Corpus-Based Translation Automation of Adaptable Corpus Translation Module

Andriy Lutskiv, Roman Lutsyshyn

Ternopil Ivan Puluj National Technical University, Ruska 56, Ternopil, 46001, Ukraine[1]

Abstract

Thesis deals with the translation module development which automates corpus-based translation. The translation module is a part of an adaptable corpus tool and is implemented as a separate microservice. This translation module provides for the linguist and translator the ability to translate texts with conveying their style. The translation module is domain-specific oriented which allows to convey text style better than public cloud translation services. In this research religious and historical texts were analyzed. Neural machine translation method was justified and used. Sequence to sequence Transformer model as a neural network model was chosen. All stages of text processing by the Transformer model which based on the Multi-Head Attention mechanism were analyzed. Software libraries and toolkits for the Sequence to sequence Transformer model were analyzed and chosen. Based on chosen software libraries implemented own Transformer model implementation. Developed model comprises text preprocessing and neural network model implementation. Cost-efficient computer system which comprises hardware and software components for neural network model training was configured. Based on heuristic approach by carrying out computational experiments neural network model hyper-parameters were chosen and justified. Loss function, learning rate, perplexity and BLEU as a key model training criterion were analyzed and applied. Training and test samples of text data sets were prepared. Training and test data sets comprise language pairs of Ukrainian text fragments and their English equivalents. Configured neural network model was trained and tested. Automatic assessment approach of trained model which based on semantic closeness was suggested and tested.

Keywords 1

Natural language processing, neural machine translation, corpus-based translation, Transformer model, sequence to sequence model, deep learning, machine learning, adaptable corpus tool, Computer-Aided Translation

1. Introduction

There are many common and routine tasks in translation studies which arising today. Finding-out domain-specific terminology by translation material analysis, translating material from source language to one or more other target languages, creating or choosing appropriate specialized dictionaries, proofreading of translated texts are just a few of them. Nowadays all these tasks could be automated or fully resolved by means of semi or fully automated CAT tools. Important role among these techniques play corpora and corpus-based translation approach. Corpus linguistic technologies allow to simplify, clarify and increase the quality of the translation process. Corpus-based translation approach becomes increasingly accessible through advances in computer technologies. This approach implementation involves the use of computational linguistics, machine learning, deep learning, big data mining and other related information technologies. Researchers developed corpus tool [1-4] which allows to:

- ingest text data and preprocess it by making tokenization, POS- and syntactical tagging;

COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, April 22–23, 2021, Kharkiv, Ukraine
EMAIL: l.andriy@gmail.com (A. Lutskiv); lutsyshyn.ds@gmail.com@gmail.com (R. Lutsyshyn)
ORCID: 0000-0002-9250-4075 (A. Lutskiv); 0000-0002-3390-874X (R. Lutsyshyn)

- build frequency dictionaries;
- find and extract collocations with frequencies;
- find semantically close words and texts;
- find the most important concepts and related terms and documents;
- find importance of the terms for different documents;
- provide ability to work with parallel texts in different languages: there are equivalent text pairs English-Ukrainian, Russian-Ukrainian, English-Russian and also French, German and Italian texts are ingested.

This corpus-tool text data processing workflows are based on different natural language processing methods, statistical methods and machine learning techniques in particular Latent Semantic Analysis [5]. Thus, developed a corpora data platform – web-tool for a linguist and a translator which provides multiple corpora capabilities: data processing engine for extract transform and load text data into parallel corpora and user interface to make queries and to search. This research covers the process of extending this corpus data platform tool by implementing own text translation module for text translation in some narrowly specialized subject areas with a higher level of translation accuracy and cost efficiency than public cloud services.

2. Objectives of corpora-tool translation module development

For translation of ingested to corpora texts could be applied Google Translator or Microsoft Translation Service which successfully used for some common subject areas' texts, but there are few drawbacks of this approach:

- for some texts, especially in religious or historical domain this can cause style mismatching;
- in some cases, the language of the investigated texts is outdated or ancient;
- for some cases special dictionaries maybe needed.

This module should be able to translate custom thematically related texts, in particular:

- to convey the meaning of the text;
- to translate from ancient languages;
- to translate old dialects and dialects;
- to take into account stylistics of the translated text.

All these criteria could be fulfilled by using adaptable translation tool. The main objectives of this findings are focused on:

- choosing the most efficient machine translation (MT) method;
- finding computational time and memory efficient methods to represent words as a vector of numbers (semantic vectors);
- choosing the most accurate and efficient language models which could be implemented in machine translation methods;
- preparing testing text data samples;
- implementing the translation module as a software;
- verifying suggested approach.

While implementing this translation module the main translation quality criteria we are trying to achieve are:

- adequacy – is a criterion which depicts if all the meaning was expressed from the source to the target language;
- fidelity – is a criterion which reflects the accuracy of the source text meanings translation;
- fluency – is a criterion which shows grammatical correctness and how easy to interpret them.

3. Justification of machine translation method for corpus tool

Nowadays such methods of MT are used [6]: Rule-based MT (RBMT), Statistical MT (SMT) [7], Neural MT (NMT) [8,9] and hybrid (combines two of these methods).

Machine translation method choice based on developed corpus-tool and appropriate corpora data platform peculiarities:

1. rather small text datasets;
2. lack of good dictionaries in a few cases;
3. bilingual texts are partially available;
4. limited human resources;
5. highly specialized;
6. method conveys the text stylistics.

Now described above corpus tool was used for religious and historical texts analysis, so translation module should properly handle this text style.

The RBMT method requires good dictionaries, but sometimes task of the linguist is to build the dictionary based on the investigated text. So, criterion 2 is not always achievable. Also, RBMT requires manually set large number of rules and its configuration takes too much time, which is not comply with the criterion 4. Despite other advantages, like domain-independency, reusability of rules for new languages and unnecessity of bilingual texts, this method cannot be used in our case.

The SMT [7] method requires less manual work from linguist and provides more fluent translation if language model chosen properly. In addition, this method requires parallel texts, which are sometimes not available, and to ensure the appropriate level of statistics, we must provide a large number of texts. Therefore, according to criteria 1 and 3, this method can be partially used. However, this method meets criteria 2 and 4.

The NMT [8] method conform to criteria 1, 4, 5 and partially 2 and 3, but this method has its own drawbacks [9]. While develop our translation module we propose our own solutions to overcome some of these method issues. And some of these issues are not important in our case. We have to consider few of them:

The first issue "NMT systems have lower quality out of domain" [9] caused by training datasets variety used to train NMT models. The issue concerns Google Translate, Microsoft Translate and other general-purpose mass MT systems. Our corpora data platform is domain specialized, so training data and processed by trained model texts will be of the same domain and will have very similar stylistics.

The second issue states that small datasets can cause "Under low resource conditions, NMT produces fluent output unrelated to the input." These issues could be fixed only by proofreading and developed corpus tool provides additional tools to help linguist fix these possible issues.

The third issue concerns with the problem of rare words translation. If NMT does not find the target string, then term should be transliterated. Because, if in a language the form or ending of some words depends on the way in which they are used in sentences, we can obtain unpredictable results. Such problems can arise in highly-inflected languages (e.g. Latin, Polish, and Finnish) and could be partially resolved by using of byte-pair encoding.

The fourth issue – very long sentences translated with lower translation quality. This issue appeared on the sentences with more than 60 words. This problem could be solved by using attention mechanism [10]. Text sequence length depends on hardware architectural peculiarities, as usual vectors have sizes of 512. Too long vectors could cause performance degradation.

The fifth issue concerns with the usage of attention model which "does not always fulfill the role of a word alignment model". Very good option of corpus tool is ability to do alignment of a parallel texts – assigning to all of the words in the source text – equivalents in the target language. This problem directly related to the training data set and generated dictionary. Partially using of multi-head attention allows to increase translation accuracy.

The sixth problem caused by multiple translations of the same word. MT have weighted words for translation, but sometimes proper choice of this target word is questionable, because there is no strict assessment that this translation is good and that one is not. This issue also could be resolved in the proofreading stage. So, evident to use elements of SMT approaches when the linguist or translator are performing machine translations proofreading.

