



Software Engineering Department ORT Braude College

Course 61771 - Extended Project in Software Engineering

**Prose Style Transferring Agent Document**

In Partial Fulfillment of the Requirements for Final Project in Software Engineering (Course 61771)

Karmiel – January 2020

Din Golan 203247697

Matan Peer 305186702 Supervisor:

Dr. Dvora Toledano-Kitai

Advisor:

Prof. Zeev (Vladimir) Volkovich

Contents

[INTRODUCTION 4](#_bookmark0)

* 1. [Organization of the paper](#_Organization_of_the) 4

[BACKGROUND 4](#_bookmark1)

* 1. [Neural Network 4](#_bookmark2)
  2. [Recurrent Neural Network 5](#_Recurrent_Neural_Networks)
     1. [Bidirectional RNN 6](#_Bidirectional_RNN)
     2. [Deep RNN 6](#_Deep_RNN)
     3. [RNN Parameters 6](#_RNN_Parameters)
     4. [How RNN Memorizes 7](#_How_RNN_Memorizes)
     5. [Multi-Layer RNN 7](#_Multi-Layer_RNN)
  3. [RNN Encoder-Decoder](#_RNN_Encoder-Decoder) 7
     1. [RNN Encoder 8](#_RNN_Encoder)
     2. [RNN Decoder 8](#_RNN_Decoder)
     3. [Tanh 9](#_Tanh)
  4. [Style Transfer 9](#_Style_Transfer)
  5. [Machine Learning 9](#_Machine_Learning)
     1. [Supervised Learning 9](#_Supervised_Learning)
     2. [Unsupervised Learning 10](#_Unsupervised_Learning)
  6. [Natural Language Processing](#_Natural_Language_Processing) 10
     1. [Tasks 10](#_Tasks)
  7. [Deep Learning 11](#_Deep_Learning)
  8. [Long-Short Term Memory 11](#_Long-short_term_memory)
  9. [Sequence to Sequence 11](#_Sequence_to_sequence)
     1. [Sequence to Sequence Learning 12](#_Sequence_to_sequence_1)
     2. [One-to-Many setting 12](#_One-to-many_setting)
     3. [Many-to-One setting 12](#_Many-to-one_setting)
     4. [Many-to-Many setting 13](#_Many-to-many_setting)
     5. [TensorFlow](#_TensorFlow) 13
  10. [Bilingual Evaluation Understandy (BLEU) 13](#_Bilingual_evaluation_understandy)
      1. [Baseline of Bilingual Evaluation Understandy metrics 14](#_Baseline_of_bilingual)
      2. [Modified *n*-gram Precision](#_Modified_n-gram_precision) 14
      3. [Modified *n­*-gram Precision on blocks of text](#_Modified_n-gram_precision_1) 14
      4. [Ranking systems using only Modified *n*-gram Precision](#_Ranking_systems_using) 15
      5. [Bilingual Evaluation Understandy - Details](#_Bilingual_evaluation_understandy_1) 15
      6. [Bilingual Evaluation Understandy - Evaluation](#_Bilingual_evaluation_understandy_2) 16

[APPROACH](#_APPROACH_1) 17

* 1. [Architecture I 18](#_Architecture)
     1. [Encoder Model 18](#_Encoder_Model_1)
     2. [Attention 18](#_Attention_1)
     3. [Pointer Model 19](#_Pointer_Model)
     4. [Decoder Model 19](#_Decoder_Model_1)
     5. [Output Prediction 19](#_Output_Prediction_1)
  2. [Architecture II](#_Architecture_II) 20

[EXPECTED RESULTS 21](#_EXPECTED_RESULT)

[SOFTWARE ENGINEERING DOCUMENTATION](#_SOFTWARE_ENGINEERING_DOCUMENTATION) 21

* 1. [Use Case 21](#_bookmark37)
  2. [Class Diagram 23](#_bookmark38)
  3. [Sequence Diagram 24](#_bookmark39)
  4. [User Interface 24](#_User_Interface)
     1. [Windows 24](#_Windows)
     2. [Errors and alerts 30](#_Errors_and_alerts)
  5. [Testing Plan 32](#_5.5._Testing_Plan)

[RESULTS AND CONCLUSIONS 33](#_RESULTS_AND_CONCLUSIONS)

* 1. [Results 33](#_Results)
     1. [First example 33](#_First_example_-)
     2. [Second example 34](#_Second_example_–)
     3. [Compression between examples 36](#_Compression_between_first_1)
     4. [Third example 37](#_Third_example_–_1)
     5. [Fourth example 39](#_Fourth_example_–_1)
     6. [Compression between examples 41](#_Compression_between_third_1)
  2. [Conclusions 41](#_Conclusions)

7. [REFERENCES 42](#_bookmark44)

***Abstract:*** *In the prose style transfer task a system, provided with text input and a target prose style, produced output which preserves the meaning of the input text but changes the style. These systems require parallel data to assessment the results and use with parallel data for training. Currently, there are some publicly available corpora for this task. In this work, we identify a high-quality source of text that aligned and distinct in different versions of the bible. We provide a standard split, into training, development and testing data, of our public domain corpus versions. The corpus is parallel because many bible versions are included inside him. The sentences in the corpus is aligned due to the presence of chapter and verse numbers in the text. In addition to the corpus, we present the results, as measured by the BLEU (bilingual evaluation understands) metrics of several models that have been trained on our data, which can act as base for future research. While we present these data as a style transfer corpus, we believe that this may be useful for other natural language tasks as well.*

***Keywords:*** *Recurrent Neural Networks, Prose Style transfer, Sequence to sequence (Seq2Seq), BLEU (bilingual evaluation understand) metrics.*

# INTRODUCTION

Written prose is one way that we can communicate our thoughts to each other. Given a message, there are many ways to write a sentence that capable to pass the information, even when they are all written in the same language. Sentences can actually communicate the same information but makes that with various styles.

When writing a sentence, we refer not only the semantic content we want to communicate, but also the style, or manner, that we express it. Another wording may convey different levels of familiarity or politeness with the reader, which present different cultural information about the writer, and being easier to understand for particular populations.

Style transfer is the task of rewriting a sentence, its mean we need to preserve the meaning but change the style. There are a lot of characters of the prose that can contribute to the style transfer of the text including vocabulary level, using with passive or active voice, sentence length, tone and level of formality.

Systems typically require parallel data in the corpus for testing and training their results. The parallel data is intended for evaluation. We believe that one of the main obstacles for style transfer is the relatively low amount of parallel data in the corpus. The main contributions of our work are identification of high parallel corpus and publication of basic results for our corpus.[[1](#_ENREF_1)]

## Organization of the paper

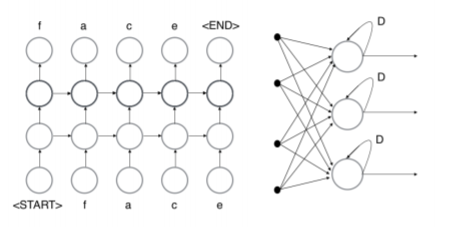
Section 2 - we start by recurrent neural network as the basic architecture for style transfer modeling, and how it is related to additional models in details. Section 3 - we propose architecture for style transferring while our goal is to make style transfer from source prose to target prose with sequence to sequence (Seq2Seq) model. Section 4 - we briefly discuss the expected result that we want to get. Section 5 - consist of preliminary software engineering documents - use case, class diagram, sequence diagram, and initial GUI, concluded by a short text plan section.

# BACKGROUND

## Neural Network

Neural network is a biologically inspired programming pattern that allows to computers to learns from observed data. It consists of many interconnected processing elements, neurons, and working together to solve a problem. In neural network, all the inputs, and outputs are independent from each other, but for many tasks that results is a not a good performance. If the next word in a sentence is predictable, it is necessary to know which word came before it, and for that we need the recurrent neural network (RNN) for sequential information [[2](#_ENREF_2)].

Neural network is a major part of artificial intelligence which allows networks to adjust this hidden layer in situations where the result doesn’t match the hoping.



*Figure 1 - Neural network*

<https://arxiv.org/pdf/1703.03091.pdf>

## Recurrent Neural Networks

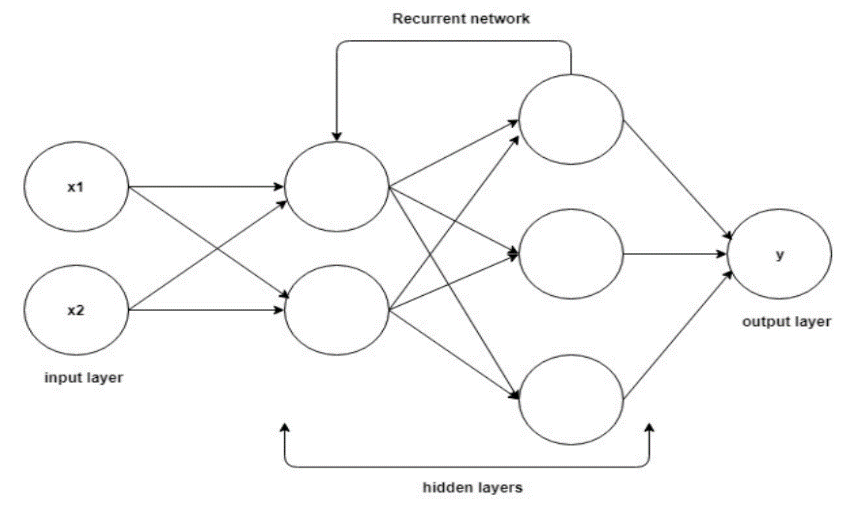
Recurrent neural network (RNN) is a neural network that include from output *y* and hidden state *h* which operates on a variable length sequence x = (). In RNN there have a loop, which allow information to persist. The hidden state of the RNN, at each time step *t*, is updated by -

*Equation 1 – The hidden state of RNN*

RNN can learn the probability of distribution across a sequence by being trained to predict the next symbol in a sequence. In that case, at each timestamp *t*, the output is the conditional distribution .

A multinomial distribution (1-of-K coding) can be output by using a SoftMax activation function for all possible symbols , where are the rows of a weight matrix *W* [[3](#_ENREF_3)].

*Equation 2 – SoftMax activation function*



*Figure 2 – Recurrent Neural network*<https://hackernoon.com/rnn-or-recurrent-neural-network-for-noobs-a9afbb00e860>

### Bidirectional RNN

The idea which the output at time *t* may not only depend on the previous elements in the sequence, but also future elements, is the base of Bidirectional RNN's.

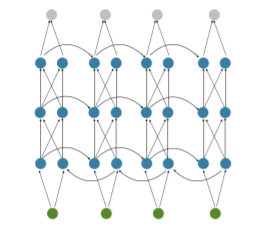


*Figure 3 – Bidirectional Recurrent Neural network*

<https://arxiv.org/pdf/1703.03091.pdf>

### Deep RNN

Deep RNNs are similar to Bidirectional RNN's, only that we now have multiple layers per time step. In practice Deep RNN gives us a higher learning capacity, but we also need more training data.

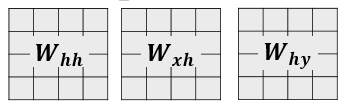


*Figure 4 – Bidirectional Recurrent Neural network*

<https://arxiv.org/pdf/1703.03091.pdf>

### RNN Parameters

In RNN there have three matrixes –



*Figure 5 – RNN’s Matrixes*

In training, these matrixes initialized with random numbers. We try to get the matrixes that give rise to desirable behavior, as measured with some loss of function that expresses our preferences to what kinds of outputs *y,* we like to see in response to our input sequence *x*.

### How RNN Memorizes

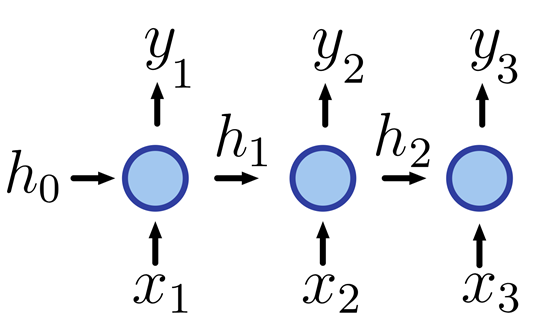
RNN steps –

1. Update the hidden state –

*Equation 3 – Hidden state*

1. Compute the output vector –

*Equation 4 – Vector in RNN*



*Figure 6 – RNN steps*

### Multi-Layer RNN

We can form a 2-Layer recurrent neural network as follows –

First, RNN is receiving the input vectors. Second, RNN is receiving the output of the first RNN as an input. In fact, RNN with multiple layers is made by independent RNN’s that not know each other and only pass the output of one of them as input of the others.

## RNN Encoder-Decoder

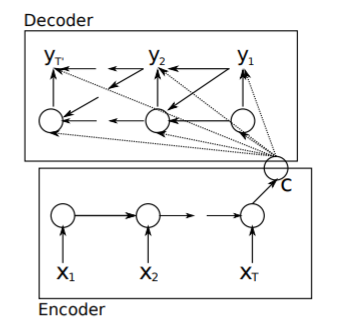
RNN architecture learns to encode a variable sequence into a fixed vector representation and to decode given fixed vector representation back into a variable sequence.

