Experimenting with character-level translations Hungarian-English

Abstract

The Hungarian language presents notable differences, compared to the English language, that deem translating between the two a difficult task. In this report we describe the experiments we conducted using the Open-Sub dataset, while training a neural translation model to translate Hungarian character sequences to English. We explain why the task was difficult and what the experiments aimed at achieving. We define in detail the structural characteristics of the model we used. Finally, we analyze and interpret its results.

1 Introduction

1.1 The Hungarian language

The Hungarian language belongs to the Uralic languages, like Esthonian and Finnish. As such, it is quite different from other Indo-European languages, like English. It is a morphologically agglutinative language. This means it is a synthetic language, with words that may contain different morphemes to define their meaning, but, these morphemes are not altered after their union. Due to this, the meaning is easily deciphered. Nevertheless, Hungarian words can be compounds as well as derived, mainly, from suffixes, which leads to another characteristic: the Hungarian uses vowel harmony. The various suffixes have more than one form and their use depends on the vowel of the head word. For instance, "lát" (English: "see") will become "látok" ("I see"), but "szeret" ("love/like") becomes "szeretek" ("I like"). Note that the corresponging for "I cannot see" in Hungarian is "nemlátok", a synthetic word deriving by adding the prefix "nem" which is "no". Moreover, the Hungarian is a topic-prominent language. That is, the word order depends on the topic-comment structure of the sentence, specifically, the word order typically is: topic-comment, verb and other parts. Table 1 contains more Hungarian language characteristics, taken from the World Atlas of Language Structures Dryer and Haspelmath [2013].

1.2 Translating to English

Some of the characteristics we mention above deem the translation task difficult and complicated. In Table 1 we provide characteristics where Hungarian disagree with English, according to the WALS. It can be seen that the two languages have significant differences that can easily affect the translations. For example, whereas in the English language there are semantic gender assignment rules, the Hungarian have none. Another example is the lack of Perfect in Hungarian, which is expressed periphrastically in English. Such differences are expected to create 'gaps' in the translations. More examples of differences between the two languages are presented in Table 1. Furthermore, the dataset we use consists of translated film-subtitles. This alone creates discrepancies. Subtitle translations are usually adjusted, so that they are fluent but not necessarily close to the source sentences; in other words, in subtitles it is the context that is translated. For instance, the Hungarian phrase "Hát ne agyalj" is translated as "Blow your mind", but "Hát" in English means "back" and "agyalj" "invent". A translation model learning from these sentences will not even link "ne" to "no".

1.3 Research Question

This project's main goal is to produce translations from Hungarian to English, using the Open-Sub dataset and a basic encoder-decoder model neural translation model. Taking into account the morphological, grammatical and syntactical differences between the two languages, we chose to experiment with character-level translations. In particular, we examined how the encoder-decoder model can perform when the inputs are the character sequences and not the words. Our choice was made because character-level models are more flexible than phrase-based ones. Phrase-based models, for example, will not perform well when new, not-in-the-vocabulary words are introduced. Apart from this, in character-level models there is no need for explicit segmentation and they can possibly model more morphological variants. The rest of this report is organized as follows: In Section 2 we describe the translation model we used, with details of its architecture and parameters and explain the evaluation measures we considered. In Section 3 we present and analyze the results of the experiments.

In Section 4 we provide relevant work on the subject and conclude this report with possible experimental extensions. We include in the Appendix all the Tables and the Figures we refer to.

2 Methods and Measures

For our experiments we used an attentional RNN encoder-decoder model, similar to the one from coursework 2, for all the experiments. In this section we present the model's architecture and characteristics and describe how is is implemented using the Chainer library. We also explain how the results of our experiments will be evaluated and the measures we focused on.

2.1 Model Architecture

A neural translation model models the probability $p(y|x)$ of translating a source sentence x to a target sequence y. In the Encoder-Decoder model, first the encoder reads an input sentence, which is a sequence of vectors $x = (x_1, ..., x_n)$, where n corresponds to the number of words in an input sentence. For the encoder we used recurrent neural networks, that can hold information of the sequence in their states. More specifically we used LSTM's.