Despite possible mentioned issues NMT method can reduce the amount of the translator's routine work by full draft translation of the document. In this case the role of translator should be shifted from a translator to a proofreader and an expert in translation quality assessment.

4. Translation module development

Translation module of adaptable corpus tool could be treated as a Computer Aided Translation tool (CAT). Development of the modern CAT tool comprises multiple different aspects, but in this finding, we are focusing only on the implementation of text translation functionality and development of user interface will not be covered. Translation module implemented as a microservice which receives API request over HTTP with the texts in the source language and returns HTTP responses with the texts in the target language. This microservice is implemented as a Docker container (Fig.1).

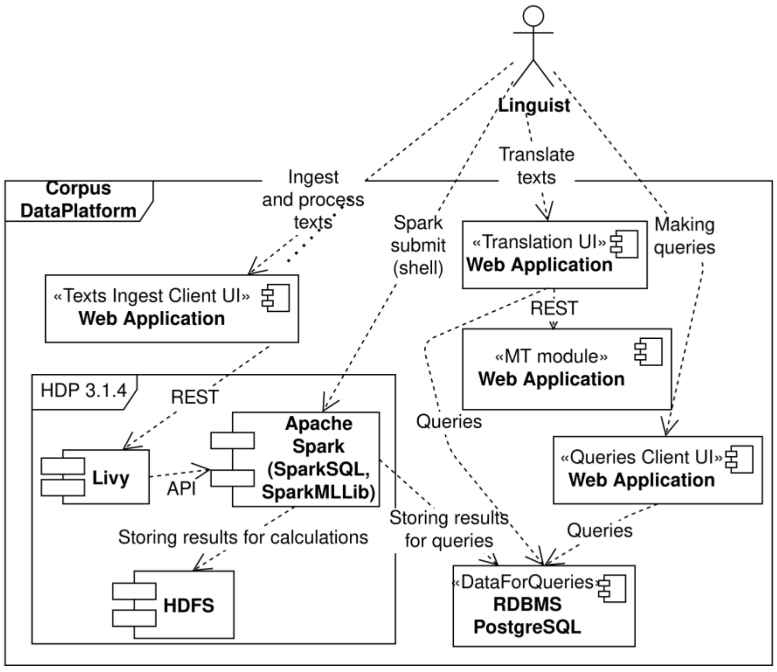

Linguist
Ingest and process texts
Translate texts
Spark submit (shell)
Making queries
Corpus DataPlatform
«Translation UI» Web Application
«Texts Ingest Client UI» Web Application
REST
«MT module» Web Application
HDP 3.1.4
REST
Apache Spark (SparkSQL, SparkMLLib)
Queries
Livy
API
«Queries Client UI» Web Application
Storing results for calculations
Storing results for queries
Queries
HDFS
«DataForQueries» RDBMS PostgreSQL

Figure 1: Translation module in the corpus data platform [3]

As shown above NMT method is the most suitable for our case. Also, previously developed corpus tool [1-3] will be partially used to provide some SMT features in the proofreading stage. So, to meet a specific requirements of this NMT-based CAT tool the following tasks should be resolved:

1. Training and test data sets preparation.
2. Text feature extraction and encoding.
3. Neural network model choosing, justification and implementation.
4. Chosen neural network model training with appropriate hyper-parameters to achieve sufficient translation accuracy level.
5. Developed translation module testing.

4.1. Training data set preparation

In this and previous findings [1-3] adaptable corpus oriented on the processing of historical and religious texts which have specific language style. Developed corpus tool was tested on different editions of the Bibles in different languages: English, Ukrainian, Russian, German, French and Italian [2]. These data sets described in [2,3]. In this research devoted to the translation module of adaptable corpus tool English [11] and Ukrainian Bibles [12] translations were used. For the training purposes 28474 pairs of Ukrainian equivalents in English text fragments were extracted and prepared. The size of this fragments was usually 1-3 sentences long. The size of the fragment is limited by computer system hardware capabilities of GPU [13]. These text fragments were semantically related. In this research only Ukrainian to English translation direction is demonstrated. Other directions and languages also were analyzed.

4.2. Text feature extraction and encoding

At the beginning of the text processing sentences from the source text by sentence tokenization [14] will be extracted. Sentences are treated as translation units. An equivalence of the source and target texts by these cognitive units could be established. Also, sentence will be tokenized into the terms (words). Thus, after tokenization, the sequence of terms with special tags which point on the beginnings and endings of the sentences will be obtained.

The second stage of text preprocessing is subword tokenization to eliminate rare word translation problem which was mentioned earlier in this finding. This issue will be resolved by byte pair encoding (BPE)[14]. By using this approach common pairs of consecutive characters will be replaced with a bytes that does not appear in that text data. The rare word will be split up into more frequent subwords. For example, word "counterattacked" appeared in the [11] HCSB Bible only once. But words "attacked", "attack" appeared 50 and 40 times respectively. Also, there are another words "counteract", "counter" and "act".

The third stage is vectorization – conversion text to vectors of numbers, due the fact that neural networks work only with numbers. So, in the first step text to number vector encoder was used. This task could be resolved in different ways [15-18]. In this context we are talking about vector semantics which are based on embeddings. Embeddings take into account word meanings and based on text distributions. Vectorization provides contextualized word representation. For resolving this task algorithm with common name word2vec will be used. Word embedding [16-18] is a text strings' (words or phrases) conversion to be amenable to processing by learning algorithms.

4.3. Analysis of a neural network model

The next step is choosing of neural network model for translation of the source text sequences embeddings into target text sequences. The right choice of a neural network architecture is crucial because source and target texts semantic equivalency depends on it. Neural network architecture should take into account few main characteristics of the language, such as:

- sequential structure of the language: the meaning of a word depends on the context, namely the previous and subsequent words.
- in the text an idea or a concept is represented by a word or a group of words. The same ideas in different languages could be represented with different number of the words in different orders.

Thus, neural network model should take into account these peculiarities. NMT accuracy depends on using of a proper language model – a statistical model where probabilities are assigned to words and sentences. This approach allows to predict the next word in the text sequence. Sequential structure of the language could be represented with a Sequence to sequence class [19] of neural networks models – models which allow to convert sequences from one domain to sequence of another domain. In our case sentences in source and target languages will be treated as two sequences. And processor of source sentence will be act as encoder and target sentence processor as decoder. As encoders and decoders widely used Recurrent Neural Networks (RNN) [14, 20]. But, for cases which require learning long-term temporal dependencies training of the standard RNN is rather difficult. Due to the vanishing gradient problem - the loss function gradient decays exponentially with time. Long Short-Term Memory (LSTM) or Gated Recurrent Unit as RNN subtypes for resolving this problem could be used. But, the main disadvantage of RNN models is their inability for parallelization due its sequential structure. To provide higher translation quality of the neural network model a large corpus of parallel texts for its training should be used. So, these models training are too expensive in a sense of computational time consumption.

The Transformer [10, 21, 22] is a neural networks model which represents sequential language structure and allows to parallelize the training process. Another advantage of this model that in translations for some specific domains it outperformed Google Neural Machine Translation model.

Transformers are a complex multi-component model, each component of which in itself is a model with a certain neural network architecture or machine learning tool. Therefore, each of these components may have its own parameters that directly affect the accuracy of the translation and the model learning process performance. Consider the components of the model and their parameters which are used in this study.

Transformer model as a sequence model comprises two main connected blocks: encoders and decoders stacks. In our implementation two stacks of six identical encoders and six identical decoders were used. The input of the first encoder of the encoders stack is a set of the sentences in the source language represented as a list of word embeddings. So, it is the list of 512-dimensional vectors. To provide correct order of the words in the target sentence to each of these embeddings will be added positional encoding vector of the same size. These vectors will be updated through the training process and provide distances between the words in the sentence. Calculation of positional encoding based on using sinusoidal function.

The output of the last decoder from decoders stack is the vector of floating numbers which will be mapped into words of the target language sentence.

Consider encoders stack. Each encoder contains layer of self-attention [10] mechanism and feed-forward neural network. The input of each encoder is the output of previous encoder: a list of 512-dimensional (in the Euclidean space) vector, except first encoder which input is the list of word embeddings with the positional encoding. The size of this list is the model hyperparameter and in our case was chosen equal to the longest sentence size of training data. The main advantage of the transformer model is that each term on its position flows through encoders by its own path. On the self-attention layer there are dependencies but on the feed-forward layer these dependencies are absent. This advantage provides ability to parallelize text data processing in the neural network.

