9.97	8.75	8.32
bpe2char	3.4	2.38	2.55

Table 3: Test Bleu scores for Concat-Attention.

5 Conclusion

Our results enforce our assumption that combining BPE and character methods with dot-attention generalizes the results of (Luong et al., 2015) to the WIT^3 dataset, where dot-attention outperforms our implementation of concat-attention. We further see that BPE encoding and decoding proves an optimal method for the NMT task given the time constraints of a term-project.

We further see that character level decoder models on the WIT^3 dataset are able to produce good translations, but take additional time and resources to reach their optimal weights. Finally, given our issues with the implementation of concat-attention, we note that it is critical to explicitly review the implementations provided in tutorials for conceptual mistakes in addition to runtime errors. NMT models may correctly generate translations for short sentences, even with broken attention mechanisms.

Contribution Statement

Eduardo and Akash worked on training and evaluating bpe2pbe models for each type of attention. Millie and Shasha worked on training and evaluating bpe2char models for each type of attention. We jointly contributed to evaluation, framework debugging, and paper writing.

sets, this model was evaluated using a random subset of the each test set, rather than using the entire sets.

Acknowledgments

Our main code was built upon resources found in the Practical Pytorch github tutorial (Robertson, 2017). To build the byte-pair encoding, we followed the procedure and code described by (Sennrich et al., 2016b)[7]. The tokenization and text normalization strategy for both, BPE and character encoding followed the code from Lee et.al. (Lee et al., 2016)[8]. The BLEU scores were estimated using Moses Statistical Machine Translation System `multi-bleu.perl`[9] This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2018. Neural versus phrase-based mt quality: An in-depth analysis on english–german and english–french. *Computer Speech & Language*, 49(Supplement C):52 – 70.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May.

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. *CoRR*, abs/1511.07916.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *CoRR*, abs/1603.06147.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *CoRR*, abs/1610.03017.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-

[7]Taken from: https://github.com/rsennrich/subword-nmt

[8]Available at: https://github.com/nyu-dl/dl4mt-c2c

[9]Available at: https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Sean Robertson. 2017. Practical pytorch: https://github.com/spro/practical-pytorch/blob/master/seq2seq-translation/seq2seq-translation-batched.ipynb.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Edinburgh neural machine translation systems for WMT 16. *CoRR*, abs/1606.02891.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Appendix A: Concat Attention Implementation Issues

As mentioned in our Results and Analysis section, we discovered issues with our implementation of the concatenation method of attention. These were discovered too late for a correction to impact the models represented here, which is why our main comparison is between character and BPE encoding for Luong (dot) attention rather than across both dot- and concat-attention as we originally intended. While investigating this issue, we found that the key difference was between the code base implementation we used as a reference and the actual Luong attention paper:

$$score(h_t, \bar{h}_s) = \begin{cases} h_t^\top \bar{h}_s & dot \\ h_t^\top \mathbf{W}_a \bar{h}_s & general \\ v_a^\top \mathbf{W}_a [h_t; \bar{h}_s] & concat \end{cases}$$

(Robertson, 2017)

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & dot \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & general \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & concat \end{cases}$$

(Luong et al., 2015)

Figure 3: Comparison of "Practical Pytorch" version of equations with original paper. The first image is a screen shot of the equations from the tutorial, while the second is a screen shot of the equations from the Luong paper. Both images were captured December 17, 2017.

These equations are identical except for a key element: the tutorial equations are missing a tanh nonlinearity in the concat-attention function, which is an essential element of the Luong implementation of the concat function. We did not catch this issue until the last-minute for a few reasons. First, the translations output by the concat attention models seemed relatively appropriate for the input sentences - for both BPE and character level target sentences, models with this method of attention produced distinct words with meanings close to those of the original sentence. Examples may be seen in the last section of the appendix. The validation Bleu scores for these models were additionally similar to the early stages of the dot-attention models.

We further evaluated our concat-attention based on local runs of visualizations. However, with this approach we neglected to realize that we were using shortened sentences for CPU computation, which ultimately would impact our overall results.

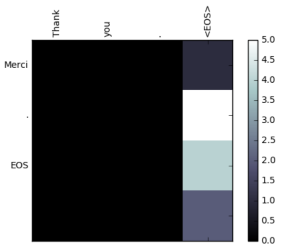

Figure 4: Local attention results

Our error became clear once we began generating visualizations of our final concat-attention models, which can be seen in the final section of this appendix (Figures 7, 8, 9, and 10). It is clear from these visualizations that the attention is not learning as it should – weights are focused entirely on a single word of the input sentence.

