Embedding based neural machine translation from Finnish to Macedonian

Dejan Porjazovski, Olli Kaplas and Pellervo Ruponen

***Abstract*—In this work we present the method to perform translation from Finnish to Macedonian. We use a neural machine translation method: encoder-decoder model with a bidirectional Long Short Term memory network. The model incorporates word embeddings for both languages trained from scratch. We assess the neural trained embeddings against separately trained embeddings visually using dimensionality reduction technique t-SNE. For sentence level translation performance, the results are not quite reaching the state of the art models. Nevertheless, a good foundation for further improvements of translation task from Finnish to Macedonian was developed.**

I. Introduction

Machine translation is the process by which computer software is used to translate a text from one natural language to another. To process any translation, the meaning of a text in the original language must be completely reconstructed in the target language. This seems straightforward but it is complex than it looks.

The automatic translation is an important research topic since translation can be applied in multiple domains in order to save cost of human translator.

Historically, the machine translation was done with different techniques, such as: rule-based and statistical methods. The rule based machine translation system [11] was based on linguistic information about source and target languages basically retrieved from (unilingual, bilingual or multilingual) dictionaries and grammars covering the main semantic, morphological, and syntactic regularities of each language respectively. This approach requires extensive expertise in grammar, syntax and semantics of the languages.

The statistical machine translation [12] models are derived from statistical analysis of bilingual text corpora. They work in a way that a document is translated according to the probability distribution $p(e|f)$ that a string e in the target language is the translation of a string f in the source language.

With the rise of the amount of data and the huge increase of the computational power of the machines, new machine translation approaches have been developed. The neural machine translation is one of them [1], [13]. This approach eliminates the need of human feature engineering and relies on neural networks.

In this work we implement a neural machine translation from Finnish to Macedonian language by using Encoder-Decoder architecture.

II. Data

As data we used the OPUS corpus database [9]. The database contains large number of sentence pairs for multitude of languages. We extracted the Finnish-Macedonian sentence pairs to be used as our data.

The dataset is created from movie subtitles from the www.opensubtitles.org site. Movies the subtitles are a natural place to have parallel sentences in different languages so it's good source translation data. The dataset consists of approximately 2.3 million sentence pairs in both Finnish and Macedonian language.

A. Preprocessing of the data

The OPUS data is relatively high quality in comparison to the data that is usually provided by for example internet scrapers but still come preprocessing steps needed to be done.

In order to make the dataset smaller, we converted the sentences to lowercase. Another thing the we did was to remove the non-letter characters. We have also converted the unicode characters to ASCII. Another important thing that we did was to limit the length of the sentences to maximum 10 words.

After the preprocessing we had about 1.9 million sentence pairs. The vocabulary size for the Finnish language was 531K and for Macedonian language was 340K. The larger size of the Finnish vocabulary can be explained by Finnish noun cases which were processed as distinct words, since lemmatization nor stemming algorithms were not used in preprocessing.

At the end we have split the data into training and testing set, where the testing set consisted of 5% of the whole dataset.

III. Methods

A. Embedding

There are multiple ways to represent words as vectors. One of the simplest methods are the count based methods that are based on the co-occurrence counts of the words. The co-occurrence can be determined based on n-grams for example.

In our implementation, to embed words to vectors we used the well established word2vec method. This method is computationally efficient for learning dense vector representations of words and it has robust libraries for Python. In word2vec two training algorithms can be used: continuous bag of words (CBOW) or skip-gram. In our implementation we decided to use the CBOW algorithm and we used 300-dimensional space as a target space.

Separate vector spaces were trained for Macedonian and Finnish.