Self-attention [10] mechanism allows to find relevancy between current processing word and other words in the sentence. Calculation of self-attention for the *i*-word in the sentence comprises follow steps:

1. Obtaining Query, Key and Value matrices *Q*, *K*, *V* of size *64×n*, where *n* – is the number of the words in the sentence. Value *n* equals to the size of the longest sentence. Rows of each of these matrices are the vectors of size 64: q_i, k_i, v_i Components of *Q*, *K*, *V* calculation could be written as follows:

$$\begin{cases} q_i = x_i \times W^Q \\ k_i = x_i \times W^K, i \in [1, n), n \in \mathbb{Z}, \\ v_i = x_i \times W^V \end{cases} \quad (1)$$

where x_i – the 512-dimensional vector which is the input of the encoder: at the first encoder it is a word embedding and at the next encoders it is the output of the previous encoder; all words of the sentence form matrix X of size $512 \times n$. W^Q, W^K, W^V – are the trained matrices of size 512×64. Initial values of these matrices are random.

2. Calculation of the score as a softmax function of q_i and k_i vectors dot product divided by $\sqrt{d_k}$ (*d* – dimension). This division by $\sqrt{d_k}$ provides more stable gradients and normalized values in the range [0,1) by the softmax function. Then this normalized score multiplied by Value vector. Scores obtained between all of the words in the sentence to provide relations between them, so vector q_i will be multiplied by $k_1 \ldots k_n$. The vectors $v_1 \ldots v_n$ will be summed up into one result vector z_i which will be the output of the self-attention layer for the word x_i.

$$z_i = \sum_{j=1}^{n} (softmax\left(\frac{q_i \times k_j}{\sqrt{d_k}}\right) \times v_j), i \in [1, n), j \in [1, n), \quad (2)$$

where d_k is the Key-vector dimension, so square root of it equals to 8. Also, self-attention calculation for the whole sentence could be written in the matrix form:

$$Z = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V, \quad (3)$$

To provide higher accuracy transformer model using multi-head self-attention mechanism which means that input sentence processing which was described above will be calculated 8 times independently and in parallel with different weight matrices which also will be trained independently. Than will be obtained 8 result matrices $Z_0 \ldots Z_7$ - attention heads. To provide single output matrix from self-attention layer $Z_0 \ldots Z_7$ matrices will be concatenated together into one matrix and multiplied by the weight matrix W^O. W^O is the matrix of size 512×64 will also be trained:

$$Z = Z_{Joined}^{0 \ldots 7} \times W^O, \quad (4)$$

where $Z_{Joined}^{0\cdots7}$ - the result of $Z_0 \dots Z_7$ matrices joining. Also these operation could be efficiently parallelized.

Thus, Z – is the output of the self-attention layer which will be the input of the next encoder's layer which is the feed-forward neural network. These feed-forward network is identical on all of the layers but has different parameters on the different layers. The input and the output of the fully connected neural network have dimensionality of 512. The hidden layer dimensionality is 2048. Neural network provides two linear transformations with ReLU activation function.

Also each of the encoder's sub-layer followed by add and normalize step with the residual connections: *LayerAddNorm(X+Z)* after self-attention sub-layer and *LayerAddNorm(Z+FFNN)* after feed-forward neural network.

Consider decoders stack. Decoding process begins after the encoding process finishes. The results of the encoding process (Key and Value vectors) from the encoders stack and Query vector from the previous decoder are used on each decoder's encoder-decoder attention sublayer of the decoders' step.

Output of each decoding process step is a value which represents a word in the target language text sequence. This process finishes after reaching special symbol. Each decoder's phase output is a vector which represents a word of the target sequence. After each decoder's phase this output will be send to the first decoder. Also, on decoder's input will be send positional encoded embeddings.

Decoder's self-attention layer can focus only on previous positions from the output sentence by masking all of the next positions after the current (before Softmax calculation in sub-layer self-attention calculation).

Output of the decoder's stack is the floating numbers vector. Linear layer and Softmax function are used to convert this vector into sequence of translated words. Linear layer is a fully-connected neural network which maps decoder's output vector into larger logits vector. Logits vector size equals to the size of our training corpus dictionary – each element of this vector represents coefficient of one unique word. The softmax layer translates logits vector into the probability. The softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The word which corresponds to the highest probability will be the translated word in the output sequence.

In the stage of conversion between neural network (softmax layer) output numbers into target text words beam search algorithm was used. The values of beam size equal to 4 and length penalty equal to 0.6 were chosen.

Thus, text vector embeddings flow in the trained model will be the same like in the model at the training process except the input of the decoder's stack, which during the training process takes translated text into its input. Trained model takes into the input (embedding positional encoder and then first encoder of encoders stack) text in the source language and return translated from the output (last decoder output and then linear with softmax layers transformations).

4.4. Neural network model training

To train sequence to sequence transformer model training text data set – the corpus of historical and religious texts was used. To provide sufficient level of translation accuracy the model should be properly evaluated during the training process. Key problems which could arise and which should be avoided are overfitting and underfitting. Also important is the cost of model training, thus computational time of the training process should be considered To achieve maximum efficiency and also sufficient translation accuracy some trade-offs between all these criteria were used.

4.4.1. Choosing and preparing training and test data sets

To achieve sufficient level of translation accuracy appropriate training data sets should be used. To train developed neural network training data set should fulfill further requirements:

- language pairs of texts: in the source and in the target language equivalents (reference translation);

- size of each pair must be in the range between 20 to 30 (60) words. It could be one, two or three logically and semantically connected sentences.
- it is desirable that the training data sets size be greater as possible.

To train designed neural model the texts of Bibles [11, 12] described in [2, 3] were used. System could be trained only on relatively short fragments of the texts (no more than 30 terms or 1-3 sentences). This size is chosen due to computational complexity and is the model peculiarity. The correctness of these text fragments were reviewed by the linguist.

4.4.2. Training model accuracy

Evident that translation metric quality depends on the quality of the neural model training. During the training process key criteria which should be in our focus: loss function, learning rate, perplexity and translation quality.

Loss function criterion

Loss function – is the function which allows to calculate the model error: how trained function or candidate (set of neuron weights) referred to the objective function. The loss function is calculated as the cross-entropy of the probability distributions of the trained model output - q and the predicted value – p. Cross-entropy was obtained by calculating the Kullback–Leibler divergence [23, 24]:

$$D(p|q) = \sum_{i}^{N} p\,(x_i) \cdot \big(logp(x_i) - logq(x_i)\big) \tag{5}$$

This value shows exactly how much information is lost when we approximate one distribution with another, thus the larger loss function value is worst.

Learning rate criterion

Learning rate – is a hyperparameter that determines how much of the model weights should be updated at each subsequent training epoch. It is one of the most important hyperparameters of the model. The learning rate value is in the range *[0,1)*. Too low learning rate value has impact on the training time, but too large learning rate value can cause unstable training process. Also, reasonable to change this value during the training process. In our research adaptive learning rates were used. In this research Averaged Stochastic Gradient Descent method was used (pytorch.optim.ASGD implementation). So, when during the training process this value becomes smaller and combination of other metrics (translation accuracy, perplexity, loss function) indicates sufficient quality of trained model the training process could be finished.

Perplexity criterion

Perplexity – is a language model intrinsic evaluation metric of how well a probability distribution predicts a sample. The sequence W of N-words perplexity could be represented as the exponent of the cross-entropy:

$$PP(W) = 2^{H(W)} = 2^{-\frac{1}{N}log_2 P(w_1, w_2, \ldots, w_N)} \tag{6}$$

It is the average number of words that can be encoded with *H(W)* bits. A low perplexity indicates good prediction.

