Machine translation (from Russian language to Crimean Tatar language and vice versa)

Makhmudov Orkhan, Tsybrik Denis, Korzh Bogdan

May 2023

Abstract

The project report contains a description of a model's for machine translation from Crimean Tatar to Russian built with neural machine learning techniques. Collected and published first dataset for parallel corpus [crh] - [ru]. The data and the solution's code are distributed through GitHub in the following repository: `https://github.com/kut666/NLP_project`.

1 Introduction

Machine translation is one of the main tasks of natural language processing. Discussing the topic of the project, we decided that it would be cool to implement translators from Crimean Tatar into Russian and vice versa. Currently no machine translators for Crimean Tatar were published, but dictionaries and phrasebooks can be found. The translator we created in this project can be helpful for those who want to learn the language, talk to a native speakers and better understand Crimean Tatar language and culture.

1.1 Team

Orkhan Makhmudov collected data from phrasebook A.G. Goryanov, visualized comparison of models and word embedings.

Denis Tsybrik: collected data from laws, learned and validated GRU, GRU+Attention, LSTM+Attention -based models.

Korzh Bogdan rewrote the transliterator from latin to cyrillic, build a parser to collect more than 8000 translation pairs from news sites. Some of those websites were not translated to Russian, but had Ukrainian translation instead, so Yandex Cloud API was used to translate targets to Russian, a total of 2 million characters, learned and validated Transormer+Attention -based models.

2 Related Work

As we described earlier, there are no translators of this kind, so the examples below will be presented for similarly working translators in other languages. The principle of neural machine translation is clear, we need seq2seq models, encoders, an intermediate vector and decoders.

Encoder - It accepts a single element of the input sequence at each time step, process it, collects information for that element and propagates it forward.

Intermediate vector - This is the final internal state produced from the encoder part of the model. It contains information about the entire input sequence to help the decoder make accurate predictions.

Decoder - given the entire sentence, it predicts an output at each time step.

List of approaches to the task of neural machine translation that we used:

- GRU
- GRU+ Attention
- LSTM + Attention
- Seq2seq with transformer

3 Model Description

All the above described approaches were tried to solve this problem to find out which architecture will give us the highest BLEU score.

1) GRU
2) GRU + Attention
3) LSTM + Attention
4) Seq2seq with transformer

3.1 GRU and LSTM

The most common sequence-to-sequence (seq2seq) models are **encoder-decoder** models, which often use a **recurrent neural network** (RNN) to **encode** the source (input) sentence into a single vector. In this report, we'll refer to this single vector as a **context vector**. You can think of the context vector as being an abstract representation of the entire input sentence. This vector is then **decoded** by a second RNN which learns to output the target (output) sentence by generating it one word at a time.

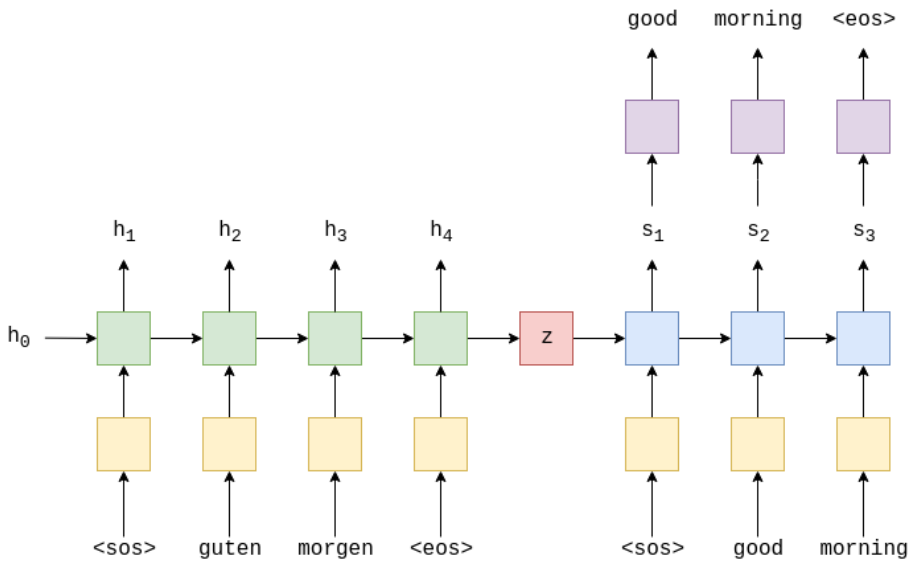

Figure 1: Seq2Seq model

The above image shows an example translation. The input/source sentence, "guten morgen", is input into the encoder (green) one word at a time. We also append a **start of sequence** (‘<sos>‘) and **end of sequence** (‘<eos>‘) token to the start and end of sentence, respectively. At each time-step, the input to the encoder RNN is both the current word, x_t, as well as the hidden state from the previous time-step, h_{t-1}, and the encoder RNN outputs a new hidden state h_t. You can think of the hidden state as a vector representation of the sentence so far. The RNN can be represented as a function of both of x_t and h_{t-1}:

$$h_t = \text{EncoderRNN}(x_t, h_{t-1})$$

We're using the term RNN generally here, it could be any recurrent architecture, such as an **LSTM** (Long Short-Term Memory) or a **GRU** (Gated Recurrent Unit).