Before passing the input sentence to the encoder's LSTM, we create an embedding layer (one-hot encoding). The encoder LSTM, then, performs two reads of the sequence, one from left to right as ordered and calculates its forward hidden states, another from right to left and calculates is backward hidden states. We concatenate the forward and backward states, and this sequence is passed to the decoder. The decoder is trained to predict (translate) the next word y_t, given the context vector and the previously predicted words $y_1, ..., y_{t-1}$. In other words, for a sequence vector $\mathbf{y}$, the decoder is trained to define the probability $p(y) = \prod_{t=1}^{T} p(y_t|y_1, ..., y_{t-1}, c)$, where c is the context vector. For the decoder we also used LSTM's, that we initialize on the last encoder LSTM's calculated states. The input is again one-hot encoded.

We differ from the provided model in the implementation of attention. For our project we chose to implement local attention as described by Luong et al. [2015], because of the way we processed our data, which we will describe in Section 3.1. The attention layer is implemented after the encoder and takes as input the hidden states of the encoder's LSTM. We derive the context vector c_t as described, aiming to gain information on the source-side. The context vector is concatenated with the hidden states vector and passed through the softmax function to predict the next word y_t with probability $p(y_t|y_1, ..., y_{t-1}, c_t)$. To compute this context vector we first calculate the aligned position $p_t = t$, assuming the source and target sequences are monotonically aligned (local-m). The context vector is then computed as a weighted average over a small window around the position p_t. The weights (alignment vector) a_t are calculated similarly to the global attention implementation, by comparing the source hidden states h_s in the window and the current target's hidden states h_t.

2.2 Chainer Implementation

To implement the translation model we used the Chainer framework. To create a neural network in Chainer, we use links, that are objects that hold parameters and can, therefore, execute different functions. The basic linear `link links.Linear()` , for example, represents the mathematical operation for linear regression. We will now describe the links we used for this model's architecture.

We use `links.EmbedID()` to create the embedding layer, which will create the one-hot encoding of a sentence. Then, we add the encoder's LSTM layers with `links.LSTM()`, first for the forward read, and then for the backward read. We use the same link types for the decoder, and embedding layer followed by the LSTM layers. After each LSTM, we use `functions.dropout()` to apply dropout, that is some connections of the previous layer are skipped during training, with probability 0.2. For the decoder we use an embedding link followed by the LSTM ones. The attentional layer is a `links.Linear()` link, since, as described above, it requires concatenations and a `functions.softmax()` computation. The final layer is the output layer, which is a linear link as well. The output is then passed through the softmax function to give the probability distribution and predict the next word.

Links in Chainer hold parameters, which are Variables and can keep gradient information. Thus, we process the input data by storing it in Variables. Chainer can make use of these parameters for the forward and backward propagations required to run this model. For the training on all the experiments, we use the default `Adam` optimizer, with $a = 0.001$. We also used the `Gradient Clipping` hook function, as in the baseline model, for re-scaling all the gradients within the threshold (5).

2.3 Evaluation Measures

In order to evaluate the results of the experiments we compute the BLEU score and the perplexity of the translations on the validation set. We cannot compare to the results of the provided baseline, since we have

altered the data (Section 3.1) and the predicted output is different at each time step than that of the baseline. Nevertheless, we graphically represent the perplexity and BLEU score during the training. Moreover, we provide examples of the output translations and compare to the target ones, to check the model's performance.

3 Results and Analysis

In this section we first describe how are data is altered, to serve our purpose. Then we present and justify the experiments we conducted. Finally, we analyze their results.

3.1 Processing the Data

The question we focused on is how a character-level translation will affect the performance of the described encoder-decoder model. In order to do that, we decided to process the data accordingly. The dataset we chose is the provided Open-Sub dataset, with subtitles of various films. We altered both the Hungarian and the English files, by inputting an empty character between the word letters. The model was then trained using these files. Of course, we re-created the vocabulary and dictionaries, since we did not use the baseline files. We chose to implement local attention, because the authors in Luong et al. [2015] note that it was proven better for longer sequences. Indeed, in the processed data we used, the sequences were significantly long, and the global attentional representation would grow quadratically with the sentence length. In Table 2 we include information of both the original dataset and our processed one, after using the basic tokenization function. We note that the vocabulary size for the data we used is significantly small. In Figure 1 we present a graphical representation of the relation between the original Hungarian and English sentences.