Neural network translation quality criterion

The quality of NMT could be assessed by linguist expert. But due to multiple repetitions of neural network training (multiple epochs with multiple batches) and large amount of training data set assessment process should be automated by using MT quality metrics [25-27]. This quantitative characteristic directly related to human translation assessment. For training NMT used pairs in source and target languages. The simplest way to compare source and translation texts is to compare them by full text matching but this approach contradicts with multiplicity of translations. Thus, in this finding was used one of the MT quality metrics which highly correlate with human evaluation and takes into account morphology, synonyms and possibly different word order. By choosing any translation quality metric we have to keep in mind that it used to assess the quality of a developed model, but not the quality of translation. Among widely used MT qualitative metrics we focused in the few of them which in short will be considered and justified below. Also, few of these metrics (precision, recall, f-score) are itself the components of other metrics (BLEU, METEOR, ROUGE-n).

Precision is one of the simplest quality assessment approach, because it is taking into account the percentage of machine translation words that are correct. This metric doesn't take into account relations of the words.

Recall is a more precise metric because takes into account the percentage of reference translation words are transferred.

F-score (harmonic mean) is the balance between precision and recall.

WER (Word Error Rate)[26] is a word-based metric using Livenstein's weighted distance which takes into account insertions, deletions and substitutions.

TER (Translation Error Rate)[26] metric takes into account reordering of the text sequences (shift operations) and eliminate few WER disadvantages.

METEOR (Metric for Evaluation of Translation with Explicit ORdering)[26, 27] recall oriented metric which score based on calculation of fragmentation penalty (number of short sequences of consecutive matches divided by the number of unigram matches) and harmonic mean of precision and recall. This metric doesn't support direct exploitation from multiple reference translations.

ROUGE-n (Recall-Oriented Understudy for Gisting Evaluation)[28] is oriented on evaluation automatic summarization models. It is a system of metrics which comparing model produced translation against a set of reference translations.

BLEU (Bi-Lingual Evaluation Understudy) [25, 26] precision-oriented metric based on comparing *n*-grams' number in MT system output that matches reference translation texts and numbering this matches without taking into account their positions. BLEU is a measure of fluency rather than semantic similarity between machine and reference translations.

NIST (metric from US National Institute of Standards and Technology) metric based on the BLEU, but calculates informativeness of each n-gram by adding weights. Sometimes these weights could be questionable and also while using this criterion higher than in BLEU computation complexity should be taken into account.

Between considered metrics the most suitable are BLUE, METEOR and ROUGE. Due to language independency, wide adoption and low calculation complexity the BLUE metric was chosen. This metric is widely used because it highly correlate with human assessment, has several variants and supports multiple references. The BLEU metric based on modified precision p_n which represents adequacy and fluency of translation and used for the *n*-grams of length *n*. In a general form it could be written as follows:

$$p_n = \frac{\sum_{i=1}^{N} min\left(\text{Count\_any\_rt}, max(\text{cnt\_rt}_1, \text{cnt\_rt}_2, \ldots, \text{cnt\_rt}_m)\right)_i}{\sum_{i=1}^{N} \text{Count\_mt}_i}, n, m, i \in \mathbb{Z}, \tag{7}$$

where *N* – number of *n*-gram *i* in machine translation candidate;

Count_any_rt - maximum number of times *n*-gram *i* from machine translation candidate occurs in any single reference translation;

cnt_rt_m - number of *n*-gram *i* in reference translation *m*;

Count_mt_i - number of *n*-gram *i* in machine translation candidate.

In this research only one reference translation was used for BLEU calculation, so formula (7) could be rewritten in a form:

$$p_n = \frac{\sum_{i=1}^{N} \text{cnt\_rt}_i}{\sum_{i=1}^{N} \text{Count\_mt}_i}, n, m, i \in \mathbb{Z}, \tag{8}$$

To decrease the impact of too short translations and to compensate the absence of recall metric, p_n multiplied by (exponential) brevity penalty (*BP*):

$$BP = \begin{cases} 1, & wc_{mt} > wc_{ref} \\ e^{(1-wc_{ref}/wc_{mt})}, & wc_{mt} \le wc_{ref} \end{cases}, wc_{mt} \in \mathbb{Z}, wc_{ref} \in \mathbb{Z}, \tag{9}$$

where wc_{ref} – reference translation word count, wc_{mt} – machine translation word count.

Thus, BLEU metric could be calculated by the formula:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \cdot logp_n) \tag{10}$$

where w_n – weight of each modified precision (if N = 4, w_n = ¼ = 0.25).

Usually unigrams, bigrams, trigrams and fourgrams in BLEU metric are analyzed. In this case, the geometric mean of $BLEU_n$ scores will be considered as an overall BLEU score. The BLEU score could be represented in the range [0, 1] or could be scaled to the range [0, 100] (in our finding we are using this range):

- scores over 30 generally indicate understandable translations;
- scores over 50 usually indicate fluent and good translations.

Thus, MT output will be checked on matching the reference translations by the length, by properly chosen words and by word order.

4.4.3. Computer system implementation details

Based on cost and computational efficiency software and hardware components of the computer system were chosen.

Hardware components

It is important to notice that for neural model training process needed much higher computational resources than for using a trained model. Also, due to building an adaptable corpus tool, we have to take into the need of retraining the neural network. Theoretically neural networks can be trained on any hardware, but in practice even for small data sets computer equipped with the GPU and sufficient amount of memory is needed. In other case we will not obtain results in a reasonable amount of time. Minimal system requirements are 64 GB of memory and NVIDIA GPU[13] with at least 8GB of memory and no specific requirements for the processor.

There are two choices: using on-premise or cloud solutions. Computer system power consumption is not higher than two or three usual desktop personal computers so was not taken into account. Cloud hardware instances equipped with appropriate GPU accelerators of Amazon Web Services, Google Cloud Platform and Microsoft Azure were analyzed by authors. By the cloud provider's prices analysis the cost-effectiveness of on-premise solution was indicated. Thus, the on-premise solution with the characteristics shown in the table 1 have been chosen.

Table 1

Hardware components of the computer system for training the neural network model

Hardware component type	Vendor	Amount, pcs
Motherboard	MSI B450-A Pro Max Socket AM4	1
Videocard	EVGA GeForce GTX 1080 Ti SC Black Edition Gaming, 11GB GDDR5X	4
CPU	AMD Ryzen 3 3100	1
RAM	DDR4 16GB/3200 Team Elite	4
Case	GameMax ET-212-NP-U3	1
Power Supply	Asus ROG Thor 1200W	1

Software components

While developing the NMT module was used the most popular language in the data science domain – Python, because it has a large number of different libraries for text processing. Nowadays the most popular toolkits for deep learning are PyTorch[29] and TensorFlow with Keras [30]. These toolkits are backend parts of multiple different Python NLP deep learning libraries. There are a few implementations of the transformer approach which are based on these toolkits. Among these Transformer implementations the most suitable for our use are:

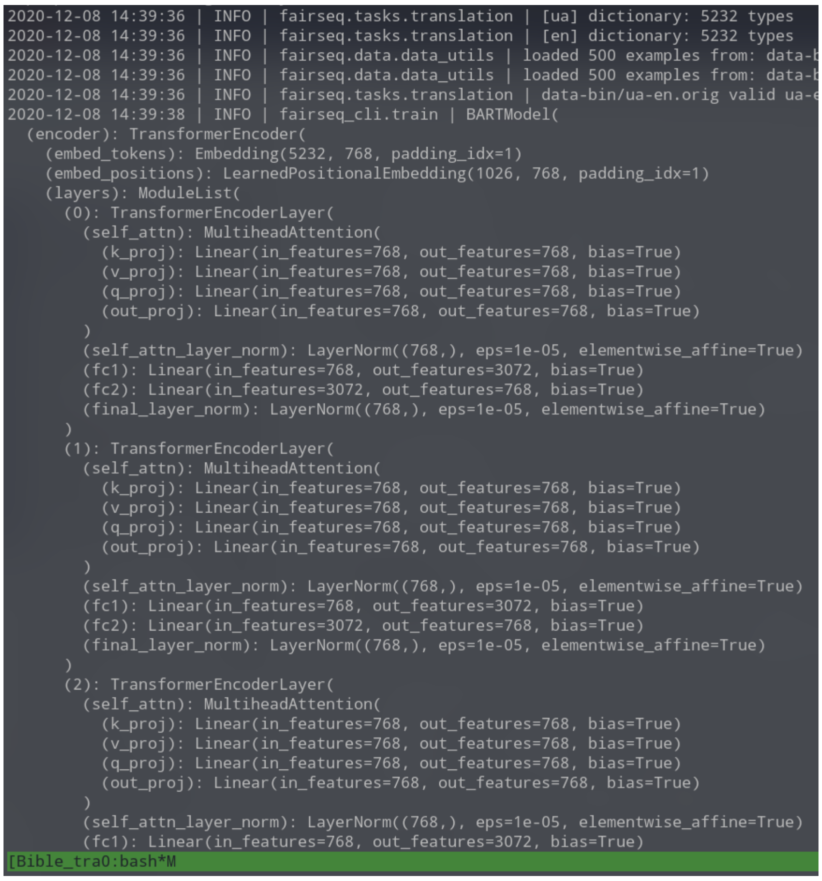
- Fairseq developed by Facebook AI Research and based on PyTorch [31];
- Tensor2Tensor developed by Google Brain Team and based on Tensorflow[32];
- OpenNMT developed by Harvard NLP Project and as a backend can use Lua, PyTorch and Tensorflow [33];

- AllenNLP developed by AI2 (Allen Institute for AI) and based on PyTorch[34].