The encoder is an RNN that reads each symbol of an input sequence *x* sequentially. When the encoder finish to reads all the symbols, the hidden state of the RNN changes according to Equation (1). After the encoder reading the end of the sequence, the hidden state of the RNN is a summary *c* of the whole input sequence.

The decoder of the recommended model is another RNN which is trained to produce the output sequence by predicting the next symbol given the hidden state .

*Equation 5 – The hidden state of decoder at time t*

Once the RNN Encoder-Decoder is trained, the model can be used in two ways. The first way is to use the model for generating a target sequence given an input sequence. The second way is to use the model for scoring a given pair of input and output sequence.



*Figure 7 – RNN Encoder-Decoder*

<https://arxiv.org/pdf/1406.1078.pdf>

### RNN Encoder

Let us denote a source phrase by and target phrase by. Each phrase is a sequence of K-dimensional vectors, such that only one element of the vector is 1 and all the others are 0. The index of the active element indicates the word represented by the vector.

Each word of the source phrase is embedded in a 500-dimensional vector space:. The hidden state of encoder consists of 1000 hidden units and each one of them at time *t* is computed by -

*Equation 6 – The hidden state of encoder at time t*

*Equation 7 – Parameters that belongs to Equation (4)*

and are logistic sigmoid function and an element-wise multiplication, respectively. Once the hidden state at the N step (the end of the source phrase) is computed, the representation of the source phrase *c* is –

*Equation 8 – Representation of source phrase*

### RNN Decoder

The decoder starts by initialized the hidden state with -

*Equation 9 – The Initialized hidden state of decoder at time t*

The hidden state at time *t* of the decoder is computed by -

*Equation 10 – The hidden state of decoder at time t*

*Equation 11 – Parameters that belongs to Equation (8)*

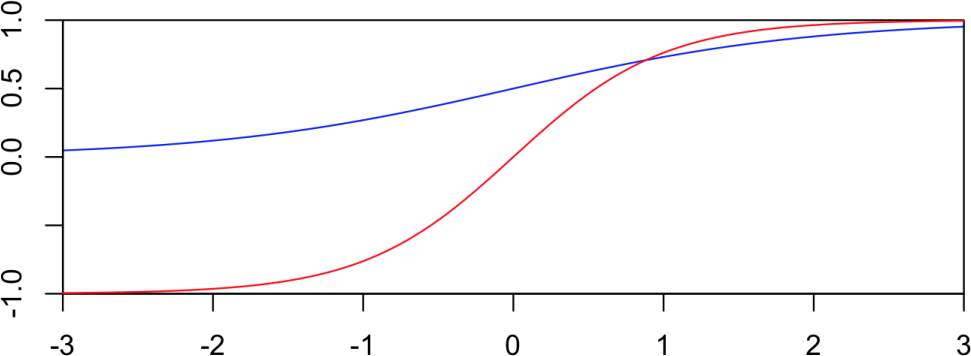
Unlike the encoder which simply encodes the source phrase, the decoder is learned to create a target phrase. The decoder computes the probability of creating *j* word by Equation (2).

### Tanh

Is a hyperbolic tangent function which is a rescaled logistic sigmoid function that its output range is [-1, 1]. The relationship between the value of the function at a point and his derivative reduces the computational burden during training according to Fausset.

*Equation 12 – Tanh(x) with g(x) as parameter*

*Equation 13 – Sigmoid function [0, 1] , Equation 12 – Tanh(x)*



*Figure 8 – Tanh(x) function (red)* [*https://brenocon.com/blog/2013/10/tanh-is-a-rescaled-logistic-sigmoid-function/*](https://brenocon.com/blog/2013/10/tanh-is-a-rescaled-logistic-sigmoid-function/)

## Style Transfer

Style transfer is known by the technique of recomposing a source image or text into the style of other target image of text. The content of the source stays the same with a different style, more likely as the target destination style.

## Machine Learning

There are several approaches for Machine Learning. The most significant is data mining. When People try to make relationship between multiple features, they often making mistakes during the analyses. Therefore, finding solutions to different problems is difficult to them.

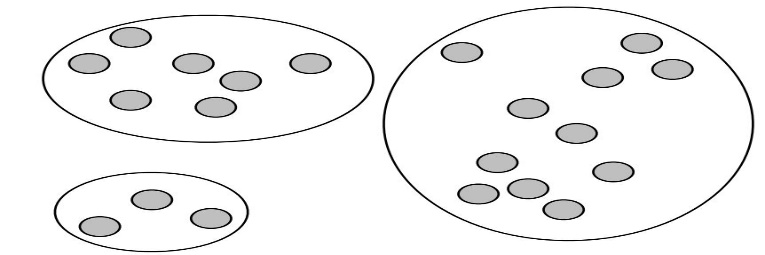
Machine learning can often be successfully applied to these problems. Furthermore, Machine learning can improve the efficiency of systems and the designs of machines. Each instance of dataset that used by machine learning represented set of features that identical. The set of features can be continuous, categorial or binary. If instances are given with known labels, then the learning is called supervised, otherwise it called unsupervised [[4](#_ENREF_4)].

### Supervised Learning

In supervised learning, we have training data encoded as pairs where the correct output is often manually annotated, and the learning goal is to adjust these parameters in order to fit the data. Types of features used to represent the input and varying degree of complexity [[5](#_ENREF_5)].

### Unsupervised Learning

In unsupervised learning, we try to learn the underlying patterns of our data. If there any correlations between that features, we can cluster our dataset into few groups which behave similarly.



*Figure 9 – Example of unsupervised data*

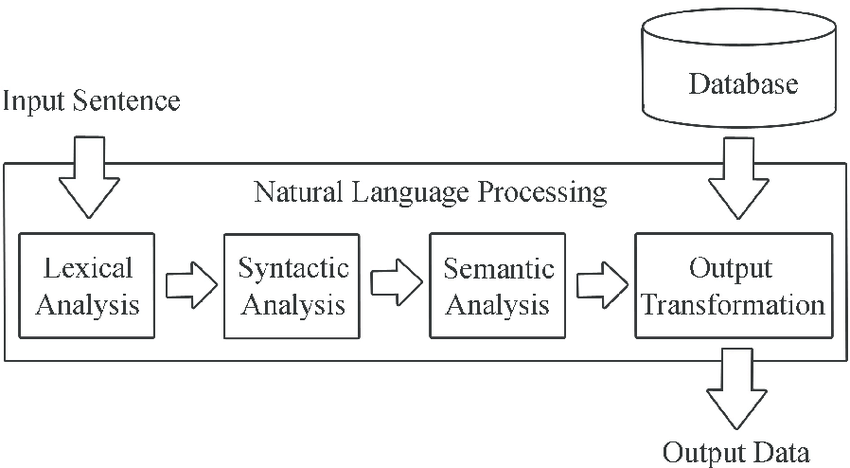
## Natural Language Processing

The field of natural language processing aims to convert human language into a formal representation is easy to computer to manipulate. Today, there have applications that includes information , machine translation, summarization, search and human computer interfaces.

### Tasks

The first task is part-of-speech tagging. This task labeling each word with a unique tag that indicates on syntactic role, for example – plural noun, verbs. The second task is chunking, also called shallow parsing. This task labeling segments of a sentence with syntactic constituent such as noun or verb phrase. Each word is assigned only one unique tag, often encoded as a begin chunk or inside chunk tag.

The third task is entity recognition. This task labeling atomic elements in the sentence into categories such as person, company, or location. The fourth task is semantic role. This task labeling a semantic role to syntactic constituent in a sentence. The fifth task is language models. A language model traditionally estimates the probability of the next word in a sequence. The sixth task is semantically related words. This task predicting whether two words are semantically related [[6](#_ENREF_6)].

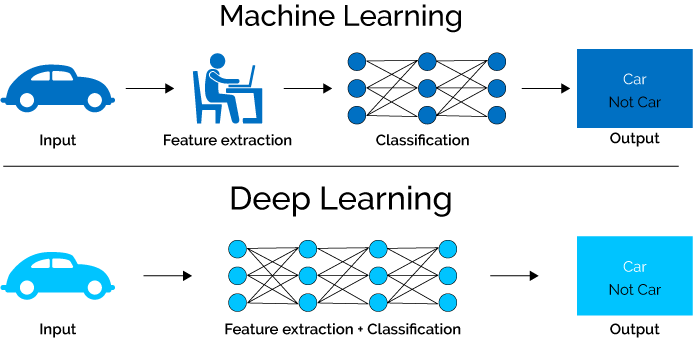
****

*Figure 10 – Natural language processing*

## Deep Learning

Deep learning refers to class of machine learning techniques, where many layers of information processing stages in hierarchical architectures are exploited for pattern classification and for feature or representation learning.

Deep learning techniques developed so far two additional key properties - The generative nature of the model, which typically requires adding an additional top layer to perform discriminative tasks, and an unsupervised pretraining step that makes an effective use of large amounts of unlabeled training data for extracting structures and regularities in the input features [[7](#_ENREF_7)].

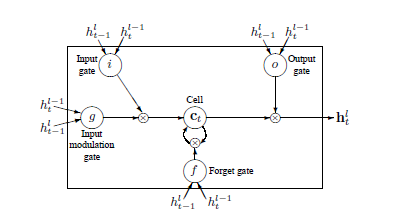


*Figure 11 – Example of unsupervised data*

## Long-Short Term Memory

Long short-term memory was introduced to allow a recurrent neural network to store information for an extended period .[[1](#_ENREF_1)] The LSTM has complicated dynamics that allow it to easily memorize information for an extended number of timestamps.

The long-term memory is stored in a vector of memory cells. Although many LSTM architectures that differ in their connectivity structure and activation functions, all LSTM architectures have explicit memory cells for storing information for long periods of time. The LSTM can decide to overwrite the memory cell, retrieve it, or keep it for the next timestamp [[8](#_ENREF_8)].



*Figure 12 – Graphical representation of LSTM memory cells*

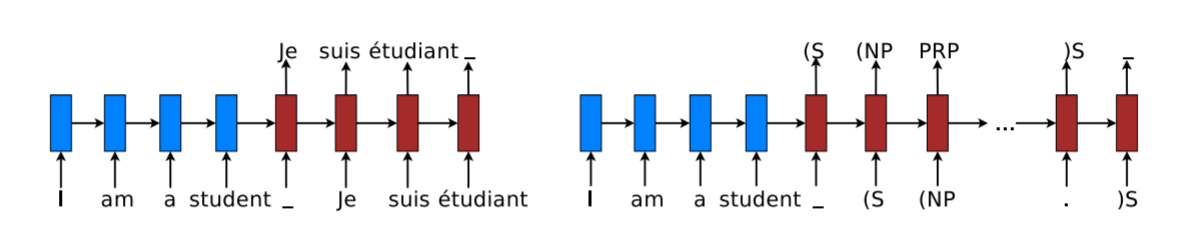
<https://arxiv.org/pdf/1409.2329.pdf>

## Sequence to Sequence

The Sequence to sequence model was created and used in conjunction with statistical methods to perform machine translation. The model consists of a recurrent neural network acting as an encoder, which generate an embedding of the full sequence of inputs. This sentence is then used by another recurrent neural network which acts as a decoder and generate a sequence that corresponding to the original input sequence. The Sequence to sequence model was adapted to use multiple LSTM (Long-short term memory) layers for the encoder side and the decoder side.

The model receives two languages as a pair and can learn to translate between them although they never appeared in the training data. The model adds artificial tags at the beginning of each source example to indicate the languages that belongs to the target during the decoding process. These models generally require many training examples to produce high-quality results.

For Sequence to sequence model there have three multi-task learning settings: (a) the *One-to-Many* setting – where the encoder is shared between some tasks such as machine translation and syntactic parsing, (b) the *Many-to-One* setting – useful when only the decoder can be shared, as in the case of translation and image caption generation, and (c) the *Many-to-Many* setting – where multiple encoders and decoders are shared, which is the case unsupervised objectives and translation.



*Figure 13 – Sequence to sequence learning example – (left) machine translation , (right) constituent parsing*

<https://arxiv.org/pdf/1511.06114.pdf>

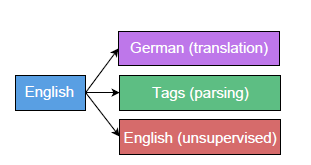
### Sequence to Sequence Learning

Sequence to sequence learning aims to directly model the conditional probability of mapping an input sequence, , into an output sequence, . The encoder calculates a representation *s* for each input sequence. Base on that input representation, the decoder create output sequence, one unit at a time, and hence, decomposes the conditional probability as –

*Equation 14 – Tanh(x) with g(x) as parameter*

### One-to-Many setting

This scheme consists one decoder and multiple decoders for tasks in which the encoder can be shared. The input for each task is a sequence of English words. A separate decoder is used to generate each sequence of output units which can be either a sequence of tags for constituency parsing, for machine translation, and for autoencoders.