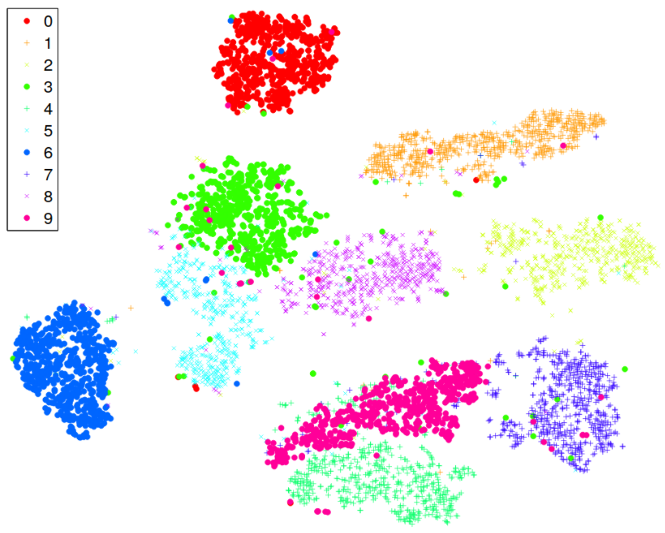


Figure 1. Visualization of MNIST data by t-SNE. [3]

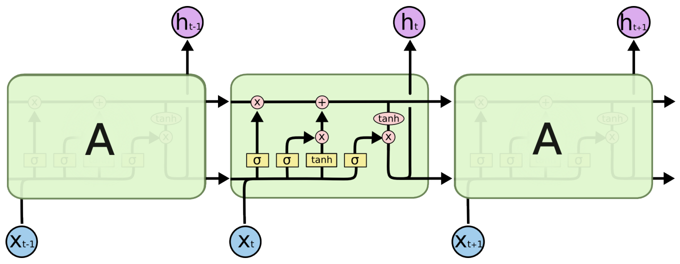


Figure 2. Graphical representation of three units of LSTM network. [5]

B. *Visualizing embeddings with t-SNE*

The embeddings are are representation of words in a high dimensional continuous space. There are many ways of visualizing embeddings with dimensionality reduction techniques. Usually this is done by reducing the dimensionality to two or three dimensions, which can be visualized in a graph. By far the best performing dimensionality reduction technique t-SNE was introduced by Van Haalten and Hinton [3]. By fitting a continuous t-distributions and minimizing their KL-divergence a representation is attained conserving much of the similarity of the data in the lower dimensional representation.

We use t-SNE to visualize and study the fitted embeddings. t-SNE is able to contain much from the local similarity as also the global structure such as clusters and scales (Figure 1).

C. *LSTM Recurren neural networks*

The translation is performed by the Long short-term memory cell (LSTM). LSTM is sophisticated architecture that is especially suitable for processing of the sequenced data by solving the problems that standard RNNs have, such as the vanishing and exploding gradient.

The LSTM cells used to be used in state-of-the art techniques in machine translation still in 2016 when Google published an article about their machine translation system

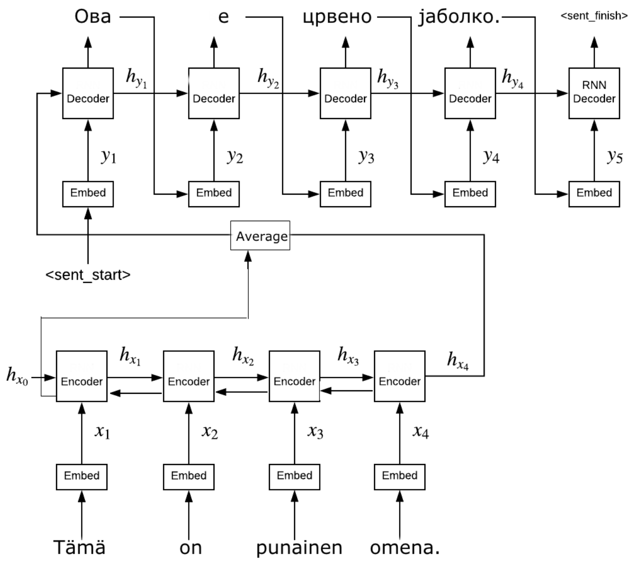


Figure 3. Structure of our encoder-decoder network.

[13]. Inspired by the architecture we went forward and implemented a bidirectional LSTM based encoder-decoder model. We considered to implement two layered version of this architecture but we ran out of time to train the model.