Also there are frameworks: PaddlePaddle (PArallel Distributed Deep Learning developed by Baidu)[35], Sockeye (based on Apache MXNet)[36], Lingvo (based on Tensorflow) and others.

To resolve the main goal of our work Fairseq framework was used. It allows linguists and computational linguist experts to train custom models for translation, summarization, language modeling and other text generation tasks. The main advantage of this framework is possibility of full implementation of the ideas described in [10]. Other advantages of this framework are higher ability for configuration and easy use for production purposes. Among other advantages of this framework it is possibility to be executed on computational clusters equipped with multiple GPUs which allows to process large datasets.

Thus, our corpus tool [2,3] was used for text preprocessing. The tasks of useless characters filtering, sentence extraction, POS-tagging, feature extraction and few other tasks related machine learning are implemented using Java, Apache Spark, Stanford Core NLP, LanguageTool and other libraries. And NMT module was implemented with Python and Fairseq framework. Fragment of Transformer's encoder stack structure implementation is depicted in Figure 2.



```
2020-12-08 14:39:36 | INFO | fairseq.tasks.translation | [ua] dictionary: 5232 types
2020-12-08 14:39:36 | INFO | fairseq.tasks.translation | [en] dictionary: 5232 types
2020-12-08 14:39:36 | INFO | fairseq.data.data_utils | loaded 500 examples from: data-b
2020-12-08 14:39:36 | INFO | fairseq.data.data_utils | loaded 500 examples from: data-b
2020-12-08 14:39:36 | INFO | fairseq.tasks.translation | data-bin/ua-en.orig valid ua-e
2020-12-08 14:39:38 | INFO | fairseq_cli.train | BARTModel(
  (encoder): TransformerEncoder(
    (embed_tokens): Embedding(5232, 768, padding_idx=1)
    (embed_positions): LearnedPositionalEmbedding(1026, 768, padding_idx=1)
    (layers): ModuleList(
      (0): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (k_proj): Linear(in_features=768, out_features=768, bias=True)
          (v_proj): Linear(in_features=768, out_features=768, bias=True)
          (q_proj): Linear(in_features=768, out_features=768, bias=True)
          (out_proj): Linear(in_features=768, out_features=768, bias=True)
        )
        (self_attn_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (fc1): Linear(in_features=768, out_features=3072, bias=True)
        (fc2): Linear(in_features=3072, out_features=768, bias=True)
        (final_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      )
      (1): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (k_proj): Linear(in_features=768, out_features=768, bias=True)
          (v_proj): Linear(in_features=768, out_features=768, bias=True)
          (q_proj): Linear(in_features=768, out_features=768, bias=True)
          (out_proj): Linear(in_features=768, out_features=768, bias=True)
        )
        (self_attn_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (fc1): Linear(in_features=768, out_features=3072, bias=True)
        (fc2): Linear(in_features=3072, out_features=768, bias=True)
        (final_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      )
      (2): TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (k_proj): Linear(in_features=768, out_features=768, bias=True)
          (v_proj): Linear(in_features=768, out_features=768, bias=True)
          (q_proj): Linear(in_features=768, out_features=768, bias=True)
          (out_proj): Linear(in_features=768, out_features=768, bias=True)
        )
        (self_attn_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (fc1): Linear(in_features=768, out_features=3072, bias=True)
[Bible_tra0:bash*M
```

Figure 2: Transformer's Encoder layer implementation with Fairseq

4.4.4. Training process

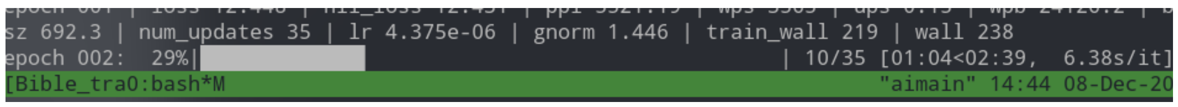
The training process comprises:

- preparing training and test data sets with the same texts in the source and in the target languages;
- configuring hyper-parameters of the model;
- estimating the accuracy of the training model during the training process to avoid underfitting or overfitting;
- assessing the quality of translation after the model was trained.

Translation module text preprocessing was implemented with a Python language, NLTK toolkit and other Python NLP libraries. Also for text data ingestion and preparation developed corpora-tool [2,3] was used. After reading and preprocessing of the input data set it was divided into train and test samples. Each of these samples contain Ukrainian-English pairs of texts.

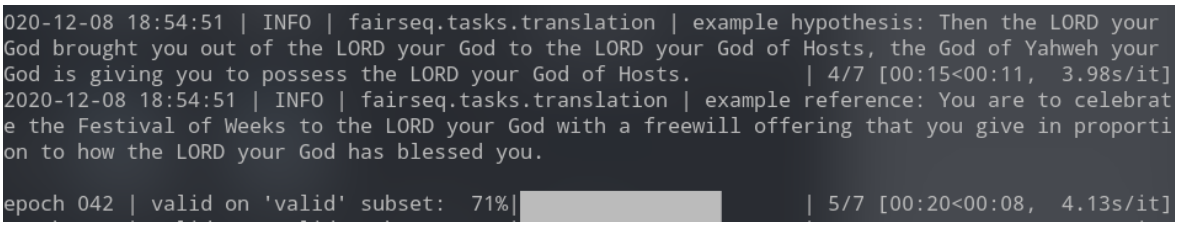
Sequence to sequence Transformer neural network model was implemented with a Fairseq toolkit which was described above and shown in Figure 2.

Training process was performed using configured computer system (Table 1) and its fragment is shown below (Fig. 3, Fig. 4).



```
sz 692.3 | num_updates 35 | lr 4.375e-06 | gnorm 1.446 | train_wall 219 | wall 238
epoch 002:  29%|                                               | 10/35 [01:04<02:39,  6.38s/it]
[Bible_tra0:bash*M                                                 "aimain" 14:44 08-Dec-20
```