Appendix B: Character Level dot-attention visualizations

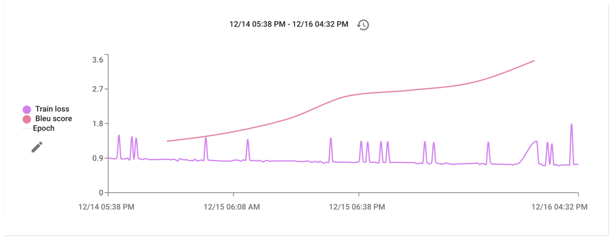

Figure 5: Learning Curve for bpe2char model with Dot Attention

[10]the spikes in training loss are iterations where teacher forcing was not enforced.

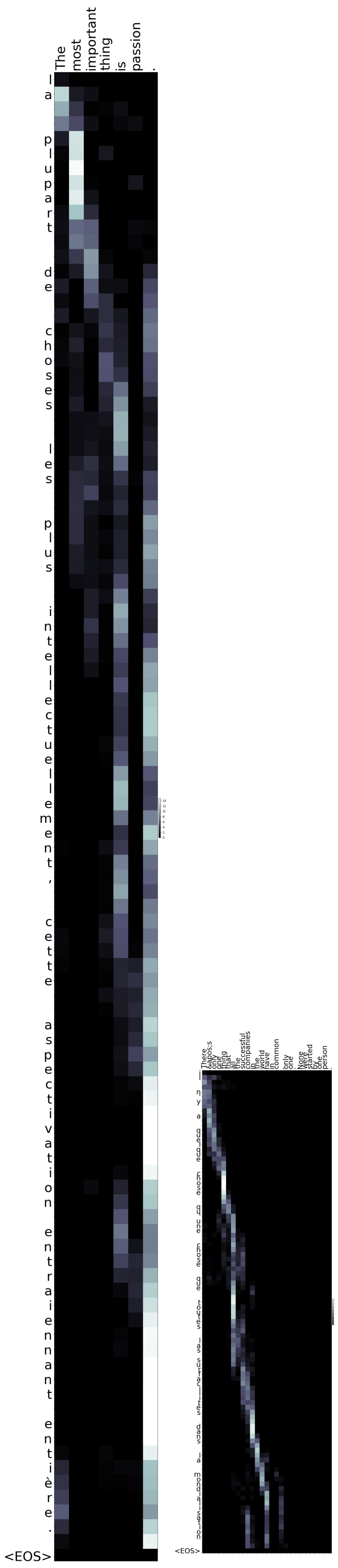

Figure 6: Figures for bpe2char model with Dot attention, one simple & one hard sentence

Appendix C: Dataset Analytics

Table 4 shows some basic statistics about the train, validation, development, and test sets for each of the encoding methods. Within a method and a language, there is no significant difference in the distributions of each of these sets.