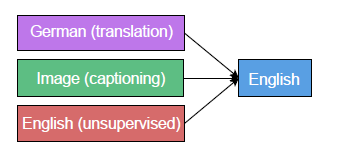


*Figure 14 – One-to-Many setting: one encoder, multiple decoders. This scheme is useful for multi  
target translation, or between different tasks*

<https://arxiv.org/pdf/1511.06114.pdf>

### Many-to-One setting

This scheme is the opposite of the *One-to-Many* setting. It consists multiple encoders and one decoder. *Many-to-One* is useful for tasks which only the decoder can be shared, for example, when our tasks include machine translation and image caption generation. In addition, from a machine translation perspective, this setting can benefit from a large amount of monolingual data on the target side, which is a standard practice in machine translation system.

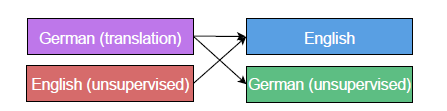


*Figure 15 – Many-to-One setting: multiple encoders, one decoder. This scheme is handy for tasks   
in which only the decoder can be shared*

<https://arxiv.org/pdf/1511.06114.pdf>

### Many-to-Many setting

This scheme is the most general. It consists multiple encoders and multiple decoders. We consider this scheme in a limited context of machine translation to utilize the large monolingual corpora in both the source and the target languages [[9](#_ENREF_9)].



*Figure 16 – Many-to-Many setting: multiple encoders, multiple decoders. This scheme consists   
single translation task and two unsupervised autoencoder tasks*

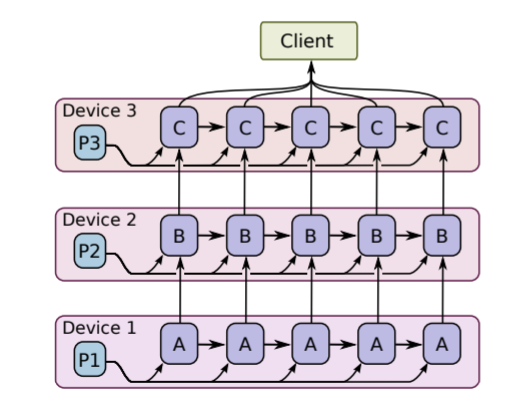
<https://arxiv.org/pdf/1511.06114.pdf>



### TensorFlow

Sequence to sequence model implemented using a publicly available library which itself makes use of the API provided by TensorFlow. TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms.

TensorFlow express model parallel training where portions of the model computation are done on different computational devices simultaneously for the same batch of examples [[10](#_ENREF_10)].



*Figure 17 – Model parallel training: In this picture we have recurrent, deep LSTM model   
used for sequence to sequence learning*

<http://download.tensorflow.org/paper/whitepaper2015.pdf>

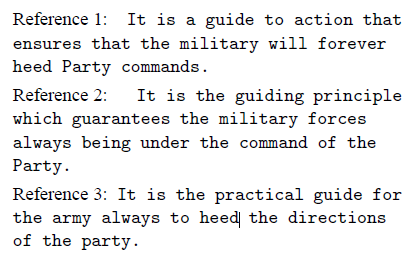
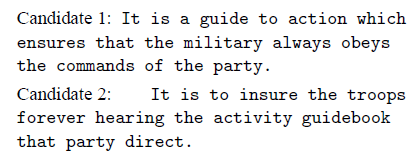
## Bilingual Evaluation Understandy (BLEU)

Bilingual evaluation understandy is a method for automatic machine translation evaluation. The method is quick, inexpensive, and language-independent, that correlates highly with human evaluation, and that has little marginal cost per run. The method is automated understandy skilled human judges which substitutes for them when there is need for quick or frequent evaluations.

Bilingual evaluation understandy only needs to match human judgment when averaged over a test corpus. Scores on individual sentences often vary from human judgments. The key to bilingual evaluation understandy success is that all systems are treated similarly and multiple human translators with different styles are used, so this effect cancels out in compressions between systems.

### Baseline of Bilingual Evaluation Understandy (BLEU) metrics

Bilingual evaluation understandy (BLEU) is a metric for comparing parallel corpora which rewards a candidate sentence for having n-grams which also appear in the target. Although it was created for evaluation of machine translation, it has been found that the scores correlated with human judgement when used to evaluate paraphrase quality. The correlation was especially strong when the source sentence and candidate sentence differed by larger amounts over words. Typically, there are many perfect translations of a given source sentence. These translations may vary in word choice or in word order even when they use the same words.



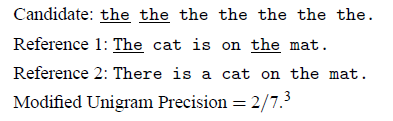
*Figure 18 – Example of two candidate's translations of a Chinese source sentence, and three  
reference human translations of the same sentence*

<https://www.aclweb.org/anthology/P02-1040.pdf>

According to Figure 10, although they appear to be on the same subject, they differ markedly in quality. The good translation is candidate 1, because he shares many words and phrases with these three references translations.

### Modified *n*-gram Precision

The cornerstone of our metric is the familiar precision measure. To calculate accuracy, we counts the number of candidate translation words (unigrams) which occur in any reference translation and then divides by the total number of words in the candidate translation.



*Figure 19 – Modified n-gram precision*

<https://www.aclweb.org/anthology/P02-1040.pdf>

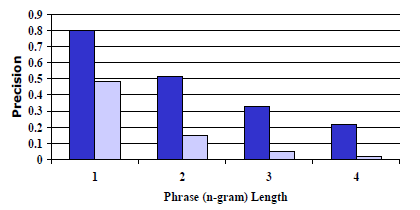
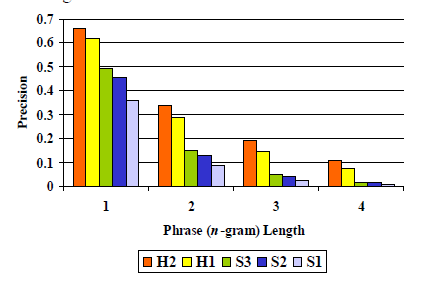
### Modified *n*-gram Precision on blocks of text

A source sentence may translate too many target sentences, in which case we abuse terminology and refer to the corresponding target sentences. First, we compute *n*-gram that matches sentence by sentence. Next, we add the clipped *n­*-gram counts for all the candidates sentences and divide them by the number of candidates *n*-grams in the test corpus to compute a modified precision score for entire test corpus.

*Equation 15 – Modified precision score*

### Ranking systems using only Modified *n*-gram Precision

To verify that modified *n*-gram precision is different between very good translations and bad translations, we calculated the modified precision numbers on the output of a (good) human translation and a standard (poor) machine translation system.



*Figure 20 – (Left picture): Distinguishing human from machine   
(Right Picture): Machine and human translations*

<https://www.aclweb.org/anthology/P02-1040.pdf>

According to Figure 12, the strong signal differentiating human (high precision) from machine (low precision) is striking. The difference becomes stronger when passing from unigram precision to 4-gram precision.

### Bilingual Evaluation Understandy (BLEU) - Details

First, computing the geometric1 average of the modified *n*-gram precisions , using *n*-gram up to length *N* and positive weights summing to one. Next, the meaning of variable *c* is the length of the candidate translation. Also, there has the variable *r* that represent the effective reference corpus length.

*Equation 16 – BP = brevity penalty*

Then,

*Equation 17 – formula of bilingual evaluation understandy*

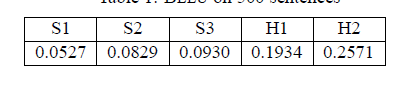
The ranking behavior is more immediately apparent in the log domain –

*Equation 18 – accurate formula of bilingual evaluation understandy*

1 The geometric average is harsh if any of the modified precisions vanish, but this should be an extremely rare event in test corpora of reasonable size (for . Using with geometric average yields stronger correlation with human judgments than our best results using an arithmetic average.

### Bilingual Evaluation Understandy - Evaluation

The bilingual evaluation understandy metric ranges from 0 to 1. Few translations are given a score of 1, unless they are the same as the reference translation. For this reason, even a human translator not necessarily score 1. It is important to note that if there have a lot of reference translations per sentence, than the score will be higher [[11](#_ENREF_11)].

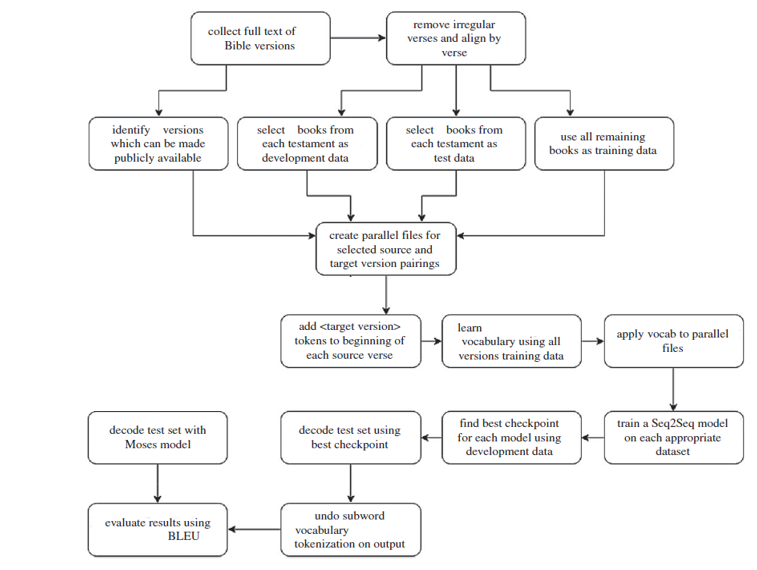


*Table 1 – Bilingual evaluation understandy on 500 sentences*

# APPROACH

In our project, we have hypothesis that the results of Seq2Seq model for the Hebrew will be accurate as the results of Seq2Seq model for English, and that’s why we want to use with Seq2Seq functionality. In terms of organization, we organized our model for bibles that written in Hebrew. The structure of Hebrew is different from English, because in English there have more letters than Hebrew, and the vocabulary size in English is bigger than the vocabulary size in Hebrew.

Our flow of the project should be as follow - collect full text of few bibles in Hebrew versions which be made publicly (the amount of bibles in Hebrew is fewer than in English), remove irregular verses and align by verse, select books from each testament as development data, test data, and training data, create Parallel files for selected source and target version pairings, add target version tokens to beginning of each source verse, learn sub words from Hebrew vocabulary using all versions training data, apply Hebrew vocabulary to parallel files, train sequence to sequence model on each dataset, find best checkpoint1 for each model using development data, decode test set using best checkpoint1, undo sub word vocabulary and tokenization on output, and finally evaluate result using bilingual evaluation understandy (BLEU).



*Figure 21 – Diagram of the project workflow*

<https://royalsocietypublishing.org/doi/pdf/10.1098/rsos.171920>

1 Models trained on smaller amounts of data tend to overfit faster. To ensure we have high-quality checkpoint we need to save them when we are training on smaller datasets. We saved a checkpoint every 5000 steps when we use multiple sources and multiple targets, and every 1000 steps when we use single target.

## Architecture I



Decoder

*Figure 22 – Architecture I: Encoder-Decoder with attention module* <https://aclweb.org/anthology/W17-4902>

The first architecture option for our model. In this model, we use with bidirectional LSTM to encode the input. Our decoder side is a mixture model of RNN module and pointer network module. There have two individual modules that share the attentions weights over encoder states. The decoder RNN predicts probability distribution of next word over the vocabulary, while the pointer model predicts the probability distribution over words in input.

Attention weights are computed by decoder hidden state, encoder representations, and sentinel vector. Attention weights are shared by decoder RNN and pointer models. The final probability distribution over vocabulary derives from decoder RNN and pointer network.



### Encoder Model

Let represent the forward encoder. represent the hidden state of encoder model at   
step *t*. The following equations describe the model –

*Equation 19 – Hidden state of encoder model when t = 0*

*Equation 20 – Hidden state of encoder model at time t*



### Attention

represent the hidden state of the decoder LSTM at step *t.* represent the decoder side of the previous step output. First, we compute a query vector. The query vector is a linear transformation of . A sentinel vector *s* is concatenated with the encoder states to create , where represents the number of tokens in encoder input sequence *x*. is a normalized attention weight vector. The value *g* represents the weight given to the decoder RNN module while computing output probabilities.