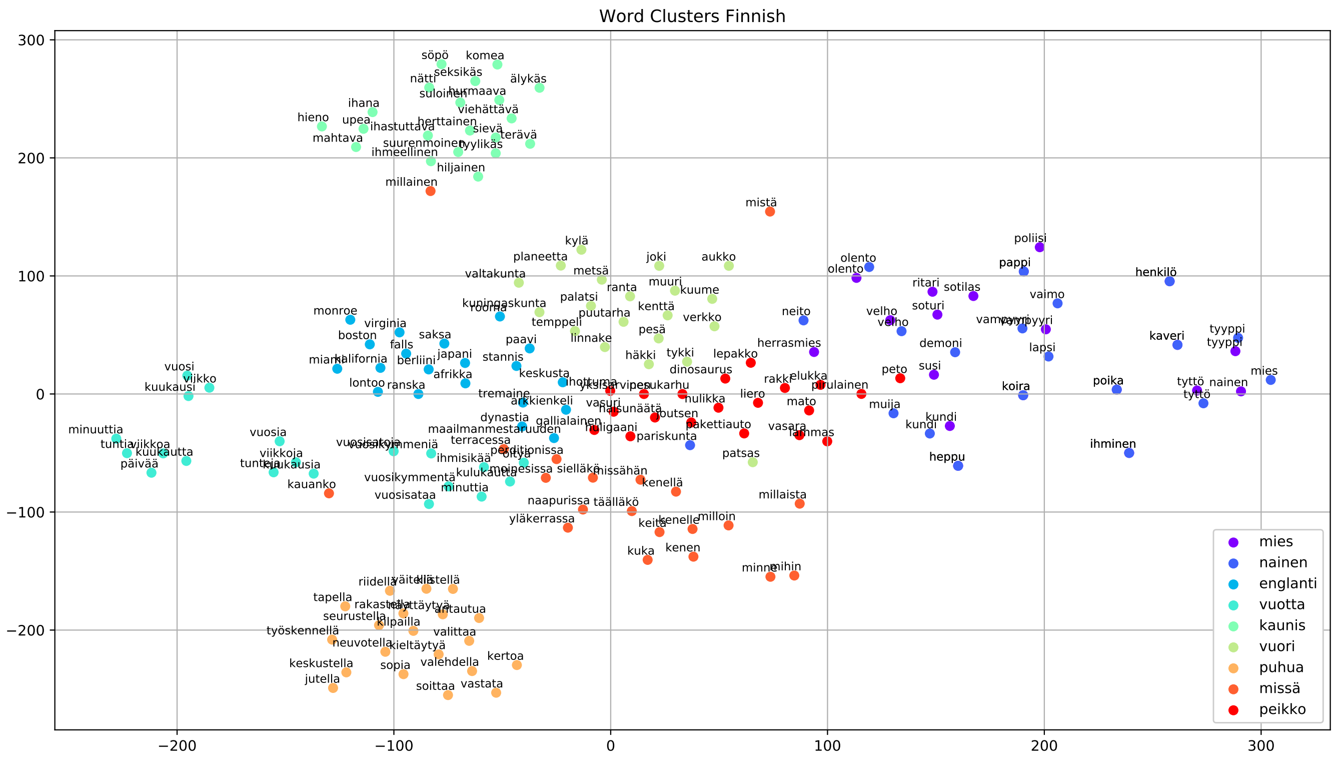
D. *Encoded-Decoder scheme*

In the Encoder-Decoder method, we trained two LSTM networks. The illustrative structure of architecture is shown in the Figure 3. The first LSTM is the encoder network, which processes the embedded word vectors of sentence in Finnish and produces a hidden state, which is a 256-dimensional vector that represents the sentence. In our architecture, the encoder works in bidirectional manner, whereas decoder works more traditionally only in the forward direction. This means that both directions of the encoder produce 256-length vector. These two vectors are averaged to produce a hidden state for a decoder LSTM. The decoder starts by taking in the hidden state produced by encoder. The decoder then produces a sequences of word vectors in Macedonian vector space. In the decoding, the hidden state is propagated through time and during each time step a word vector is generated. Half of the time, the previously outputted words are used as the previous state and the other half of the time, the true word is being used. This approach is called *teacher forcing*.