Figure 3: Neural network model training at epoch 2

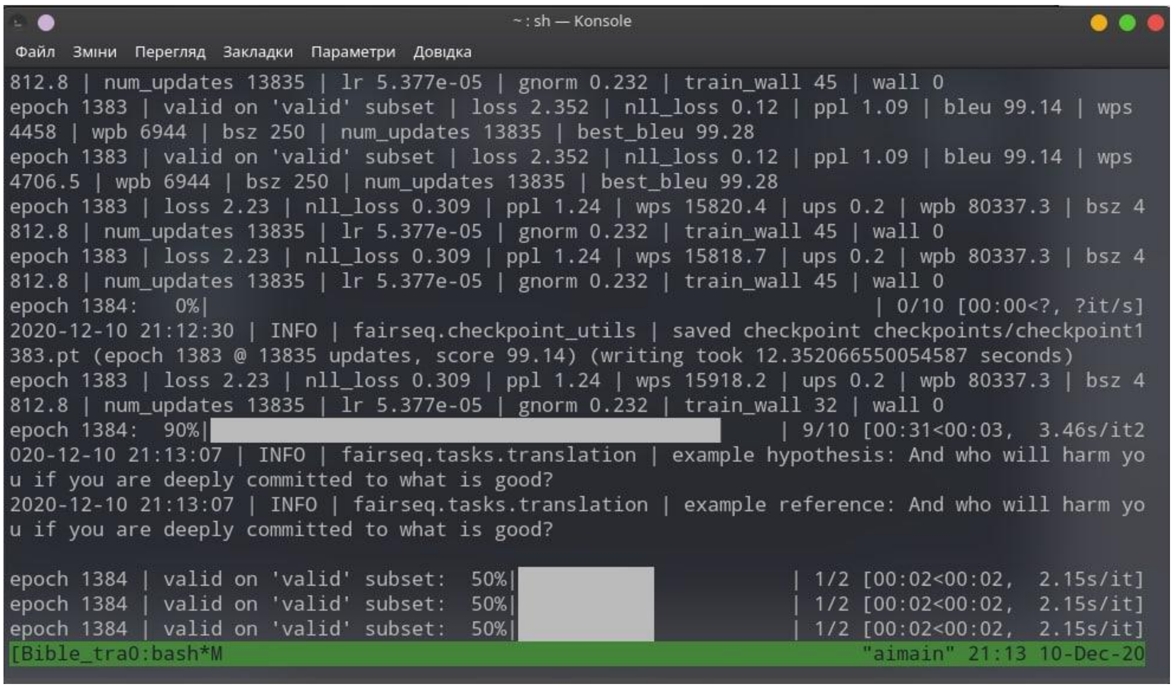


```
020-12-08 18:54:51 | INFO | fairseq.tasks.translation | example hypothesis: Then the LORD your
God brought you out of the LORD your God to the LORD your God of Hosts, the God of Yahweh your
God is giving you to possess the LORD your God of Hosts.          | 4/7 [00:15<00:11,  3.98s/it]
2020-12-08 18:54:51 | INFO | fairseq.tasks.translation | example reference: You are to celebrat
e the Festival of Weeks to the LORD your God with a freewill offering that you give in proporti
on to how the LORD your God has blessed you.

epoch 042 | valid on 'valid' subset:  71%|                          | 5/7 [00:20<00:08,  4.13s/it]
```

Figure 4: Neural network model training at epoch 42. Data validation stage.

The training process is subdivided into the epochs. During each of these epochs neural network model weights are updated by backpropagation to provide higher model accuracy and translation quality in general. The size of input data is too large to perform one epoch per one iteration, so epoch subdivided into batches. Sizes of batches depends on GPU [13] capabilities and in our case it was 2048 sentences on Figure 5 "bsz" parameter.



```
812.8 | num_updates 13835 | lr 5.377e-05 | gnorm 0.232 | train_wall 45 | wall 0
epoch 1383 | valid on 'valid' subset | loss 2.352 | nll_loss 0.12 | ppl 1.09 | bleu 99.14 | wps
4458 | wpb 6944 | bsz 250 | num_updates 13835 | best_bleu 99.28
epoch 1383 | valid on 'valid' subset | loss 2.352 | nll_loss 0.12 | ppl 1.09 | bleu 99.14 | wps
4706.5 | wpb 6944 | bsz 250 | num_updates 13835 | best_bleu 99.28
epoch 1383 | loss 2.23 | nll_loss 0.309 | ppl 1.24 | wps 15820.4 | ups 0.2 | wpb 80337.3 | bsz 4
812.8 | num_updates 13835 | lr 5.377e-05 | gnorm 0.232 | train_wall 45 | wall 0
epoch 1383 | loss 2.23 | nll_loss 0.309 | ppl 1.24 | wps 15818.7 | ups 0.2 | wpb 80337.3 | bsz 4
812.8 | num_updates 13835 | lr 5.377e-05 | gnorm 0.232 | train_wall 45 | wall 0
epoch 1384:   0%|                                                    | 0/10 [00:00<?, ?it/s]
2020-12-10 21:12:30 | INFO | fairseq.checkpoint_utils | saved checkpoint checkpoints/checkpoint1
383.pt (epoch 1383 @ 13835 updates, score 99.14) (writing took 12.352066550054587 seconds)
epoch 1383 | loss 2.23 | nll_loss 0.309 | ppl 1.24 | wps 15918.2 | ups 0.2 | wpb 80337.3 | bsz 4
812.8 | num_updates 13835 | lr 5.377e-05 | gnorm 0.232 | train_wall 32 | wall 0
epoch 1384:  90%|                                            | 9/10 [00:31<00:03,  3.46s/it2
020-12-10 21:13:07 | INFO | fairseq.tasks.translation | example hypothesis: And who will harm yo
u if you are deeply committed to what is good?
2020-12-10 21:13:07 | INFO | fairseq.tasks.translation | example reference: And who will harm yo
u if you are deeply committed to what is good?

epoch 1384 | valid on 'valid' subset:  50%|                   | 1/2 [00:02<00:02,  2.15s/it]
epoch 1384 | valid on 'valid' subset:  50%|                   | 1/2 [00:02<00:02,  2.15s/it]
epoch 1384 | valid on 'valid' subset:  50%|                   | 1/2 [00:02<00:02,  2.15s/it]
[Bible_tra0:bash*M                                                  "aimain" 21:13 10-Dec-20
```

Figure 5: Neural network model training at epoch 1384. Model is overfitted.

To estimate computer system throughput during the training process, the number of sentences per second could also be analyzed:

$$\text{sentences\_per\_second} = \frac{wps}{wpb} \cdot bsz \qquad (11)$$

In Figure 5 the follow training process characteristics were shown:

- loss – loss function (5);
- bsz – batch size;
- gnorm – L2 norm of the gradients;
- ppl – perplexity (6);
- lr – learning rate;
- wps – target words per second;
- ups – updates per second;
- wpb – words per batch;
- bleu – BLEU metric (10).

To finish the training process the follow rules were used. If during 4-5 epochs:

- the values of the loss function (5) and perplexity (6) are the same or change no more than by 0.01;
- the value of adaptive learning rate also is too small;
- the value of the BLEU metric (10) is not changing.

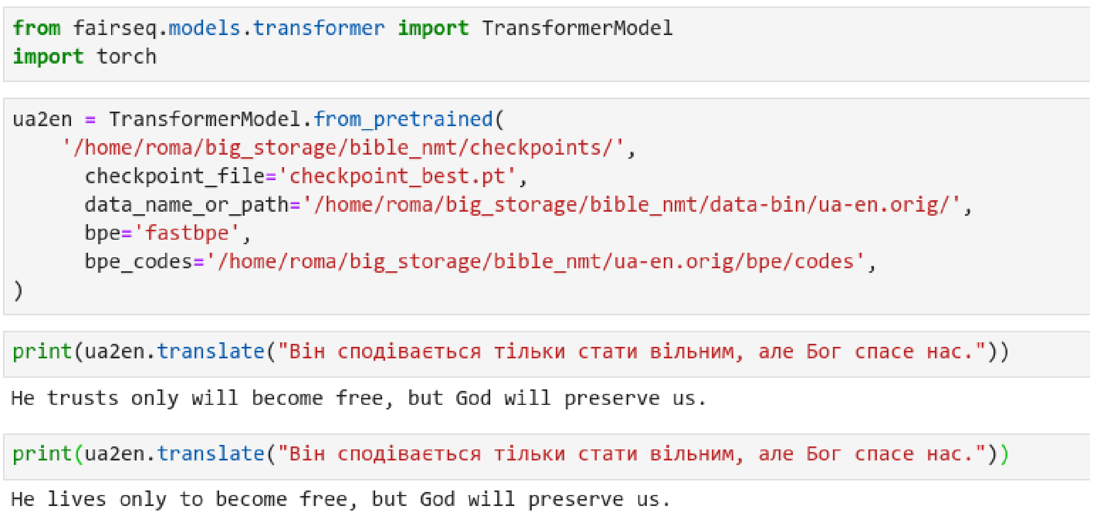
The computational experiment of the model training was performed several times. Based on described above criteria, using configured computer system with prepared data set, after 100-120 epochs training process was interrupted. It took about 2.5 days per one model training.

Also for computation experiment purposes was achieved overfitting of the model (Fig.5). This overfitting experiment took about 7 days.

The output of the training process are the model with indexed trained data terms and binary file with the weights and model states. This output is used to provide translation candidates.