file	avg_char_length	max_char_length	min_char_length	std_char_length	avg_word_length	max_word_length	min_word_length	std_word_length
train.fr	109.565428	3800.0	4.0	82.874692	21.349565	921.0	3.0	15.815075
train.en	95.759618	2688.0	2.0	71.769287	17.358149	624.0	1.0	13.252060
dev.fr	114.475761	868.0	6.0	87.313087	19.143179	153.0	1.0	14.503695
dev.en	108.768884	874.0	5.0	82.092103	19.677565	158.0	1.0	14.720997
tst2012.fr	106.830961	511.0	5.0	71.674791	18.846975	92.0	2.0	12.157174
tst2012.en	92.457295	448.0	5.0	60.573335	17.556940	85.0	2.0	10.852266
valid.fr	108.532889	2199.0	4.0	82.515385	18.373518	473.0	1.0	14.449142
valid.en	97.359470	1894.0	4.0	73.979112	17.595764	439.0	1.0	13.773840
tst2013.fr	111.419103	680.0	4.0	80.858381	19.053606	110.0	1.0	13.645185
tst2013.en	99.222222	593.0	8.0	71.918864	18.041910	100.0	1.0	12.834727
tst2014.fr	103.857471	552.0	5.0	71.023905	17.699617	91.0	1.0	12.029091
tst2014.en	90.065134	472.0	6.0	62.349229	16.235249	83.0	1.0	11.127432
train.fr.tok.bpe	120.582983	4299.0	2.0	92.059063	22.965204	1006.0	1.0	17.623550
train.en.tok.bpe	105.032902	3053.0	2.0	78.006702	21.400683	753.0	1.0	15.812163
dev.fr.tok.bpe	130.624577	990.0	7.0	99.963973	24.693348	191.0	2.0	18.563356
dev.en.tok.bpe	119.269448	954.0	7.0	89.901829	24.144307	190.0	2.0	17.779798
tst2012.fr.tok.bpe	118.790036	598.0	5.0	80.641966	22.109431	115.0	2.0	14.745273
tst2012.en.tok.bpe	99.846085	497.0	5.0	66.070182	20.056940	102.0	2.0	12.936003
valid.fr.tok.bpe	123.319639	2623.0	4.0	93.305690	23.435472	604.0	1.0	17.962211
valid.en.tok.bpe	107.407541	2185.0	5.0	80.976998	21.895240	553.0	1.0	16.614054
tst2013.fr.tok.bpe	127.484405	817.0	5.0	92.221375	24.477583	150.0	2.0	17.369507
tst2013.en.tok.bpe	109.678363	687.0	9.0	79.248689	22.561404	134.0	2.0	15.795187
tst2014.fr.tok.bpe	119.396169	695.0	6.0	81.707469	22.724904	123.0	2.0	15.221940
tst2014.en.tok.bpe	100.225287	614.0	7.0	69.173754	20.414559	116.0	2.0	13.676019
train.fr.norm.char	215.320781	7445.0	7.0	162.435875	127.114409	4566.0	6.0	95.179578
dev.fr.norm.char	227.816234	1735.0	11.0	174.584219	132.521984	1020.0	6.0	101.662191
tst2012.fr.norm.char	212.572954	1019.0	9.0	143.252393	124.589858	600.0	6.0	83.586233
valid.fr.norm.char	215.930302	4397.0	7.0	164.939741	125.777022	2653.0	4.0	96.674857
tst2013.fr.norm.char	221.666667	1359.0	7.0	161.648501	129.328460	789.0	4.0	94.284100
tst2014.fr.norm.char	206.526437	1103.0	9.0	141.981444	120.410728	642.0	5.0	82.825394

Table 4: Dataset Sizes.

Dataset	**Original Pairs**	**Trimmed Pairs**	**% of Total**
Training	178,046	104,875	58.9%
Validation	887	494	55.7%

Table 5: Impact of Trimming Dataset based on maximal character length.

Appendix D: Attention Visualizations for Concat-Attention

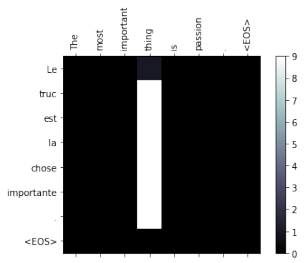

Figure 7: Concat-attention (Bahdanau based) for bpe2bpe model Example 1

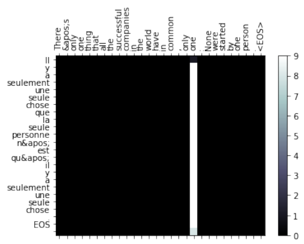

Figure 8: Concat-attention (Bahdanau based) for bpe2bpe model Example 2

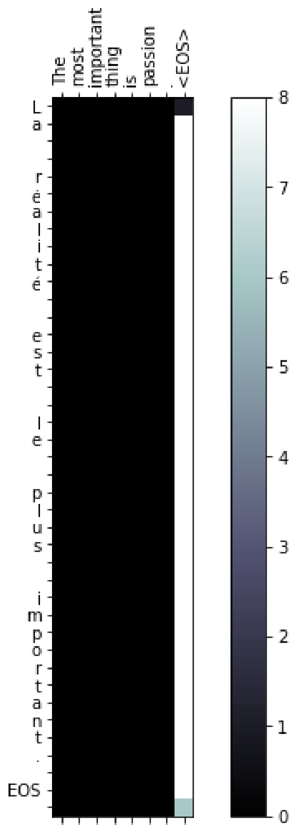

Figure 9: Concat-attention (Bahdanau based) for bpe2char model Example 1

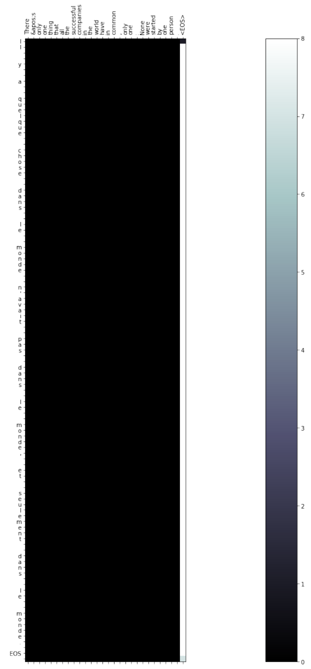

Figure 10: Concat-attention (Bahdanau based) for bpe2char model Example 2