*Equation 21 – query vector*

*Equation 22 – sentinel vector concatenated with encoder states*

*Equation 23 – attention weight vector*

*Equation 24 – Normalized attention weight*

*Equation 25 – Normalized attention weight vector*

*Equation 26 – Weight given to the decoder RNN module while computing output probabilities*

### Pointer Model

A pair of corresponding original and modern sentences have significant vocabulary overlap. There are lot of proper nouns and rare words which might not be predicted by a sequence to sequence model. Pointer networks have been used to enable copying of tokens from input directly [[12](#_ENREF_12)]. The pointer models provide location-based attention, and output probability distribution. The formula of pointer network module can be expressed as –

*Equation 27 – Pointer network module*



### Decoder Model

Summation of encoder states weighted by corresponding attention weights yields context vector. Output probabilities over vocabulary per decoder LSTM module are computes as follow –

*Equation 28 – Summation of encoder states weighted by corresponding attention*

*Equation 29 – hidden state of decoder model at time t*

*Equation 30 – Probabilities over vocabulary per decoder LSTM module*

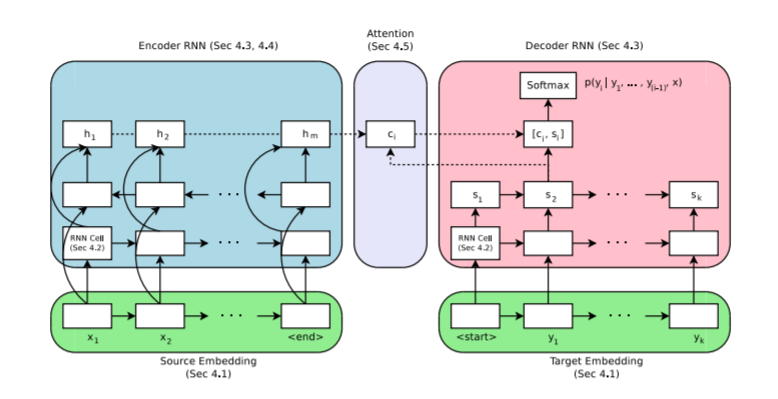


### Output Prediction

Output probability of a token *w* at step *t* is a weighted sum of probabilities from decoder LSTM model and pointer model given as follow –

*Equation 31 – Weighted sum of probabilities from decoder LSTM and pointer models*

## Architecture II



*Figure 23 – Architecture II : Encoder-Decoder with attention module*

<https://www.aclweb.org/anthology/D17-1151>

The second architecture option for our model. This model based on encoder-decoder with attention mechanism. An encoder function takes as input a sequence of source tokens and produces a sequence of states .

The encoder function is bi-directional RNN and the state produced by the backward and forward RNNs, . The decoder is an RNN that predicts the probability of a target sequence based on the sequence of states . The probability of each target token is predicted based on the recurrent state in the decoder RNN , the previous words , and a context vector [[13](#_ENREF_13)].

There has another name to the context vector . It's called the attention vector and is calculated as a weighted average of the source states.

*Equation 32 – Context vector*

The attention function calculates an unnormalized alignment score between the encoder state and decoder state .

*Equation 33 – Attention function*

The decoder outputs are a distribution over a vocabulary of fixed size V :

*Equation 34 – Decoder output distribution*

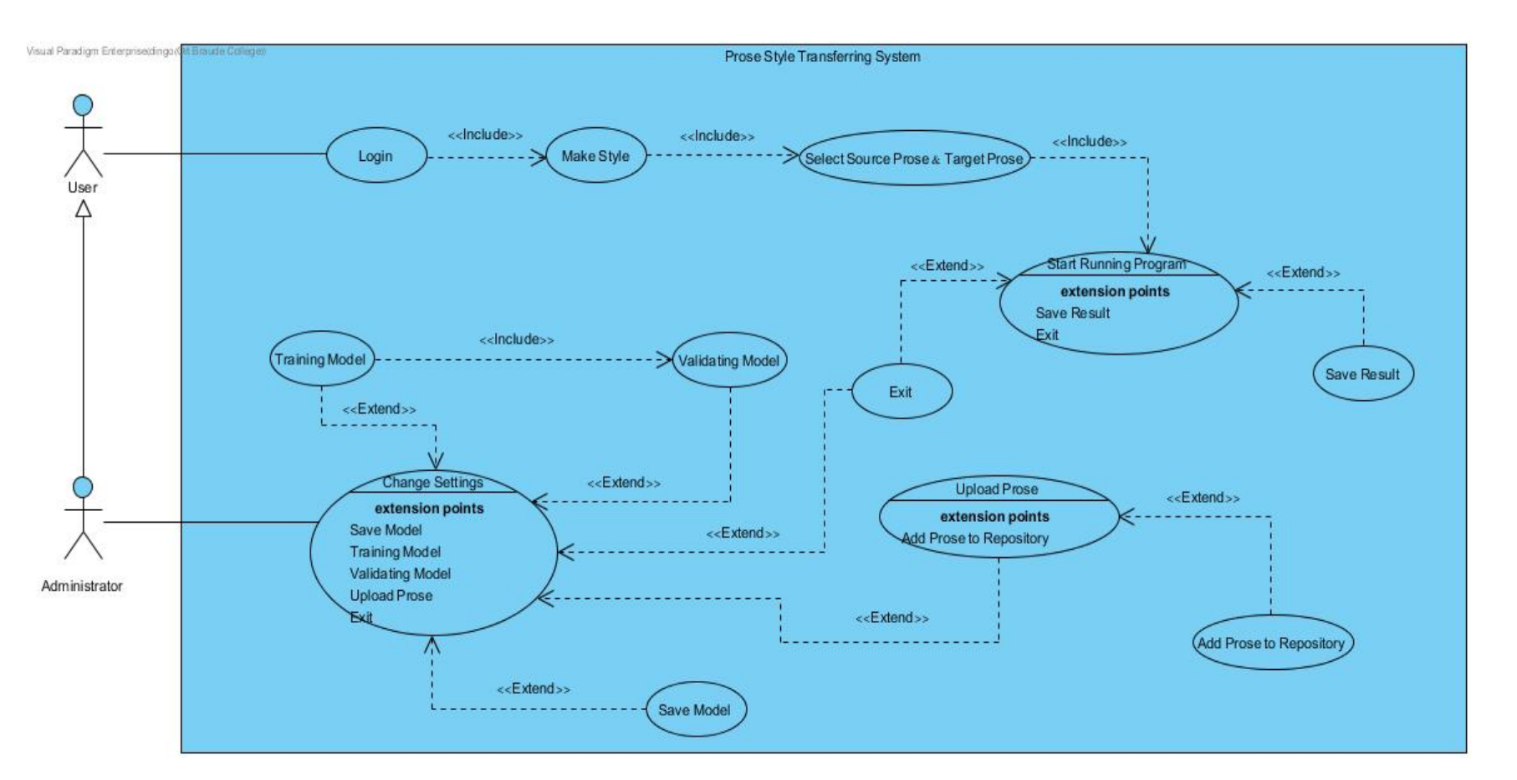
# EXPECTED RESULT

Implementation of neural machine translation methods may yield near term improvements in style transfer, and we expect that our model treats this problem as a separate task. We want to use with corpus that include parallel data because it allows us to solve similarities, and differences in nuanced between all the versions of Hebrew bibles. We hope that this project inspires the creation of style transfer architecture for Hebrew bibles.

In our project, we want to use in the flow diagram as shown in 'Approach' section for Hebrew bibles. We expect that our model will transfer the source text in another form which is close to target model, and the results for Hebrew bibles will be accurate as the results for English bibles. Furthermore, we expect that sequence (Seq2Seq) model will make a lot of changes to the source sentences and achieves the highest BLEU score.

# SOFTWARE ENGINEERING DOCUMENTATION

## Use Case



*Figure 24 – Prose style transferring use case diagram*

UC1: Make Compression -

* + - Goal - compression between source prose text to target prose text with Seq2Seq model.
    - Precondition – choose role type.
    - Possible error – select a corrupted source and target prose.
    - Limitations – select a source and target prose from repository.

|  |  |
| --- | --- |
| Actor | System |
| Click on "Start" | Verifying user select role |
| Click on "Run Model" | Open "Select Source & Target Versions" window |
| Select source prose , target prose | Open repository |
| Press "Start Running Program" | Calculate the output |
|  | Collect the proses in the repository and append them to the main corpus |
|  | Remove irregular verses and  align by verse |
|  | Split the main corpus to development, testament, and training data |
|  | Create parallel files for selected  source and target |
|  | Add target version tokens to beginning of each source verse |
|  | Learn 30000 sub words vocabulary using all versions training data |
|  | Apply vocabulary to parallel files |
|  | Train sequence to sequence model on each dataset |
|  | Find best checkpoint for each model using development data |
|  | Decode test set using best checkpoint |
|  | Undo sub word vocabulary and tokenization on output |
|  | Evaluate result using bilingual evaluation understandy (BLEU) |
| Click "Save" or "Exit" | Save / Exit |
|  | If "Save" was clicked, saving output file that contains compressions between source and target proses in the user storage |

*Table 2 – UC1 Make Compression.*

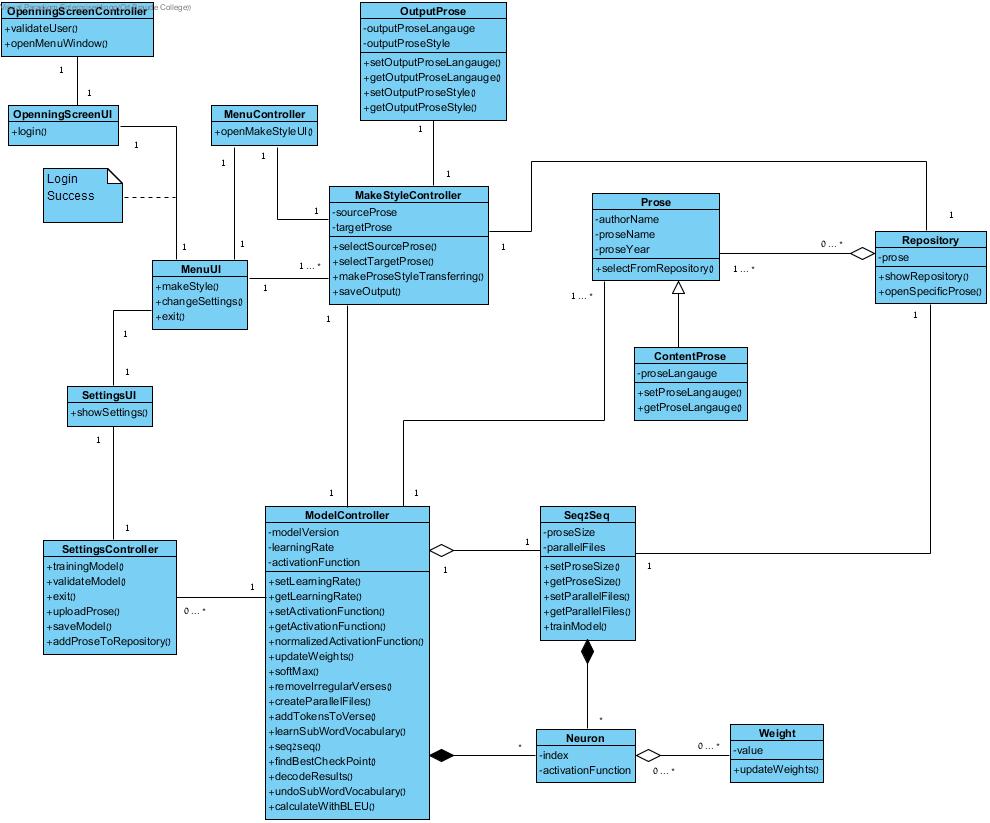
UC2: Change settings -

* + - Goal – change parameters that belongs to the model in the system.
    - Precondition – administrator login success.
    - Possible error – corrupted source and target prose.
    - Limitations – change parameters according to the range limitations.

|  |  |
| --- | --- |
| Actor | System |
| Click on "Start" | Verifying user select manager role |
| Click on "Settings" | Open "Setting" window |
| Change parameters that belongs to the model in the system | Finish saving the new parameters in the system and display message "saved successfully" |