4.4.5. NMT model translation quality assessment

Different religious texts were used to test trained model in particular the Apocrypha. The resulting translations convey the meaning and correspond to the style of the Bible. Testing of implemented model is shown in Figure 6.



```
from fairseq.models.transformer import TransformerModel
import torch
```

```
ua2en = TransformerModel.from_pretrained(
    '/home/roma/big_storage/bible_nmt/checkpoints/',
      checkpoint_file='checkpoint_best.pt',
      data_name_or_path='/home/roma/big_storage/bible_nmt/data-bin/ua-en.orig/',
      bpe='fastbpe',
      bpe_codes='/home/roma/big_storage/bible_nmt/ua-en.orig/bpe/codes',
)
```

```
print(ua2en.translate("Він сподівається тільки стати вільним, але Бог спасе нас."))
```

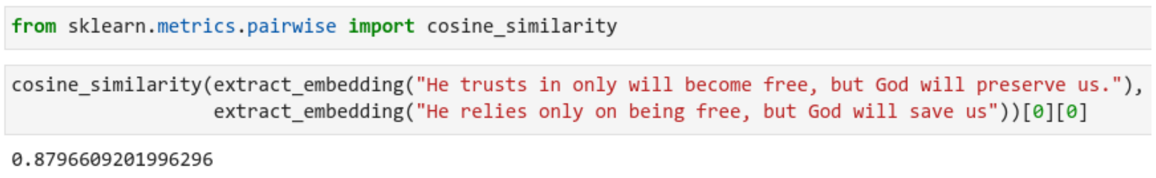
```
He trusts only will become free, but God will preserve us.
```

```
print(ua2en.translate("Він сподівається тільки стати вільним, але Бог спасе нас."))
```

```
He lives only to become free, but God will preserve us.
```

Figure 6: Translation of any text with trained model

To assess quality of translation was made assumption, that text from machine translation and text from translated test sample data set should be very semantically close and sometimes these texts should be identical. The follow approach was used: based on described above encoders text data were converted to vector space (text embeddings) and cosine similarity between these two vectors was calculated. The value of this metric is in the range [0, 1] and shows semantic similarity between machine translation and reference: 0 — semantically not close and 1 — identical strings. Example of this assessment in the Figure 7 is shown.



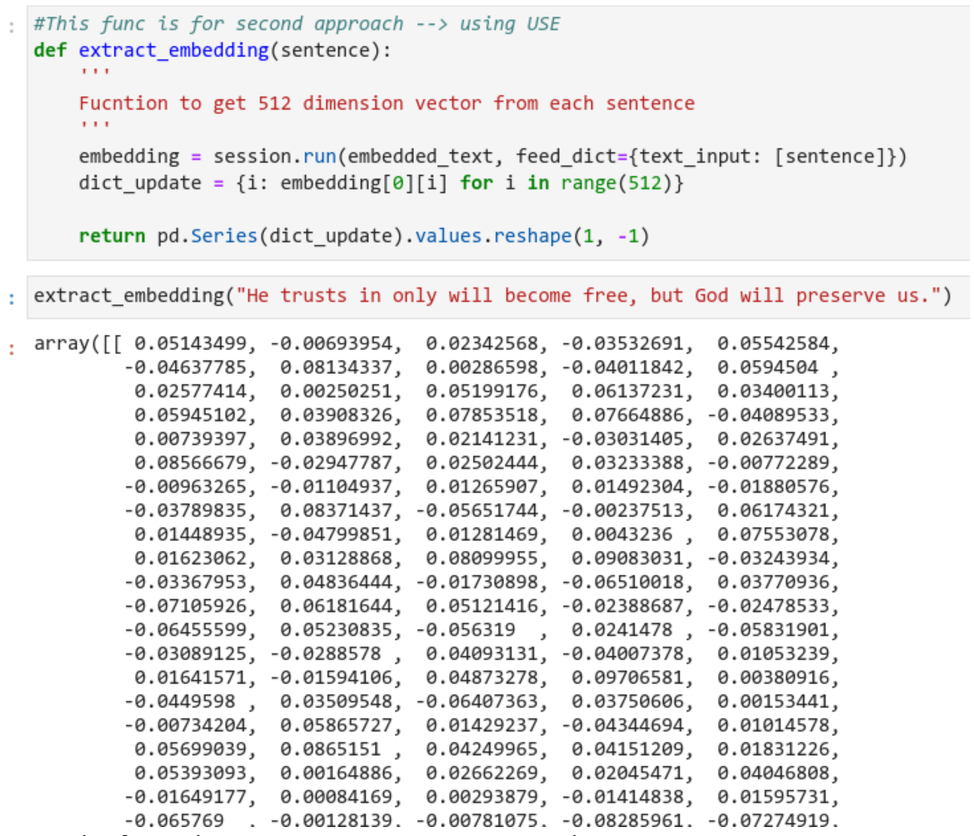
```
from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity(extract_embedding("He trusts in only will become free, but God will preserve us."),
                  extract_embedding("He relies only on being free, but God will save us"))[0][0]

0.8796609201996296
```

Figure 7: Semantic similarity between machine and reference translations assessing

Thus, obtained results could be treated as a metric of translation quality. Implementation of this approach in the Figure 8 is shown.



```
: #This func is for second approach --> using USE
  def extract_embedding(sentence):
      '''
      Fucntion to get 512 dimension vector from each sentence
      '''
      embedding = session.run(embedded_text, feed_dict={text_input: [sentence]})
      dict_update = {i: embedding[0][i] for i in range(512)}

      return pd.Series(dict_update).values.reshape(1, -1)

: extract_embedding("He trusts in only will become free, but God will preserve us.")

: array([[ 0.05143499, -0.00693954,  0.02342568, -0.03532691,  0.05542584,
          -0.04637785,  0.08134337,  0.00286598, -0.04011842,  0.0594504 ,
           0.02577414,  0.00250251,  0.05199176,  0.06137231,  0.03400113,
           0.05945102,  0.03908326,  0.07853518,  0.07664886, -0.04089533,
           0.00739397,  0.03896992,  0.02141231, -0.03031405,  0.02637491,
           0.08566679, -0.02947787,  0.02502444,  0.03233388, -0.00772289,
          -0.00963265, -0.01104937,  0.01265907,  0.01492304, -0.01880576,
          -0.03789835,  0.08371437, -0.05651744, -0.00237513,  0.06174321,
           0.01448935, -0.04799851,  0.01281469,  0.0043236 ,  0.07553078,
           0.01623062,  0.03128868,  0.08099955,  0.09083031, -0.03243934,
          -0.03367953,  0.04836444, -0.01730898, -0.06510018,  0.03770936,
          -0.07105926,  0.06181644,  0.05121416, -0.02388687, -0.02478533,
          -0.06455599,  0.05230835, -0.056319  ,  0.0241478 , -0.05831901,
          -0.03089125, -0.0288578 ,  0.04093131, -0.04007378,  0.01053239,
           0.01641571, -0.01594106,  0.04873278,  0.09706581,  0.00380916,
          -0.0449598 ,  0.03509548, -0.06407363,  0.03750606,  0.00153441,
          -0.00734204,  0.05865727,  0.01429237, -0.04344694,  0.01014578,
           0.05699039,  0.0865151 ,  0.04249965,  0.04151209,  0.01831226,
           0.05393093,  0.00164886,  0.02662269,  0.02045471,  0.04046808,
          -0.01649177,  0.00084169,  0.00293879, -0.01414838,  0.01595731,
          -0.065769  . -0.00128139. -0.00781075. -0.08285961. -0.07274919.
```

Figure 8: Approach of translation accuracy assessment implementation

Thus, to implement translation module for adaptable corpora-tool neural model was chosen and justified, configured with optimal parameters, trained and tested.

5. Conclusions

Developed translation module for corpora tool based on the neural network approach. Supervised learning for neural machine translation was used. Transformer sequence to sequence neural network for neural machine translation was chosen. Based on heuristic approach during computational experiments optimal architecture and hyper-parameters of the model were found and applied. Software tools and software libraries to implement translation module were chosen and justified. Optimal architecture of computer system for the model training in terms of the training time and the total cost of the system was proposed.