Now we have our context vector, z, we can start decoding it to get the target sentence, "good morning". Again, we append start and end of sequence tokens to the target sentence. At each time-step, the input to the decoder RNN (blue) is the current word, y_t, as well as the hidden state from the previous time-step, s_{t-1}, where the initial decoder hidden state, s_0, is the context vector, $s_0 = z = h_T$, i.e. the initial decoder hidden state is the final encoder hidden state. Thus, similar to the encoder, we can represent the decoder as:

$$s_t = \text{DecoderRNN}(y_t, s_{t-1})$$

In approaches 1 - 3 we use 2 layer LSTM or GRU and attention mechanism for both LSTM and GRU.

3.2 Seq2seq with transformer

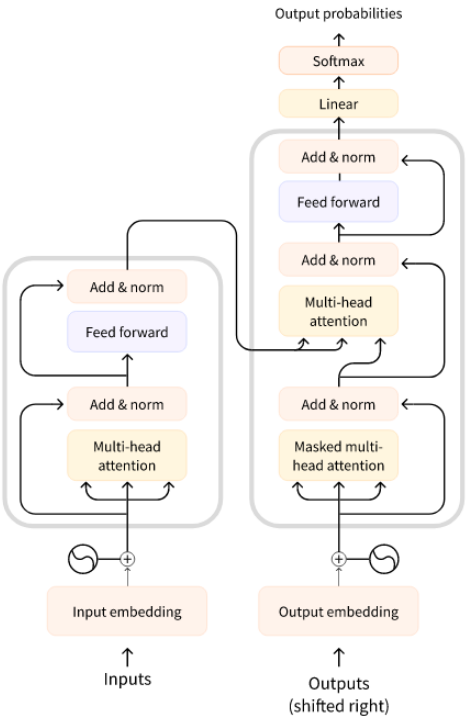

Figure 2: Original transformer architecture

The transformer architecture was originally designed for translation, this is why it fits so good for our case. It consists of an encoder and decoder built multi-head attention, normalization and feed-forward layers. In our case we also used Byte-pair encoder to encode target and source sequences.

Transformers have shown a very good BLEU score of 51 for Crimean tatar-to-Russian translation and 47 for the opposite direction (see the loss graph below).

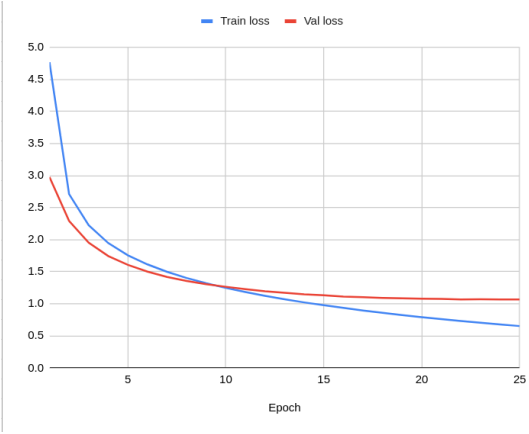

Figure 3: Training and validation loss for transformer-based NN

4 Dataset

	Train	Valid	Test
Sentence	23200	4350	1450
Vocabulary size (RU)		37,296	
Vocabulary size (CRH)		38,295	

Table 1: Statistics of the our dataset

Building the dataset was the most difficult part of this project. The dataset was compiled from three different sources: phrasebook, websites and laws. Let's analyze each of the sources in more detail.

Laws: In this part of working with data, we used written sources - the laws of the Republic of Crimea. Some kinds of laws are issued in the Republic of Crimea in three language versions: Russian, Ukrainian and Crimean Tatar

- Until 2014, the journal "Collection of normative legal Acts of the Autonomous Republic of Crimea" was published.`http://crimea.gov.ru/information/of_izdaniya/sbornik_npa_rk`,
- After 2014, the journal "Vedomosti of the state counsil of the Republic Of Crimea" is published.`http://www.crimea.gov.ru/information/of_izdaniya/vedomosti_gs_rk`

These journals contain a parallel corpus, which we was engaged in. There are was several steps in collection procedure:

1. Download documents from website

2. Convert documents from PDF to txt-file
3. Studying the structure of the document
4. Creating paragraph-pairs [crh]-[ru] from data
5. Selecting sentences from the paragraph
6. Preprocessing and combining datasets

Each of these steps is described in the corresponding project folder on Github Github.