*Table 3 – UC2 Change settings*

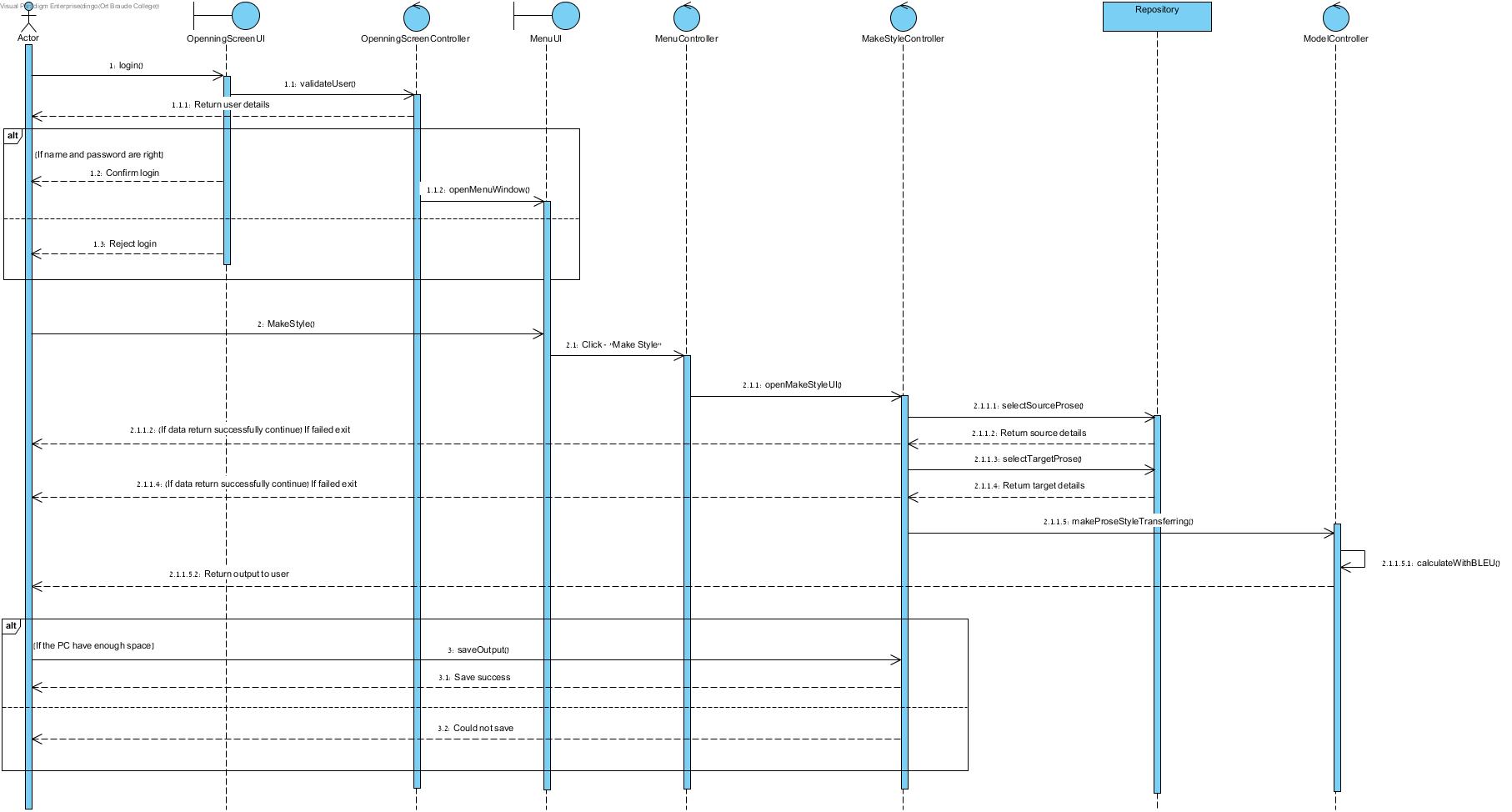
## Class Diagram



Start  
Success

*Figure 25 – Prose style transferring class diagram*

## Sequence Diagram

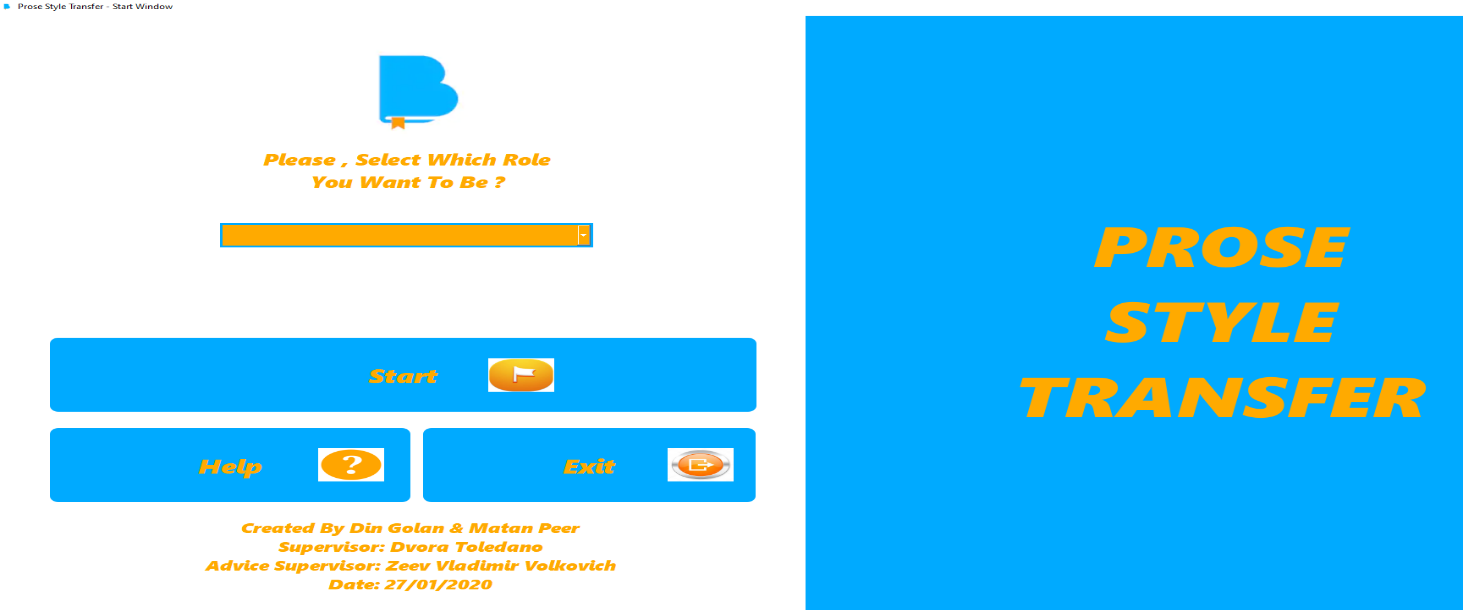


*Figure 26 – Prose style transferring sequence diagram.*

## User Interface

### Windows

Start window - The first window that the user sees when he opens the application. The user needs to select which role he wants to be in the system – regular user, manager. Also, the user can click on “Exit” button if he wants to exit from the program, and he can press on “Help” button if he wants to see explanations about the system, and this current window.



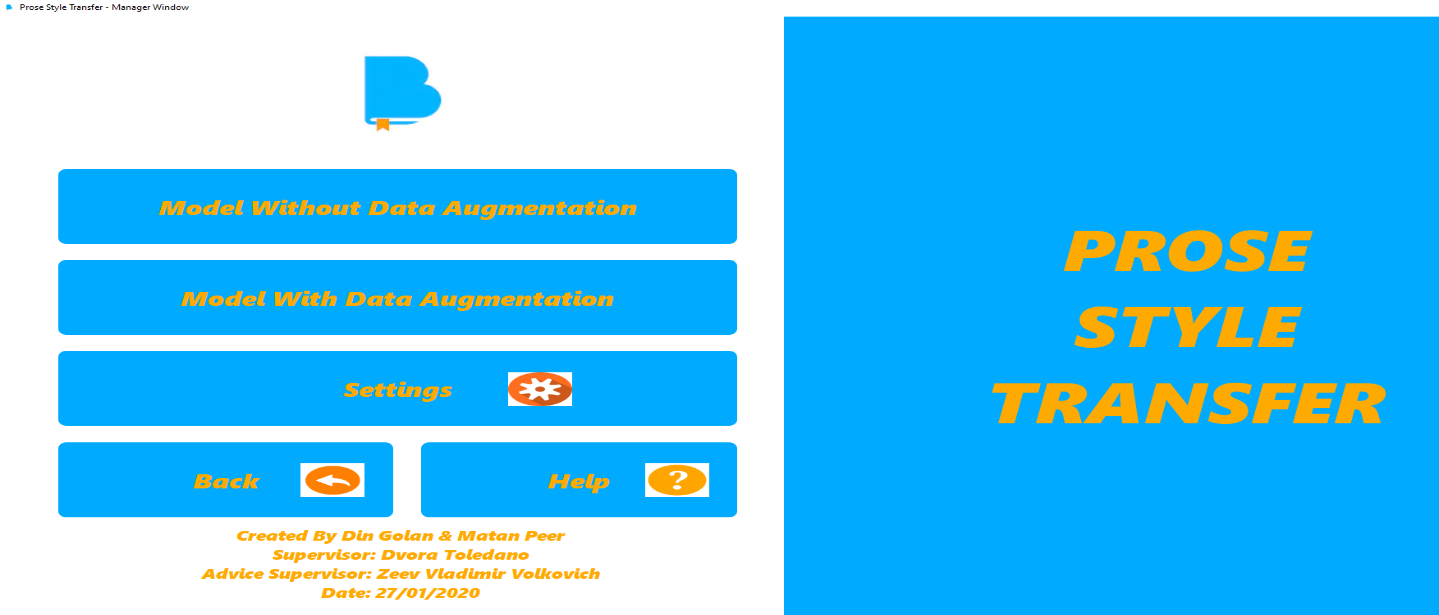
*Figure 27 - Start window*

User window - After the user decide to be regular user, the system display to him the window of regular user. In this window the user can run two types of model. First type is model without data augmentation, and second type is model with data augmentation. The differences between the models is the augmentations, it means change word to their synonyms. The model with data augmentations change the words to their synonyms, and the model without augmentations do not make these operations. Also, the user can click on “Back” button if he wants to return to previous window, and he can press on “Help” button if he wants to see explanations about the system, and this current window.



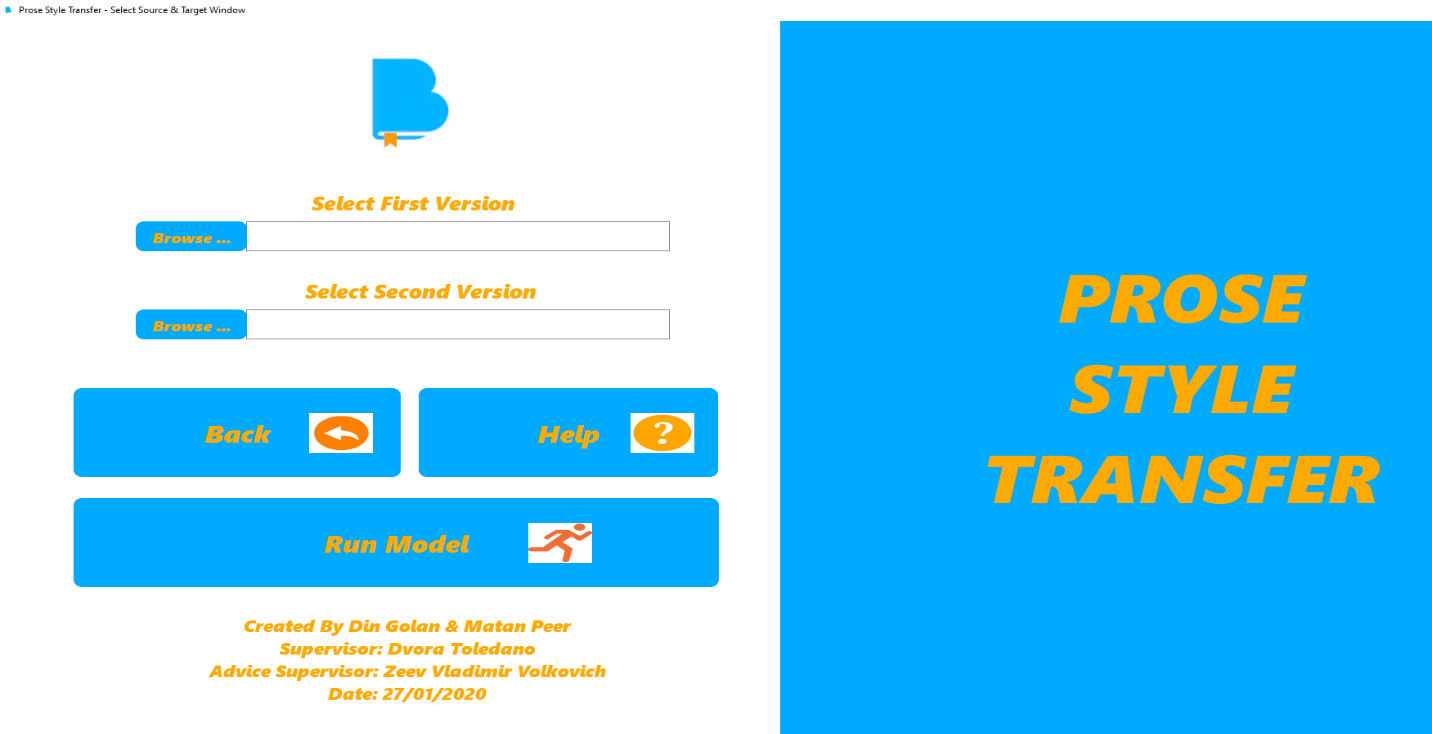
*Figure 28 - Main window for regular user*

Manager window – After the user decide to be manager, the system display to him the window of manager. In this window the manager can run two types of model like the regular user. It means that the operation of running model is similar to both roles. Also, the manager can click on “Settings” button and change some important parameters that belongs to the model. Furthermore, the manager can click on “Back” button if he wants to return to previous page, and he can click on “Help” button if he wants to see explanations about the system, and this current window.