After the LSTM generates the output of the decoder, that output goes through a linear layer and the output of that linear layer is converted into probabilities for vocabulary using a log softmax function.

E. *Training*

Training of the model was performed on a GPU using PyTorch software library and Python. This gives us a robust environment and implementation of the training algorithms.



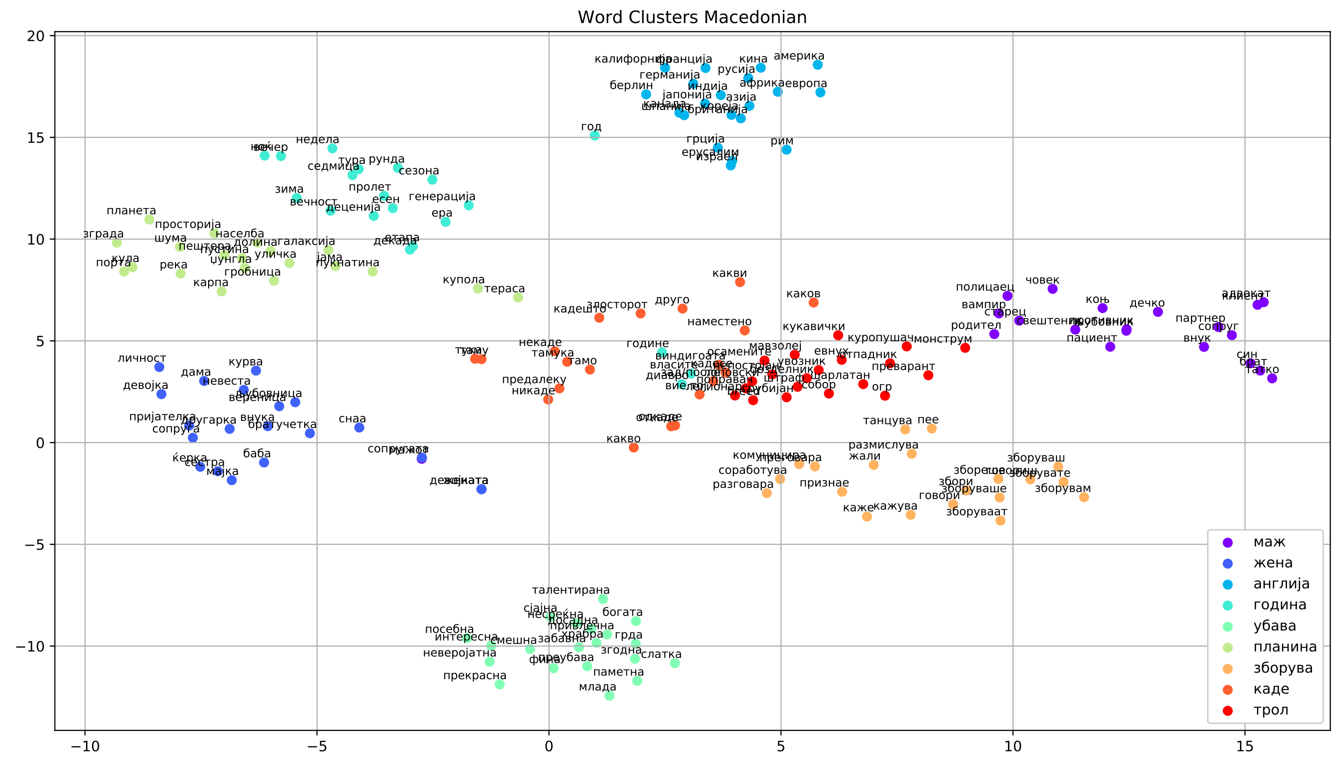
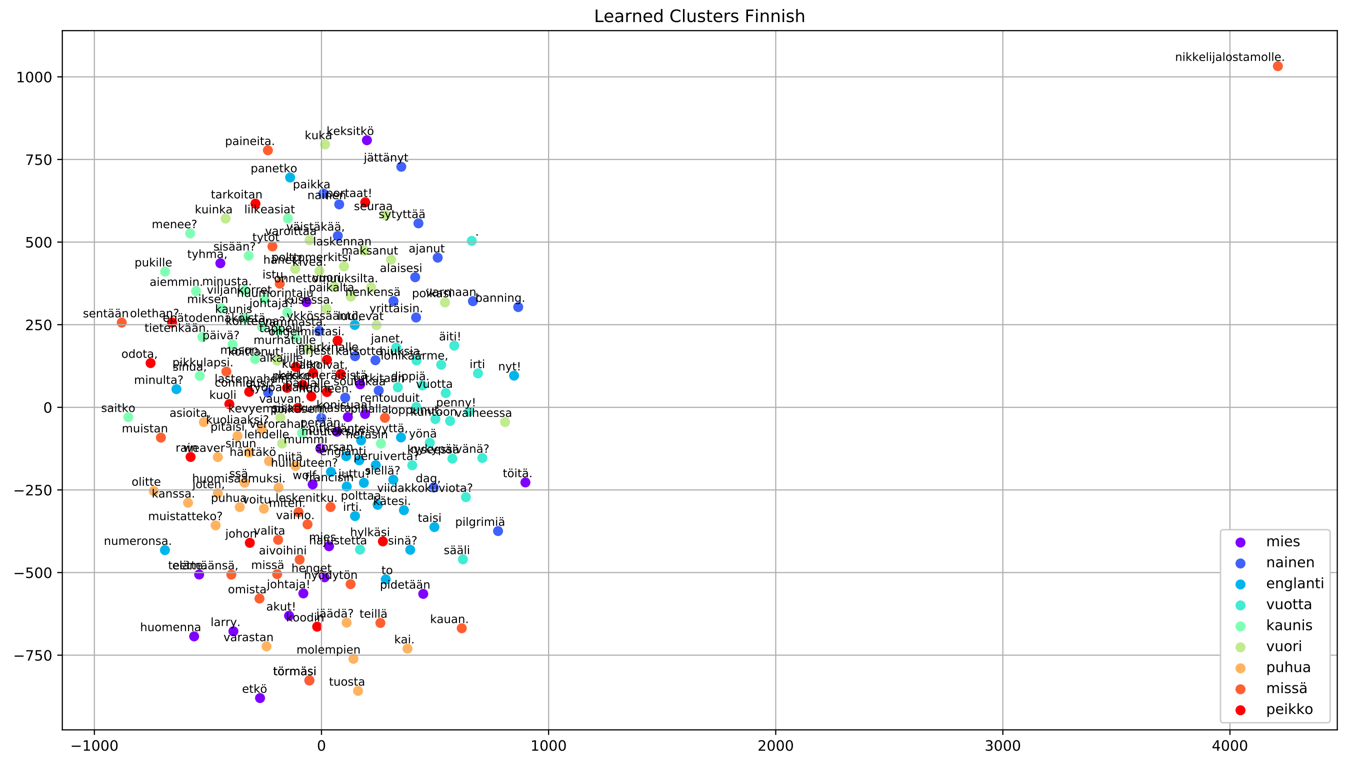