3.2 Experiments

We conducted the initial set of experiments using the first 50000 sentences in the files. The sentences were divided, 90% for the training and 10% for the validation set. This results in 45000 training sentences and 5000 validation ones. We trained the model with 3 LSTM layer for the encoder (3 forward and 3 backward read, total of 6 LSTMs) and 3 LSTM layer for the decoder, for 60 epochs. The BLEU score and perplexity, as computed after each epoch on 500 validation sentences, are plotted in Figure 2. The plots showed signs of convergence after the 30^{th} epoch. The translations, though, were bad. That is, they were not comprehensible or fluent. Some examples are accumulated in Table 3. Given this, we run the same model, using 1 layer for both the encoder (2 total) and 1 for the decoder, for 20 epochs (due to time restrictions), to see how the translations will be affected. The new plots are shown in Figure 3 and the accumulated measure scores in Table 4.

The second set of experiments was conducted using the whole dataset of 100000 sentences. The sentences were again divided, 90000 for the training and 10000 for the validation set. We again trained a model with 1 layer used in both encoder and decoder (2 total in the encoder), for 20 epochs, because of the time the model required to train and to compare with the previous results. The new plots are also shown in Figure 3 and the final values in Table 4. We also provide some translated examples in Table 3. Since time allowed, we finally let the model train using 3 layers for the encoder (6 total) and 3 for the decoder. The results are included in Figure 2 and Table 4.

3.3 Results Analysis

When we initially examined the results, they were disappointing. The model had not really learned to translate sequences of characters. We should note here that we did not expect the model to produce as good translation as the baseline. On one hand, they are character translations and are bound to not be fluent or even readable. On the other hand, there are many complicated sequences of characters that could be translated in different ways for reasons explained in Section 1.2. Hence, our expectations were low. A closer look, though, provided some insight. For example, it was interesting to discover that the translation to sentence 5196 "e l n é z é s t " was "i x c u s e m e y o r r r r e r _EOS " and came pretty close to the original translation , which was "e x c u s e m e ". We searched for more such examples and again for sentence 45001 "c e s c a" was translated as "c e s c a c a c c a c n e e e s s _EOS" was close to the target sentence, which was "c e s c a c e s c a". It also identified short words, like "nem", or "de." We include some interesting translated examples in Table 5. The actual BLEU score and perplexity values' progress show that the model indeed learned. In Table 4, the model that trained for 50000 and 3 LSTM layers on encoder-decoder gave the best results. This was expected, since the data was half what was used for training the 100000 models, and for more epochs. Also, more layers improve the final values when comparing to the 1-1 model, but the time required for the training was almost double. As we mention above, we cannot compare to the baseline. The 3-3 layered 50000 sentences(of characters) model achieved a BLEU score of 7.45 and perplexity 223.099 on the validation set, after 60 epochs. The translations of the whole file are given in the `translations.txt` file.

4 Conclusion

In this report we presented the experiments we conducted using the Open-Sub dataset, to establish how an attentional RNN encoder-decoder neural translation model performs when the input sequence consists of characters. We described the model we used and the experiment details. We provided the results of the training process and some examples of the output, noting on what we found interesting. We will conclude this report with some relevant projects on the subject of character neural machine translation and extensions that can be implemented.

4.1 Relevant Work

According to the Machine Translation Archive(http://www.mt-archive.info/), there has been little research on translating Hungarian, compared to other languages, like the Japanese, Spanish or Russian. In fact, prior to the 21^{st} century, there were almost no attempts at all.