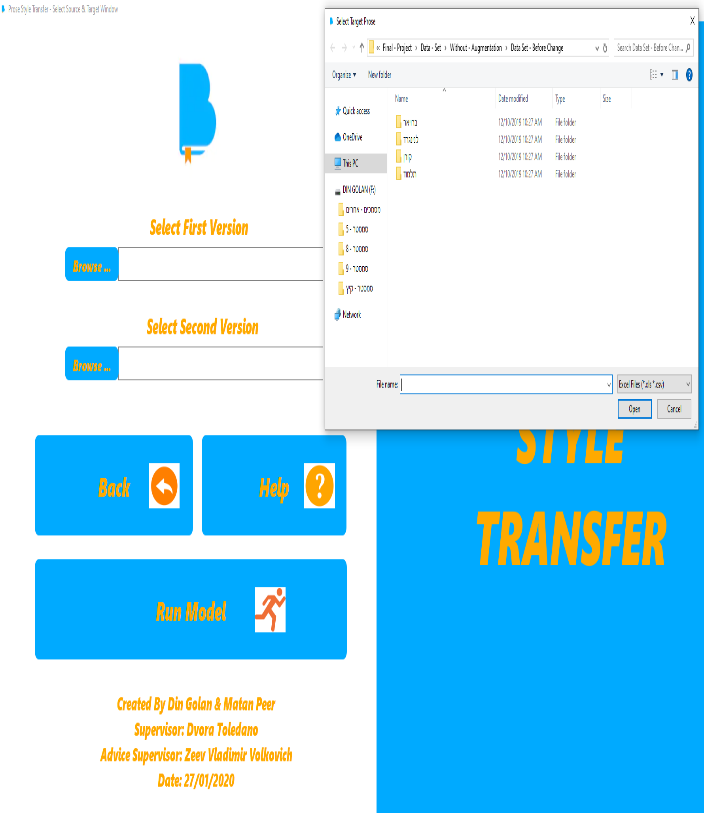
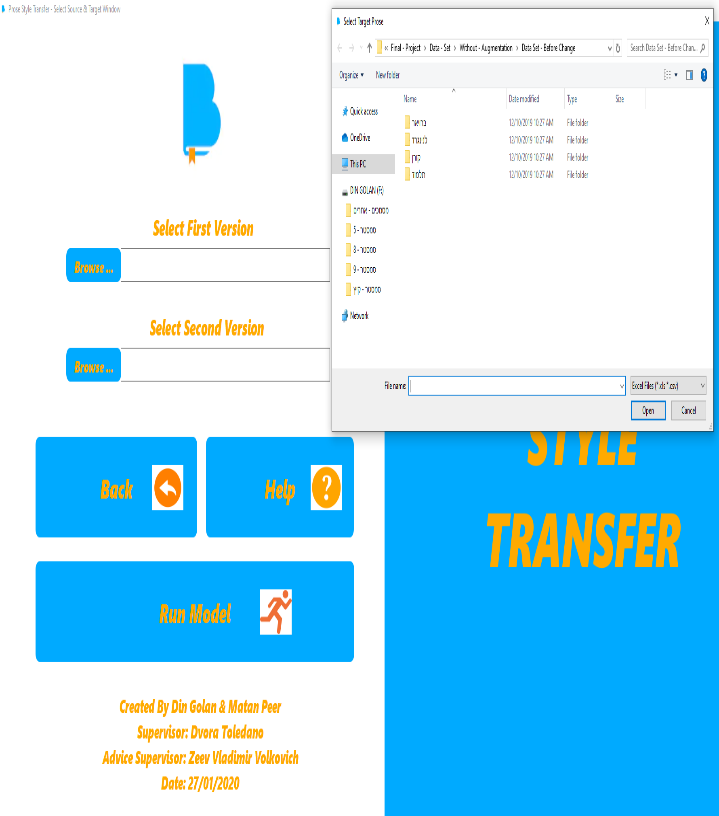
*Figure 29 – Main window for manager user*

Select source, target window – In this window the user / manager needs to select source and target prose. After the user / manager select his proses, he can run the model by click on “Run Model” button. Also, the user / manager can click on “Back” button if he wants to return to the previous page, and he can click on “Help” button if he wants to see explanations about the system, and this current window.



*Figure 30 - Select source, target prose*

The user / manager chooses source prose, and target prose from the repository and click on them, then he needs to click on the "Run Model" button.



*Figure 31 – (Left picture) the user / manager selects available source from repository,   
(Right picture) the user / manager selects target prose from repository*

Settings window – This window is available only for the manager. The manager can change important settings that belongs to the model of the system. When the manager finish to fill his parameters, he can click on “Apply” button and then the settings will save in the system. Furthermore, if the manager clicks on “Apply” button before he starts to fill parameters, he will get message from the system if he wants to use with the default parameters of the system. Also, the manager can click on “Back” button if he wants to return to the previous page, and he can click on “Help” button if he wants to see explanations about the system, and this current window.



*Figure 32 – Settings window for the manager*

Progress window – In this window we have the prediction phase. After the user / manager selects his versions, he predicts his choices with the model of the system. The user / manager needs to wait until the process of the prediction will done. When the prediction process finished the user / manager will get message from the system that he can see the results of the model. Also, the user / manager can click on “Back” button if he wants to return to the previous page, and he can click on “Help” button if he wants to see explanations about the system, and this current window.



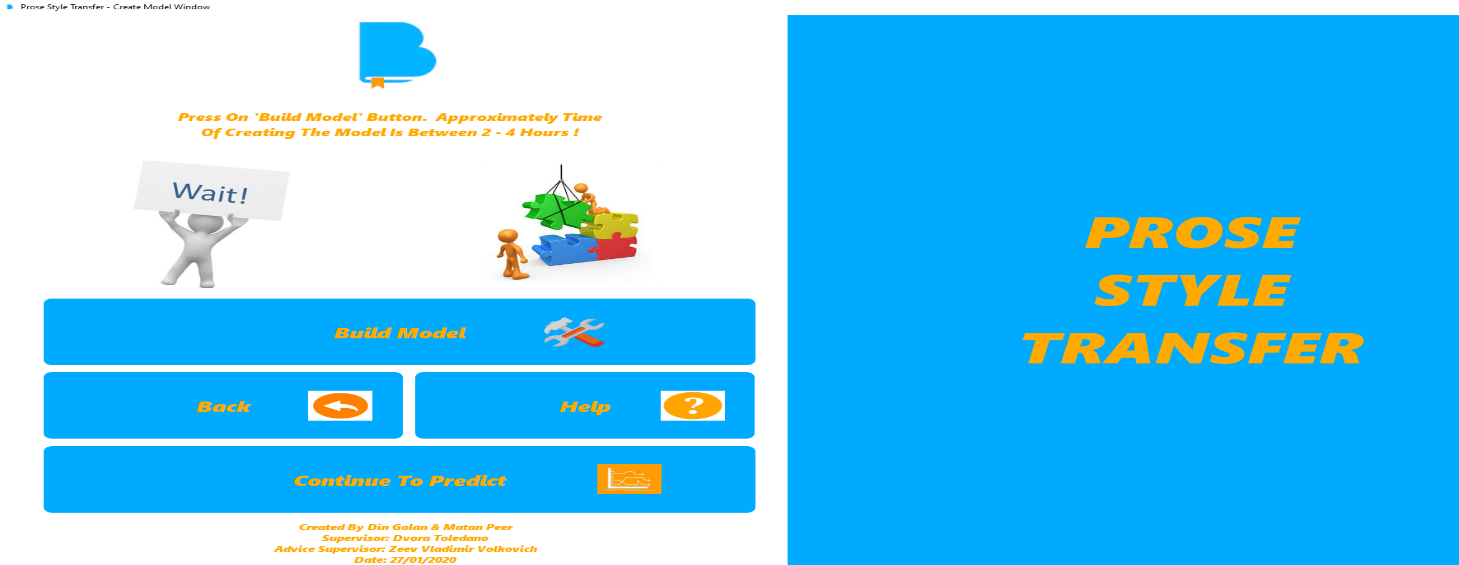
*Figure 33 – Progress window for predict phase*

Results window – The user / manager can see the results of the model. The result contains two main parts. The first part represents the accuracy between the first version and the second version that the user / manager selects. The second part represent in more details the compressions between sentences from both versions that the user / manager selects. Also, the user / manager can click on “Home” button if he wants to return to the home page, and he can click on “Help” button if he wants to see explanations about the system, and this current window. Important note – if the user is regular user and he press on “Home” button, he will come back to the “User window”, and If the user is manager and he press on “Home” button, he will come back to the “Manager window”.



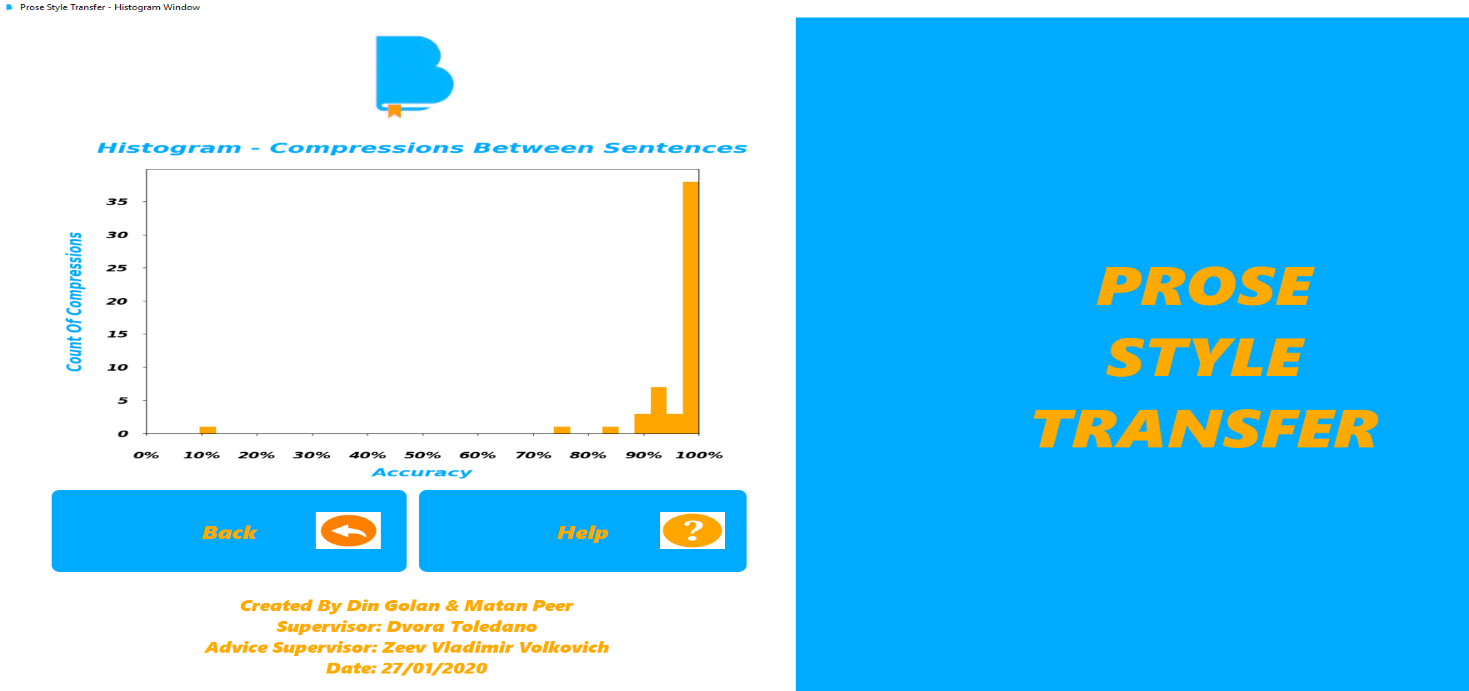
*Figure 34 – Results window, after the model finish his prediction for   
the versions that the user / manager selects*

Create model window – Only the manager can create new model for the system. If we don’t have any model in the system, the manager must create new one. If the manager wants to build new model, he needs to click on “Build Model” button. After the manager create new model, he will get a message that the model created successfully, and he can continue to the prediction phase. Also, the manager can click on “Back” button if he wants to return to the previous page, and he can click on “Help” button if he wants to see explanations about the system, and this current window.



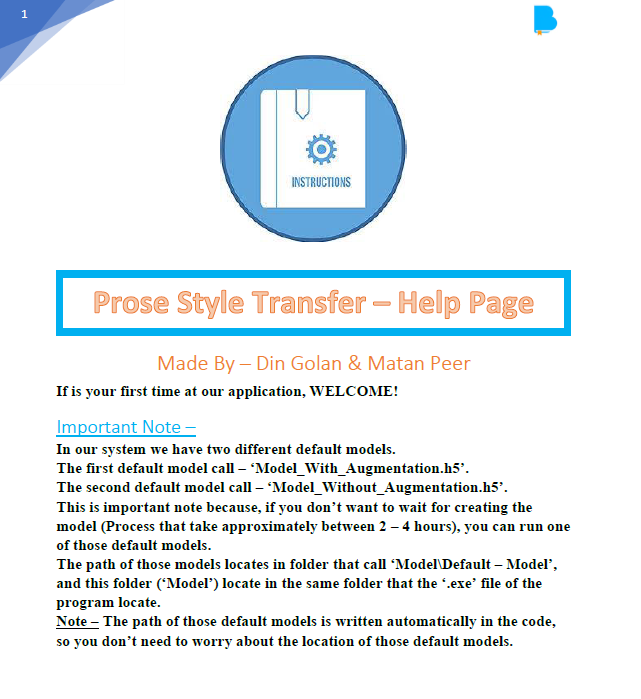
*Figure 35 – Build new model before we continue to the prediction phase*

Histogram window – In this window the user / manager can see the results of the output excel as histogram. The results at the output excel consist accuracies between sentences from both versions that the user / manager select. The histogram window represent for the user / manager the results in another way, its mean in more comfortable way.