Figure 4. t-SNE for Finnish and Macedonian separately trained embeddings for selected words. 20 closest neighbours of each words are shown.



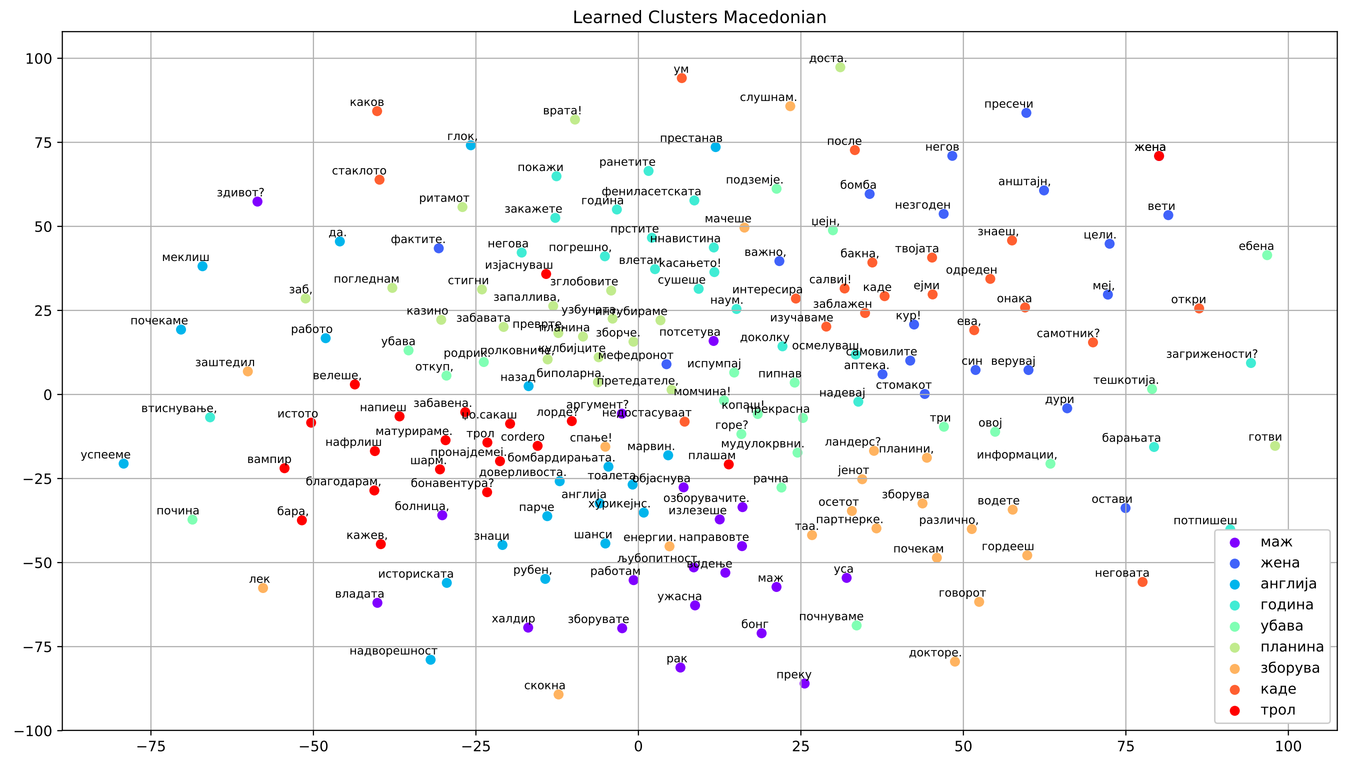


Figure 5. t-SNE for Finnish and Macedonian embeddings learned by the neural network during the training for selected words. 20 closest neighbours of each words are shown.

From here, we can see that the model did not learn the embeddings properly and it needs a lot more training.

The training parameters were as follows:

- Adam optimizer, learning rate 0.01
- Encoder: one layer bidirectional LSTM
- Decoder: one layer LSTM
- Dropout: 0.2
- Loss function: negative log likelihood
- Teacher forcing: 0.5
- Training size: 20000
- Batch size: 62
- Epochs: 30

Both the encoder and the decoder used Adam optimizer with a learning rate of 0.01. In order to prevent the overfitting of the model, we used a dropout rate of 0.2. For the translation tasks, the negative log likelihood loss is typically used, so we decided to use it as well. For the teacher forcing part, 0.5 is usually a good parameter so we decided to use that one. Even though we have more that a million sentence pairs in our dataset, we have decided to use only 20000 because of the hardware limitations and the time that it would take to train on the whole dataset. Also due to time limitations, we trained only for 30 epochs. With more epochs, the model should get better results.