On the topic of character-based translations, in neural machine translation, a very popular model is the attentional RNN encoder-decoder, as describe the authors for example in Luong et al. [2015]. Some implementations use LSTM, as do Bahdanau et al. [2014], while others use stateful GRU's, like the model with bidirectional GRU by Lee et al. [2016]. These models, though, are phrase-based. Recently, there has been a lot of research in character-based models and whether they can provide better results. An example is presented by Ling et al. [2015]. Of course, these models are not simply preprocessing the data accordingly, like we did. Instead, they use the information from a character-level process of the input sentence, as well as the alignment information of words. For instance, Luong and Manning [2016] propose a hybrid NMT system. For the character-level process, convolutional neural networks are introduced. Usually, in these translation models CNN's are paired with Highway networks. CNN's are more commonly used in image recognition, whereas RNN's are preferred for machine learning tasks that require noting the sequence, like machine translation. Nevertheless, using CNN's for the character-level encoding, prior to the RNN encoder, has in some cases produced better results than the basic RNN encoder, as for example point out the authors in .

4.2 Possible Extensions

Considering the above, it would be interesting to utilize our model's results within a phrase-based model. We could possibly apply a 1-dimensional convolution in the input sentence, to reduce its length. Then, pass the output to the standard attentional encoder-decoder model and observe how it performs. Another, different, approach is to preprocess the Hungarian and English files more, in a smart way that utilizes the individual language phenomena, such as those in Table 1. For example, we can break down the Hungarian synthetic words to their morphemes, or even remove the suffixes, and note how this affects the translations on a phrase-based model, as we speculate that the phrase-based model will be affected more than a character-based one.

References

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

Matthew S. Dryer and Martin Haspelmath, editors. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig, 2013. URL http://wals.info/.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*, 2016.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

Minh-Thang Luong and Christopher D Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*, 2016.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Appendix

Language Feature	Hungarian	English
Number of Genders	*None*	*Three*
Object-Verb order	*VO*	*OV*
Relative clause-Noun order	*Mixed*	*Relative clause-Noun*
Reduplication	*Productive full and partial reduplication*	*No productive reduplication*
Perfect	*No Perfect*	*From possessive*
Morphological Imperative	*No second-person imperatives*	*Second person number-neutral*
Position of case affixes	*Case suffixes*	*No case suffixes*
Politeness Distinctions in Pronouns	*Multiple politeness distinctions*	*Pronouns avoided*
Definite Articles	*Definite word distinct from demonstrative*	*No definite article*
Numeral classifiers	*Optional*	*Obligatory*
Verbal Person/Number Marking Syncretism	*Not syncretic*	*Syncretic*
Case Syncretism	*No syncretism*	*Core cases only*

Table 1: Examples of World Atlas language phenomena for Hungarian and English

	Maximum sentence length	Number of tokens	Number of unique tokens
Open-Sub dataset			
Hungarian	491	252979	38896
English	527	362127	13987
Processed dataset, 50000 sentences			
Hungarian	491	1274773	75
English	527	1338774	76
Processed dataset, 100000 sentences			
Hungarian	569	2452428	78
English	727	2588812	83

Table 2: Open-Sub and character-processed dataset analysis

Number	Source sentence	Target sentence	Predicted Sentence
1-1 layer, 50000 sentences, 20 epochs			
45000	- r e n d b e n	- a l l r i g h t	- a l l r i g h t g g _EOS
45001	c e s c a	c e s c a c e s c a	c e s c a s c a c a _EOS
45002	n e m l á t o k	i c a n t s e e	i o e n t n t e e e e e _EOS
3-3 layers, 50000 sentences, 60 epochs			
45000	- r e n d b e n	- a l l r i g h t	- a l l r g g h t g h t h _EOS
45001	c e s c a	c e s c a c e s c a	c e s c a c a c c a c n e e e s s _EOS
45002	n e m l á t o k	i c a n t s e e	n o i d o n t s e e e e e e e e e e e e e e
1-1 layer, 100000 sentences, 20 epochs			
90000	b i z o n y o s ö s s z e g a s a j á t p é n z e m b ő l s z á r m a z o t t r e m é l e m n e m g o n d	t h i s f a m i l y h e l p e d m e i h o p e i r e p a i d m y d e b t	y o e e e e e e e e e e e e e e e e e e e e
90001	a h a v i 1 5 0 d o l l á r o s f i z e t é s e d b ő l	s o m e o f t h e m o n e y w e n t i n t o a p r o j e c t o f m y o w n	t h e e e e e e e t e t t t t t t e e 0
90002	n e m é p p	i h o p e y o u w o n t m i n d	i h e o n t t e e e e e e e e e e e e e e
3-3 layers, 100000 sentences, 20 epochs			
90000	b i z o n y o s ö s s z e g a s a j á t p é n z e m b ő l s z á r m a z o t t r e m é l e m n e m g o n d	t h i s f a m i l y h e l p e d m e i h o p e i r e p a i d m y d e b t	y o u e e e e e e e e e e e e e e e e e e e e
90001	a h a v i 1 5 0 d o l l á r o s f i z e t é s e d b ő l	s o m e o f t h e m o n e y w e n t i n t o a p r o j e c t o f m y o w n	a h e e o t t t t t t t t t t t t $ $ t t
90002	n e m é p p	i h o p e y o u w o n t m i n d	i o i o o t t t t t t t t t t t t t t t t