*Figure 36 – Histogram window that represent in more comfortable way the   
accuracy between sentences that belongs to both versions that user / manager select*

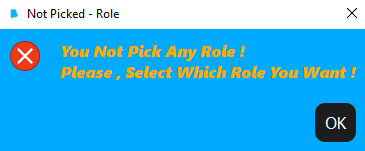
Help page – With the help page the user / manager can see all the instructions and explanations about the system. In each window, the user / manager can see explanations about each component and process in the system.



*Figure 37 – Help page that explain to the user / manager about each component and scenario in the system*

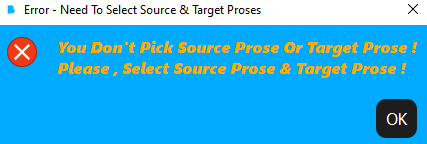
### Errors and alerts

If the user does not select any role to be in the system –



*Figure 38 – The user does not select any role – error message*

If the user / manager doesn't select prose to upload as source or target prose -



*Figure 39 – The user doesn't choose prose to upload - error message*

If the manager does not fill ant details at settings –



*Figure 40 - Message for using in default settings of the system - warning message*

If we have model in our system, the system will load it and send a message about that to the user –



*Figure 41 – The system loads the current model that exist in the system – information message*

After the predict phase finish, the system will send a message about that to the user, because after this message the user can see the results of the model –



*Figure 42 – The system sends that the prediction phase finished – information message*

## 5.5. Testing Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Test name | Description | Expected result | Actual result |
| Corrupted prose | Upload corrupted prose which cannot be open | Error | Equal |
| Missed prose when click on “Cancel” in file dialog | Press “Cancel” without select any prose | Error | Equal |
| Missed prose when running the model | Pressed on "Run Model" without selecting source or target prose | Error | Equal |
| Go back to select source & target prose | Before prediction phase, the user wants to change his prose selecting | Open select source & target versions | Pass |
| Select prose that does not exist | On file dialog, select prose that does not exist in the repository  of the system | Error | Equal |
| Go back to start window | From user window, click on “Back” and return to “Start” window | Open start window | Pass |
| Select role | Open combo box and select which role you want to be | Open user window or manager window | Pass |
| Not select role | Click on “Start” without selecting any role | Error | Equal |
| Exit from start window | On start window click on "Exit", and out from the system | Close start window | Pass |
| Save output results | When the output compressions between sentences created, save it to each folder you want | Path location in hard disk | Pass |
| Apply Settings | Fill settings details, and click on “Apply” | Pop-up window, new parameters saving successfully | Pass |
| Apply settings without filling any details | Click on “Apply” in settings window | Pop-up window, asking for using default settings | Pass |

*Table 4 – Testing plan*

# RESULTS AND CONCLUSIONS

## Results

## First example - model without augmentation and with default parameters

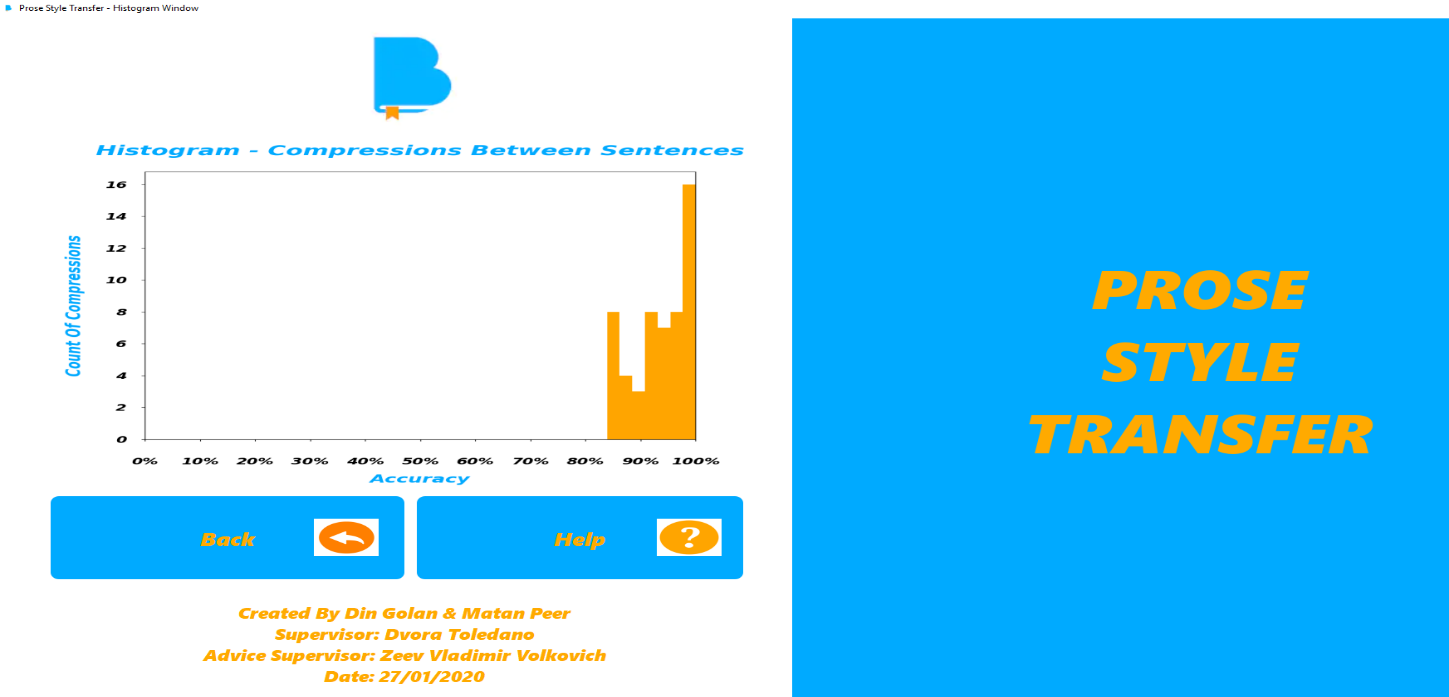
## 

## 

## *Figure 43 – Output results between two bibles versions that the user / manager selects*

## 

*Figure 44 – Results window*



*Figure 45 – Histogram window that represent in more comfortable way the results*

In this experiment we take two different bible versions and compare between them by the model of our system. The model does not use with data augmentation, its mean that he does not take some words from each version and replace them by their synonyms.

In this output result we see in more details the accuracy between sentences according to the prediction phase of the model. The prediction phase creates values of accuracy between 0 to 100. Also, the user / manager can see the results as histogram.

The parameters of the model in this example is: epochs number – 10, batch size – 64,   
training split – 0.7. Those parameters are default parameters of the system. Note - only the manager can change the parameters of the model in the system.

For summary, in this example the accuracy between the versions is approximately 67% according to the model without data augmentation.

## Second example – model with augmentation and with default parameters



*Figure 46 – Output results between two bibles versions that the user / manager selects*



*Figure 47 – Results window*



*Figure 48 – Histogram window that represent in more comfortable way the results*

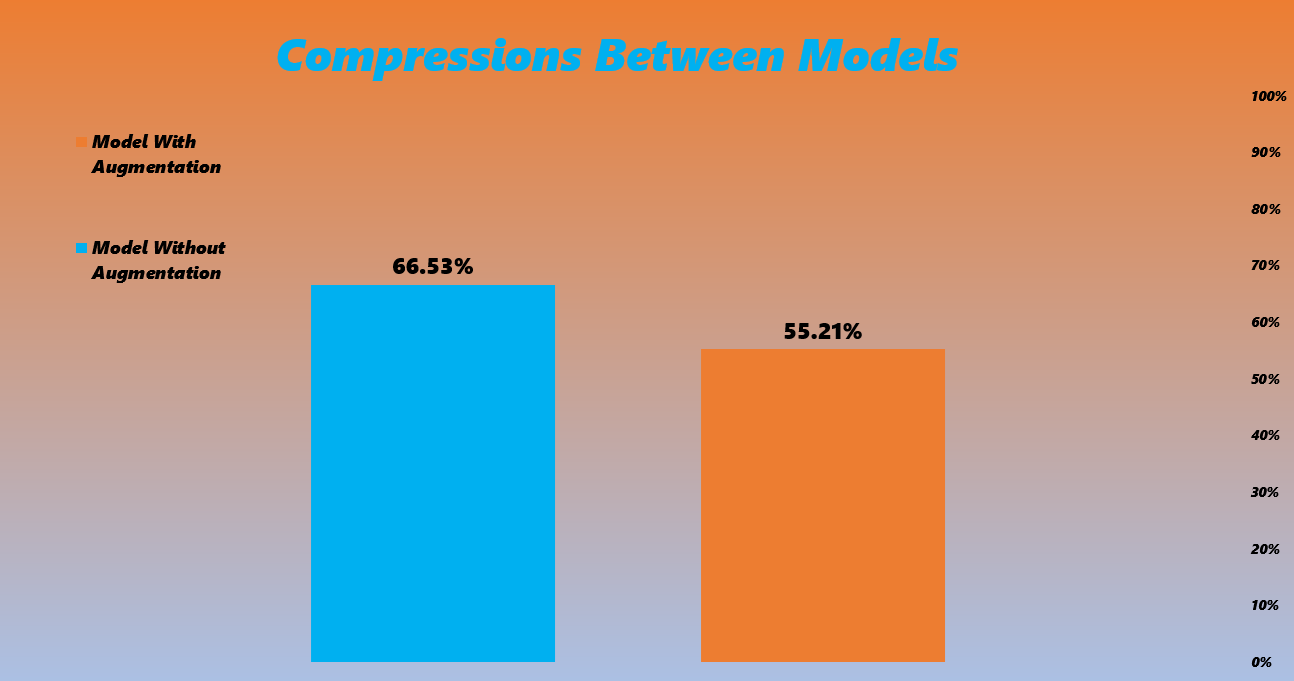
In this experiment we take two different bible versions and compare between them by the model of our system. The model use with data augmentation, its mean that he takes some words from each version and replace them by their synonyms. This operation of changing word to her synonym effects on the accuracy of the compression, its mean that the accuracy can be lower than compression between versions without data augmentation, or it can be higher.

In this output result we see in more details the accuracy between sentences according to the prediction phase of the model. The prediction phase creates values of accuracy between 0 to 100. Also, the user / manager can see the results as histogram.

The parameters of the model in this example is: epochs number – 10, batch size – 64,   
training split – 0.7. Those parameters are default parameters of the system. Note - only the manager can change the parameters of the model in the system.

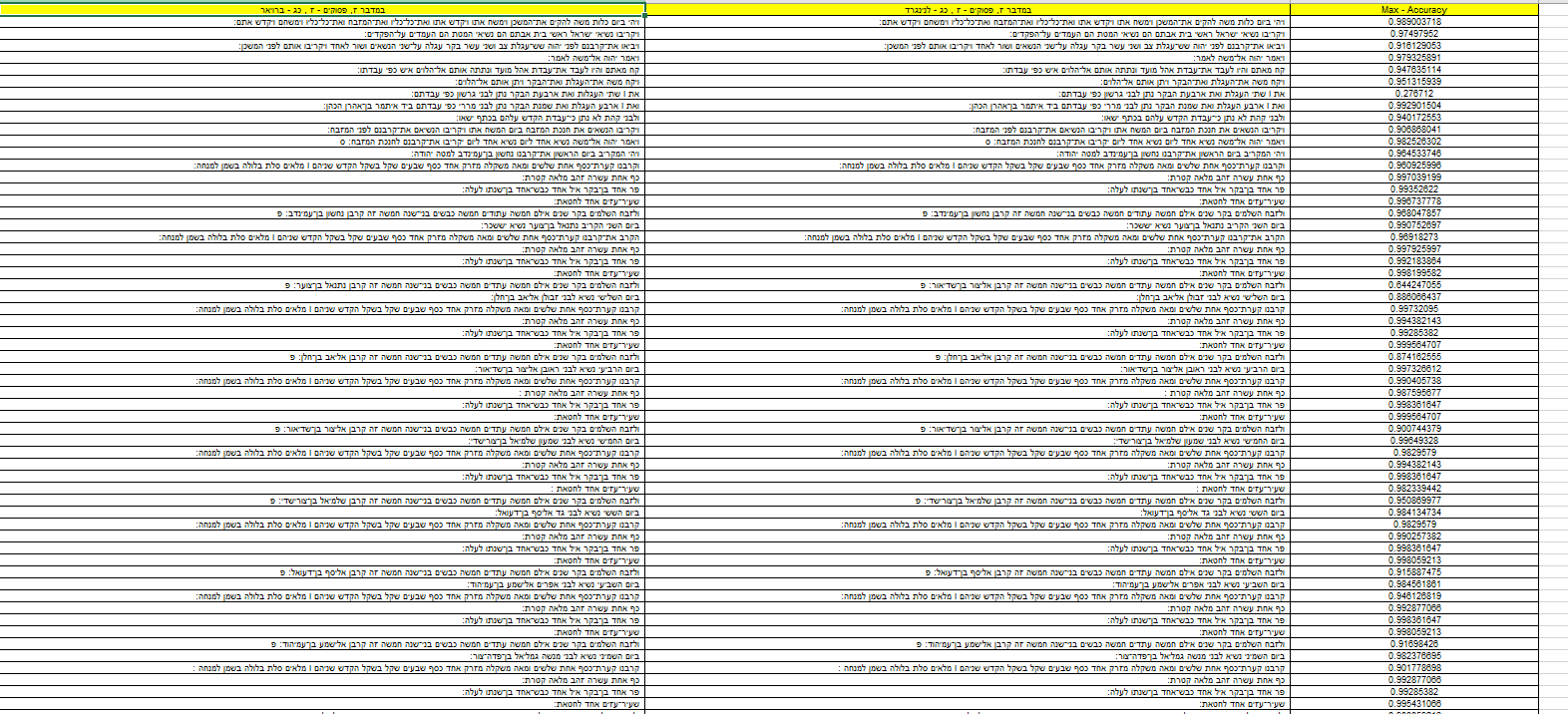
For summary, in this example the accuracy between the versions is approximately 55% according to the model with data augmentation.