PhraseBook: In this part of working with data, we used phrasebooks A. V. Goryainov (file in the repository) and Ablyaziz Veliev There are was several steps in collection procedure:

1. Download documents from website or from the author
2. Convert documents from PDF to excel file
3. Studying the structure of the document
4. Creating paragraph-pairs [crh]-[ru] from data
5. Selecting sentences from the paragraph
6. Preprocessing and combining datasets

5 Experiments

The model training was performed in Google Colaboratory with the Tesla T4 GPU and A100 GPU. The code was implemented in Python. The data from the dataset was shuffled and split into train (80% of the data), validation (15% of the data) and test (5% of the data) sets with dataset.split() method from the torchtext library. The PyTorch library was used to train the models.

5.1 Metrics

In our project we used BLEU metric to determine the quality of the translation.

BLEU formula:

$$BLEU_w(\hat{S}; S) := BP(\hat{S}; S) \cdot exp \sum_{n=1}^{\infty} w_n ln(p_n)(\hat{S}; S)$$

5.2 Experiment Setup

Now we know how to preprocess our input and output sentences into a NN-readable format. The last thing we need to do is to create a DataLoader for our data, which will take our sentences and put them together to form a batch. Problem here lies in the fact that sentences can have different sizes and items in one batch absolutely cannot. For this we use padding (remember the <pad> token). Back in version 0.8 torchtext used to provide its own custom classes which handled tokenization, <sos>, <eos> and <unk> tokens and added padding. However, since that time, torchtext ditched this functionality in order to make their dataloading API consistent with PyTorch's one. This in turn means that padding (as well as everything else mentioned) ourselves. Luckily, PyTorch's DataLoader supports custom collate functions, which seems like a great place to do all our preprocessing, including padding.

5.3 Baselines

Another important feature is that you could provide here the description of some simple approaches for your problem, like logistic regression over TF-IDF embedding for text classification. The baselines are needed is there is no previous art on the problem you are presenting.

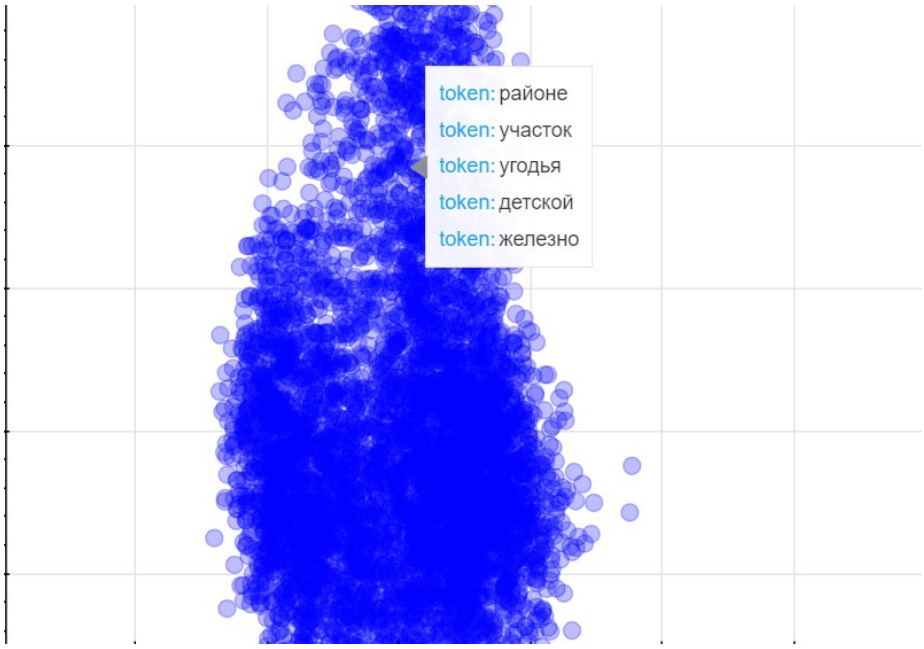

Figure 4: Word embedding visualization (RU)