F. Evaluation

To evaluate the translation we used the BLEU metric which is shown to correlate well with the human evaluation, despite of its computational simplicity [6]. The BLUE is defined as:

$$\text{BLEU} = \min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)\left(\prod_{i=1}^{n} \text{precision}_{\ i}\right)^{\frac{1}{n}}.$$

where first term is the penalty from too short translations and $\text{precision}_{\ i}$ is the precision when i-grams are matched between the sentences. In total, the matching is performed until n-grams.

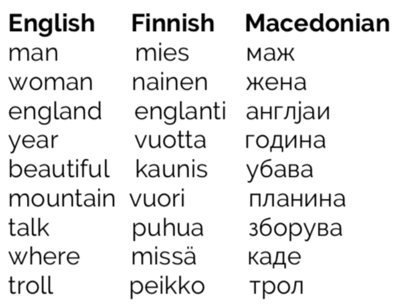
The BLEU usually performs better if there are multiple reference translations that can be used to compare the machine translation. Unfortunately, the dataset we used contained only one translation. Hence, the BLEU scores is not necessarily and optimal metric for our dataset.

IV. Results

A. Word embeddings trained separately with whole data

We trained both Finnish and Macedonian language word embeddings with the OPUS corpus data set of the respective languages. We chose 9 different types of words, with which we want to assess the goodness of the embeddings. We calculated the 20 nearest neighbours for each word and mapped them in to a t-SNE trained space. The t-SNE was trained with continuous bag of words method (CBOW), perplexity 20, initial start with PCA and 2000 iterations. The resulting graphs are depicted in Figure 5.

The words used in English/Finnish/Macedonian:



English	Finnish	Macedonian
man	mies	маж
woman	nainen	жена
england	englanti	англјаи
year	vuotta	година
beautiful	kaunis	убава
mountain	vuori	планина
talk	puhua	зборува
where	missä	каде
troll	peikko	трол

B. NMT-model: Encoder-Decoder model with bidirectional LSTM cell

C. Results for translation

The BLEU scores was calculated for 2-grams for the testing data and the score that we got is 0.0113. The way we did the calculation is that we summed the BLEU score for all the pairs in the testing data and the divided it by the number of testing pairs.

V. Conclusions

In this work we performed on sentence level translation from Finnish to Macedonian language. We got relatively good performance with the translation considering the fact that we used only limited data. The translations are not perfect, but still the network was able to model some structure of the translation. The testset BLEU score of 0.0113 tells same story: the translations are not optimal but not completely trash either. Most probably, by using more data in training, we could significantly improve the generalization of the model on the test set.

We identified several things that can be a cause for the inadequate performance of our model. First of all, the vocabulary is huge compared to the data that we used. Now we have only 20000 training examples and our vocabulary sizes are 500K and 340K. This means that our training set cannot even contain all words in vocabulary. This makes it really hard for the network to learn and is asking the network to generalize heavily with really small data.

The small training set also means that there are lot of new words in the testset, which also lower the performance. Of course, a good model should be able to generalize on new words, but if there is a lot of them, it will be more difficult. Other aspect that could cause decreased performance is the representational power of our model. In the implementation the sentence is described by 256-dimensional vector and LSTMs are 1-layer only. By increasing and tuning these parameters we could get better performance

VI. Future work

During the implementation and testing we noticed several things that could be improved. First improvement could be performed by doing better preprocessing. For example the lemmatization could be applied in preprocessing in order to reduce the vocabulary size, especially on the Finnish language. There is some great resources and software that could be used

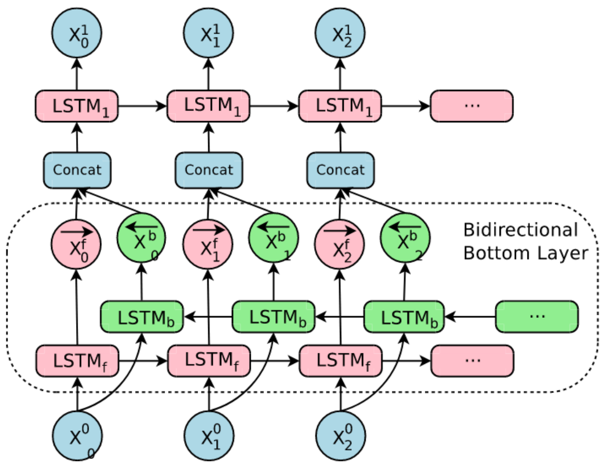


Figure 6. Multilayered bidirectional LSTM architecture of Google at 2016 [13].

for this purporse. One tool that could be used can be found from: [8]. Downside of this would be that it would have to be accounted in later stages and further steps in the system require special care in order to work with lemmatization.