Table 3: Hungarian translations from our models

	BLEU	perplexity	precision	recall	f1
50000 sentences					
1-1 layer 20 epochs	6.63	31.0999	0.4409	0.2988	0.3562
3-3 layers 60 epochs	**7.45**	223.1049	0.2549	0.2616	0.2582
100000 sentences					
1-1 layer 20 epochs	3.78	117.4855	0.2541	0.2798	0.2663
3-3 layers 20 epochs	5.79	91.4701	0.2566	0.2840	0.2696

Table 4: BLEU score, perplexity of models trained

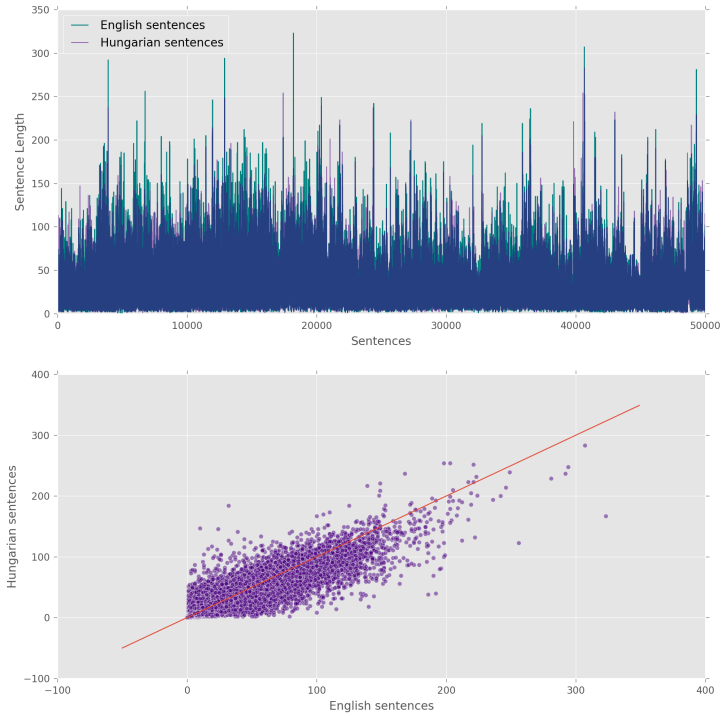

Figure 1: Hungarian and English sentence lengths, Open-Sub

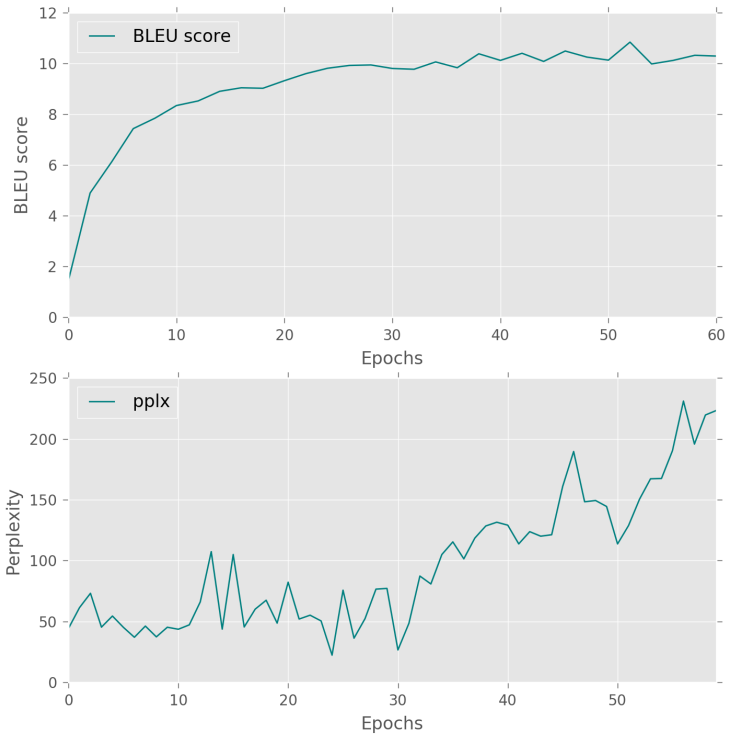

Figure 2: BLEU score and perplexity, 50000 Open-Sub sentences, 3-3 layers, 60 epochs

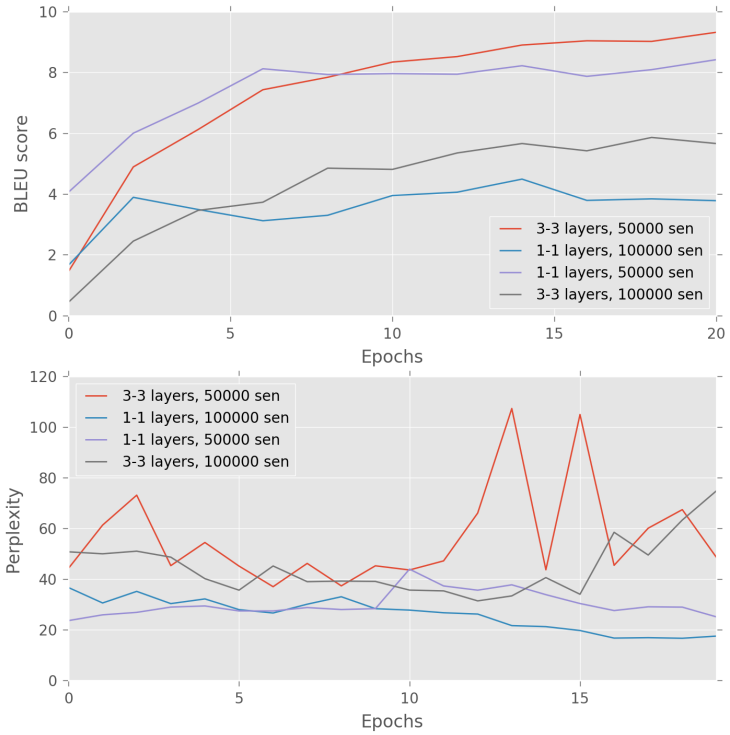

Figure 3: BLEU score and perplexity of models trained, 20 epochs

Number	Source sentence	Target sentence	Predicted Sentence
45000	- r e n d b e n	- a l l r i g h t	- a l l r i g h t g g _EOS
45013	n e m a k a r o k m e g h a l n i	i d o n t w a n t t o d i e	i d o n t w a n t t t t t t o o e e _EOS
45022	j o h n n y	j o h n n y	j o h n n y n y n y n _EOS
45023	m i t c s i n á l t á l	h e y w h a t d i d y o u d o	w h a t a r e y o u d d d d _EOS
45002	n e m l á t o k	i c a n t s e e	n o i d o n t s e e e e e e e e e e e e e e
45328	- n e m	- n o	- n o _EOS
45331	k i n g d o m	k i n g d o m	k i n g d o m d d i n e e e e _EOS
45361	Ó	u h	o h _EOS
45736	m i	w h a t	w h y _EOS
45747	d e	b u t	b u t _EOS
45763	m i k y e o n g - a h	m i k y e o n g	m i k y e o n g g o g g g g g g g g g g g
45773	h i á n y z o l	i m i s s y o u	i m i s s y o u s o o s s _EOS

Table 5: Notable predicted translations of the 3-3 50000 sentence model