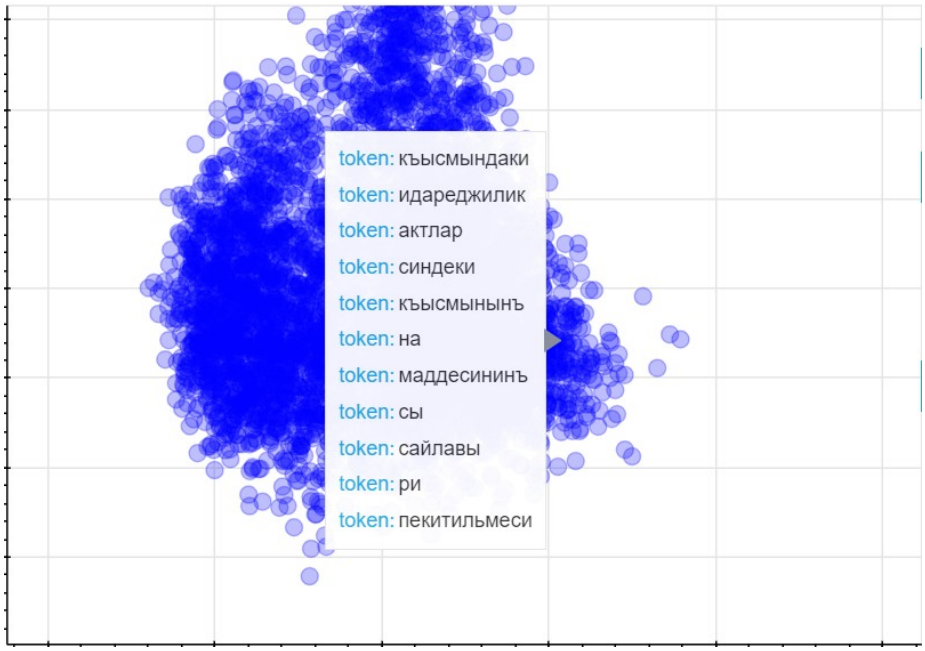

Figure 5: Word embedding visualization (CRH)

6 Results

Model	10 epochs	20 epochs	25 epochs
GRU	34.00	-	-
GRU + Attention	35.16	39.98	-
LSTM + Attention	28.39	-	-
Transformers + Attention	-	-	51.83

Table 2: BLUE-score over our dataset for different models

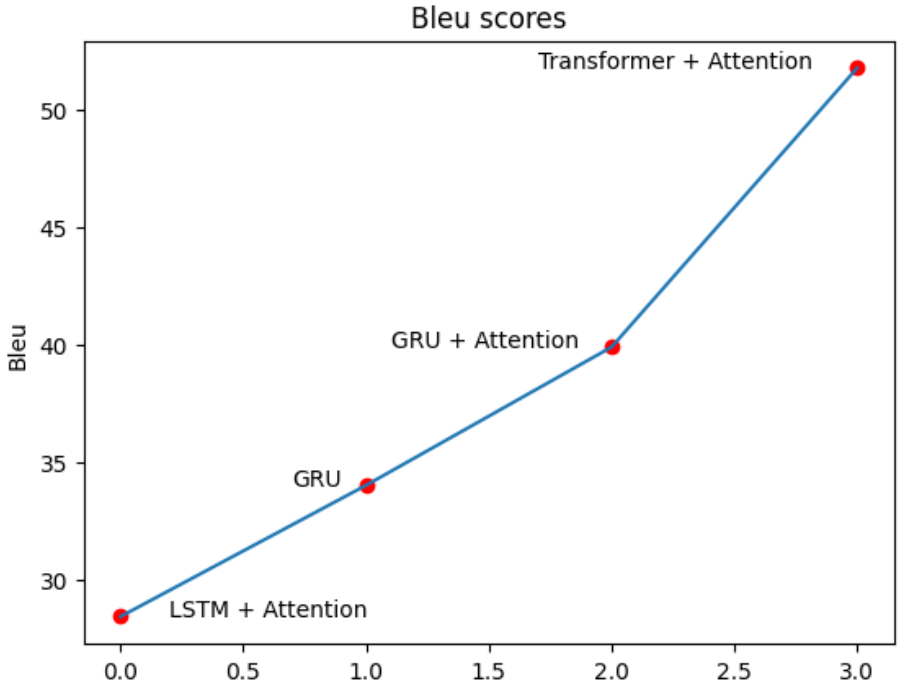

Figure 6: BLEU-score visualization for different model

Most models show decent BLEU-score, however transformers + attention showed the best score.

7 Conclusion

The first dataset of a pair of languages Crimean Tatar Russian was assembled. The maximum score of $BLUE = 51.83$ was achieved using the **Transformer**

crh: бу иляджны насыл ичмели $ru_{original}$: как принимать это лекарство $ru_{translated}$: как это лекарство пить
crh: бугунь акъшам мешгъульсинъизми $ru_{original}$: вы не заняты сегодня вечером $ru_{translated}$: сегодня вечером сегодня вечером

Table 3: Output samples for Transformer model

+ **Attention** model on 25 training epochs. This is the first work dedicated to machine translation from Crimean Tatar into Russian.