The architectural optimizations of the encoder-decoder structure could be optimized. The better approach would be to use multilayered LSTM cells (Figure 6). Due to technical difficulties and limited training hardware, we decided to use only 1-layer LSTMs in encoder and decored. Despite of this, experimentation with different parameters could be interesting topic for further research. Other architectural change that could be tested with different methods is the passing of the hidden state of the encoder. Now we just average the bidirectional vectors produced by encoder, but the other methods such as concatenation could also be used.

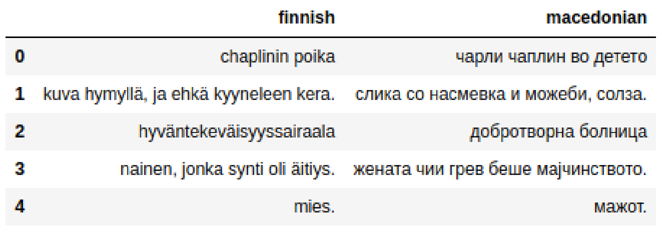
Another thing that could be done is to use attention mechanism [10] which removes the need for RNNs, gives better results and is faster to train. Also, we could have used some pre-trained models, such as BERT [2] and the fine-tune it for our specific task.

Another thing that could be done is to use pre-trained embeddings and feed them to the LSTMs. That approach usually yields better results that learning the embeddings from scratch, especially when we don't use that much data.

Due to the hardware and time limitations, we used only 30 epochs, even though the loss was still decreasing. Increasing the epochs would produce better results.

VII. Appendix

The source code can be found here:
https://github.com/Tetrix/Neural-Machine-Translation-Fin-Mk
Below we can see a small snippet of the data:



	finnish	macedonian
0	chaplinin poika	чарли чаплин во детето
1	kuva hymyllä, ja ehkä kyyneleen kera.	слика со насмевка и можеби, солза.
2	hyväntekeväisyyssairaala	добротворна болница
3	nainen, jonka synti oli äitiys.	жената чии грев беше мајчинството.
4	mies.	мажот.

Figure 7. Small snippet of the data

References

[1] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, arXiv preprint arXiv:1409.1259, (2014).

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805, (2018).

[3] L. v. d. Maaten and G. Hinton, *Visualizing data using t-sne*, Journal of machine learning research, 9 (2008), pp. 2579–2605.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781, (2013).

[5] C. Olah, *Understanding lstm networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, aug 2015.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, *Bleu: a method for automatic evaluation of machine translation*, in Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, 2002, pp. 311–318.

[7] R. Řehůřek and P. Sojka, *Software Framework for Topic Modelling with Large Corpora*, in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, May 2010, ELRA, pp. 45–50. http://is.muni.cz/publication/884893/en.

[8] M. Silfverberg, T. Ruokolainen, K. Lindén, and M. Kurimo, *Finnpos: an open-source morphological tagging and lemmatization toolkit for finnish*, Language Resources and Evaluation, 50 (2016), pp. 863–878.

[9] J. Tiedemann and L. Nygaard, *The opus corpus-parallel and free: http://logos. uio. no/opus.*, in LREC, Citeseer, 2004.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2017.

[11] Wikipedia contributors, *Rule-based machine translation — Wikipedia, the free encyclopedia*, 2019. [Online; accessed 28-April-2019].

[12] ———, *Statistical machine translation — Wikipedia, the free encyclopedia*, 2019. [Online; accessed 28-April-2019].

[13] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv preprint arXiv:1609.08144, (2016).