In about 100-120 epochs for model was trained. The method to automate translation quality assessing was suggested.

Thus, adaptable corpus tool equipped with translation module which allows to automate corpus-based translation and the tasks of linguistic research. This tool allows translators to load and translate texts with different supplement features provided by the corpora tool. This finding focusing on shifting the role of a translator from a translator to an assessor, a language expert and a proofreader.

6. References

[1] A. Lutskiv, N. Popovych, Adaptable Text Corpus Development for Specific Linguistic Research, Proceedings of IEEE International Scientific and Practical Conference Problems of Infocommunications. Science and Technology, Kyiv, (2019) 217-223.

[2] A. Lutskiv, N. Popovych, Big data-based approach to automated linguistic analysis effectiveness, Proceedings of the 2020 IEEE 3rd International Conference on Data Stream Mining and Processing, DSMP 2020, Lviv, (2020) 438-443.

[3] A. Lutskiv, N. Popovych, Big data approach to developing adaptable corpus tools CEUR Workshop Proceedings, Lviv, (2020) 374-395.

[4] N. Popovych, A. Lutskiv, A. Tyshtchuk, Corpus-Based Concept Translation, Fakhovyi ta khudozhnii pereklad: teoriia, metodolohiia, praktyka: zbirnyk naukovykh prats / za zah. red. A.H. Hudmaniana, S.I. Sydorenka. K.: Agrar Media Group, Kyiv. (2020) 306-314.

[5] T.K. Landauer, D.S. McNamara, S. Dennis, W. Kintsch, Handbook of Latent Semantic Analysis. Routledge Taylor & Francis Group, Lawrence Erlbaum Associates, Inc., 2011.

[6] M. Koponen, L. Salmi, M. Nikulin, A product and process analysis of posteditor corrections on neural, statistical and rule-based machine translation output, SpringerLink (2019). URL: https://link.springer.com/content/pdf/10.1007/s10590-019-09228-7.pdf.

[7] P. Brown, J. Cocke, S.D. Pietra, V. D. Pietra, F. Jelinek, R. Mercer, P. Roossin, A statistical approach to language translation. In Proceedings of the 12th conference on Computational linguistics-Volume 1.Association for Computational Linguistics, 1988, pp. 71-76.

[8] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi and S. Jain, Machine translation using deep learning: An overview., International Conference on Computer, Communications and Electronics (Comptelix), Jaipur, India, 2017, pp. 162-167, doi: 10.1109/COMPTELIX.2017.8003957.

[9] P. Koehn, R. Knowles, Six Challenges for Neural Machine Translation. Proceedings of the First Workshop on Neural Machine Translation, Association for Computational Linguistics,Vancouver, Canada, 2017, pp. 28-39.

[10] V. Ashish, S. Noam, P. Niki, Attention Is All You Need, Arxiv (2017). URL: https://arxiv.org/pdf/1706.03762.pdf.

[11] Christian Standard Bible, Holman Bible Publishers, Nashville, 1920, (2011).

[12] Novitni? pereklad Bibliyi Oleksandra H?zhi:(transl) Druk KT Zabelina-Fil'kovs'ka, Kyiv. 1210, (2013).

[13] Nvidia. Nvidia Deep Learning Performance Documentation, 2020. URL: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html.

[14] D. Jurafsky, J.H. Martin, Speech and Language Processing, 3rd ed. draft Prentice-Hall, Inc., Upper Saddle River, NJ, USA. (2019). URL: https://web.stanford.edu/~jurafsky/slp3/.

[15] Z. Wei, L. Feifei, W. Xiaodong, SubCharacter Chinese-English Neural Machine Translation with Wubi encoding, Arxiv (2019). URL: https://arxiv.org/pdf/1911.02737.pdf.

[16] M. Alawad and G. Tourassi, Computationally Efficient Learning of Quality Controlled Word Embeddings for Natural Language Processing, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Miami, FL, USA, 2019, pp. 134-139, doi: 10.1109/ISVLSI.2019.00033.

[17] T. Pisat, M. Bartakke and H. Patil, "Synonym Suggestion System using Word Embeddings", 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 2020, pp. 505-508, doi: 10.1109/ICOEI48184.2020.9142953.

[18] A. Sanzhar, A. Pak and J. A. Bulatovna, "The estimation of stability of semantic space generated by word embedding algorithms", International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), Pattaya, Thailand, 2018, pp. 1-5, doi: 10.1109/iSAI-NLP.2018.8692814.

[19] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks. In Advances in neural information processing systems (2014) 3104-3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

[20] K. Cho, B. Van Merri?nboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078. (2014) URL: https://arxiv.org/abs/1406.1078.

[21] Chenguang Wang, Mu Li, Alexander J. Smola. Language Models with Transformers. (2019) URL: https://arxiv.org/abs/1904.09408.

[22] G. Rohit, B. Laurent, D. Marc, G. Matthias, Character-based NMT with Transformer, Arxiv (2019). URL: https://arxiv.org/pdf/1911.04997.pdf.

[23] R. Rasipuram, M. Mathew, "Integrating articulatory features using Kullback-Leibler divergence based acoustic model for phoneme recognition", IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, 2011, pp. 5192-5195, doi: 10.1109/ICASSP.2011.5947527.

[24] S. Cui, M. Datcu, "Comparison of Kullback-Leibler divergence approximation methods between Gaussian mixture models for satellite image retrieval", IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 2015, pp. 3719-3722, doi: 10.1109/IGARSS.2015.7326631.

[25] K. Papineni, S. Roukos, T. Ward, Z. Zhu, BLEU: a Method for Automatic Evaluation of Machine Translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, 311-318.(2002) URL: https://www.aclweb.org/anthology/P02-1040.pdf

[26] A. Agarwal, A. Lavie, Meteor, M-BLEU and M-TER. Evaluation Metrics for High-Correlation with Human Rankings of Machine Translation Output. Proceedings of the Third Workshop on Statistical Machine Translation. Association for Computational Linguistics. Columbus, Ohio, USA, 115-118, (2008) URL: https://www.aclweb.org/anthology/W08-0312.pdf.

[27] L. S. Hadla, T. M. Hailat, N. Al-Kabi Mohammed, Comparative Study Between METEOR and BLEU Methods of MT: Arabic into English Translation as a Case Study. (IJACSA) International Journal of Advanced Computer Science and Applications, (2015) https://thesai.org/Downloads/Volume6No11/Paper_28-Comparative_Study_Between_METEOR_and_BLEU_Methods.pdf.

[28] K. Ganesan, ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks. Association for Computational Linguistics, (2006) URL: https://arxiv.org/pdf/1803.01937.pdf.

[29] R. Delip, Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning / R. Delip, M. Brian. - New York: O'Reilly Media, Inc, 2019.

[30] P. ?lvaro, C. Francisco, NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning, Arxiv (2018). URL: https://arxiv.org/pdf/1807.03096v3.pdf.

[31] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, M. Auli, fairseq: A Fast, Extensible Toolkit for Sequence Modeling, Arxiv (2019). URL: https://arxiv.org/pdf/1904.01038.pdf.

[32] A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. Gomez, S. Gouws, L. Jones, ?. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, J. Uszkoreit, Tensor2Tensor for Neural Machine Translation. Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track) (2018) 193-199. URL: https://www.aclweb.org/anthology/W18-1819/.

[33] G. Klein, Y. Kim, Y. Deng, J. Senellart, A.M. Rush, Opennmt: Open-source toolkit for neural machine translation. (2017). arXiv preprint arXiv:1701.02810. URL: https://arxiv.org/abs/1701.02810.

[34] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, L. Zettlemoyer, AllenNLP: A Deep Semantic Natural Language Processing Platform. URL: https://arxiv.org/abs/1803.07640.

[35] M. Yanjun, Y. Dianhai, W. Tian, W. Haifeng, PaddlePaddle: An Open-Source Deep Learning Platform from Industrial Practice. Frontiers of Data and Computing, 2019, 1(1): 105-115. URL: http://www.jfdc.cnic.cn/EN/10.11871/jfdc.issn.2096.742X.2019.01.011

[36] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton, M. Post, Sockeye: A Toolkit for Neural Machine Translation. URL: https://arxiv.org/abs/1712.05690.