## Compression between first and second examples



*Figure 49 – Compression between model without data augmentation   
and model with data augmentation*

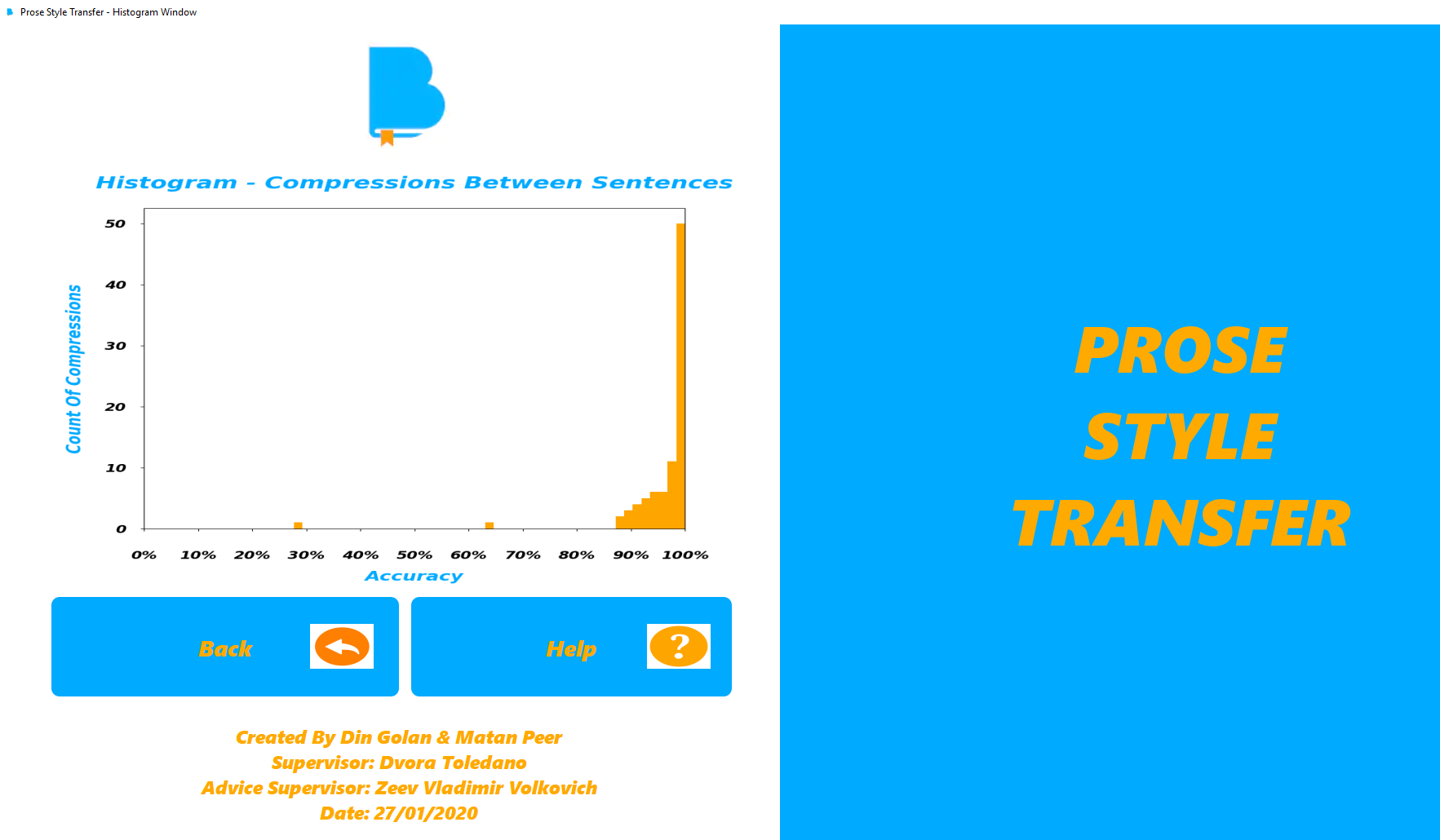
## Third example – model without augmentation and without default parameters



*Figure 50 – Output results between two bibles versions that the user / manager selects*



*Figure 51 – Results window*



*Figure 52 – Histogram window that represent in more comfortable way the results*

In this experiment we take two different bible versions and compare between them by the model of our system. The model does not use with data augmentation, its mean that he does not take some words from each version and replace them by their synonyms.

In this output result we see in more details the accuracy between sentences according to the prediction phase of the model. The prediction phase creates values of accuracy between 0 to 100. Also, the user / manager can see the results as histogram.

The parameters of the model in this example is: epochs number – 5, batch size – 50,   
training split – 0.6. Those parameters are not default parameters of the system. Note - only the manager can change the parameters of the model in the system.

For summary, in this example the accuracy between the versions is approximately 95% according to the model without data augmentation.

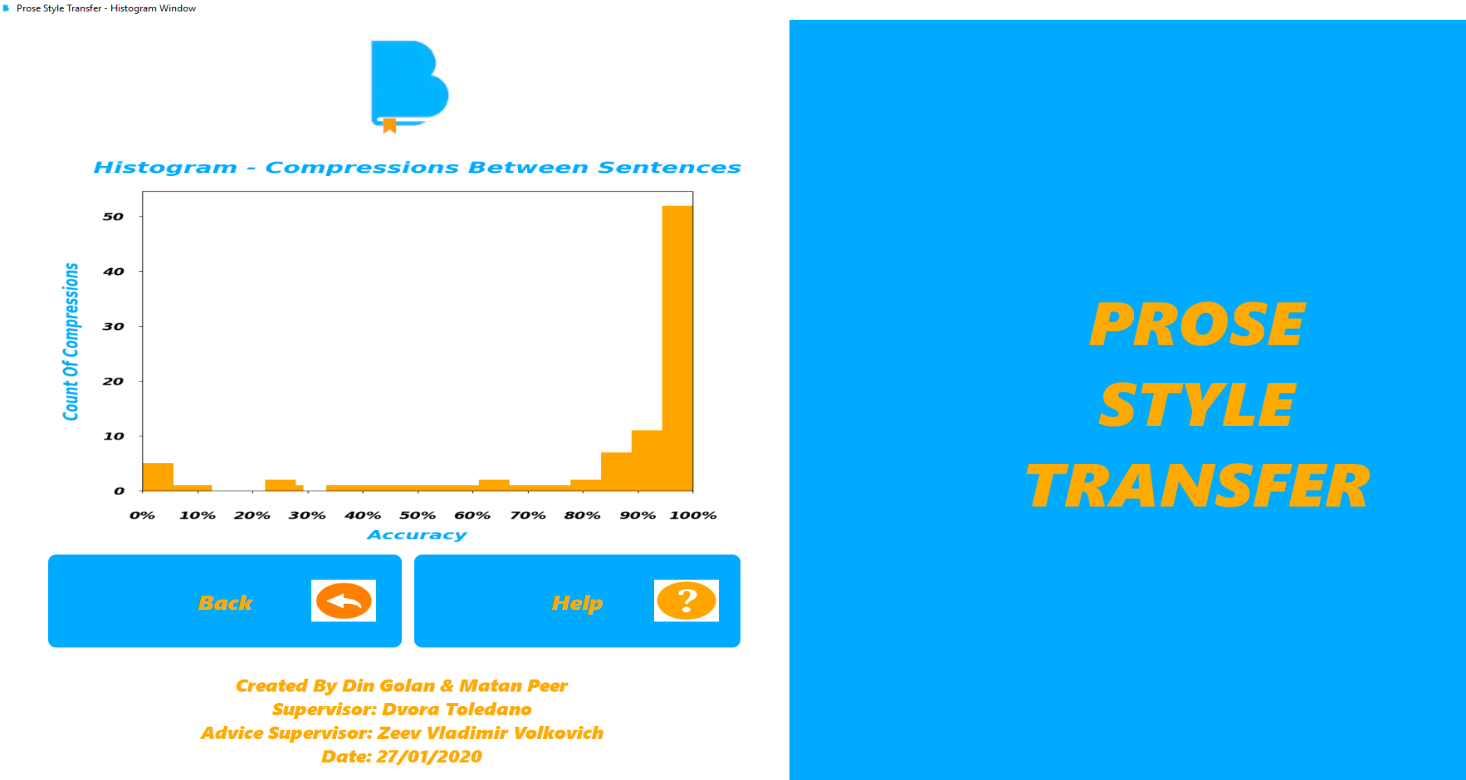
## Fourth example – model with augmentation and without default parameters



*Figure 53 – Output results between two bibles versions that the user / manager selects*



*Figure 54 – Results window*



*Figure 55 – Histogram window that represent in more comfortable way the results*

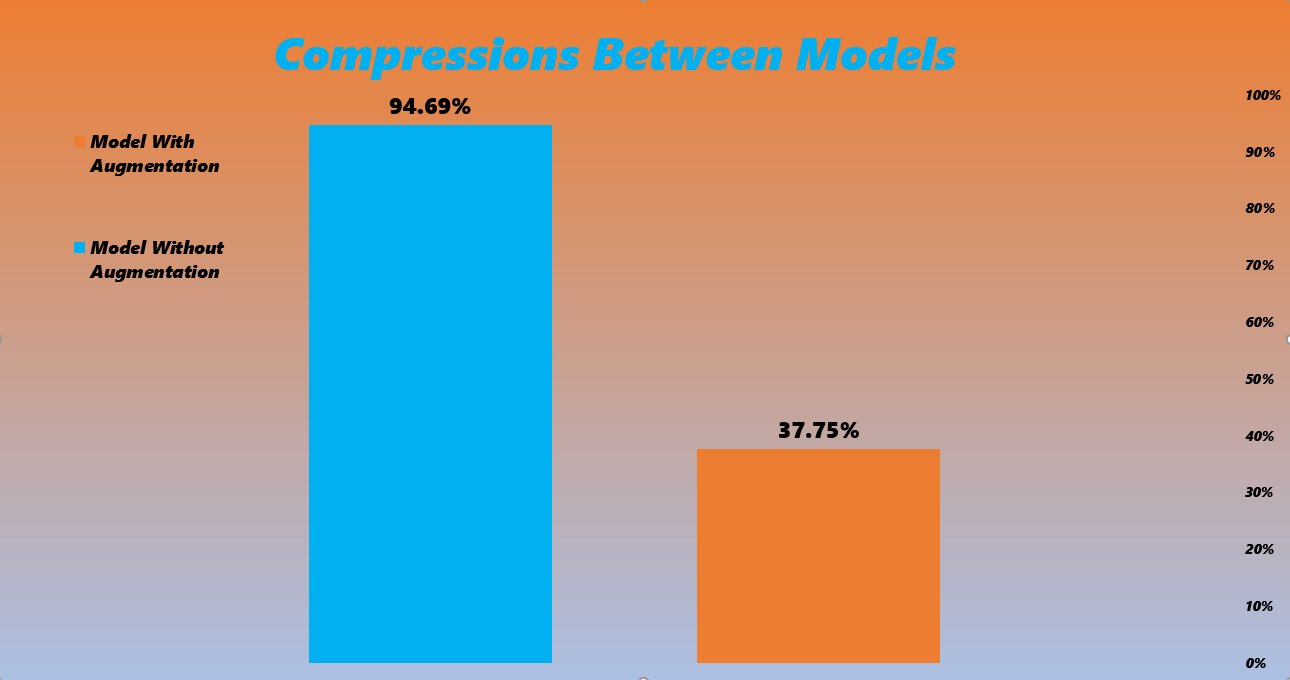
In this experiment we take two different bible versions and compare between them by the model of our system. The model use with data augmentation, its mean that he takes some words from each version and replace them by their synonyms. This operation of changing word to her synonym effects on the accuracy of the compression, its mean that the accuracy can be lower than compression between versions without data augmentation, or it can be higher.

In this output result we see in more details the accuracy between sentences according to the prediction phase of the model. The prediction phase creates values of accuracy between 0 to 100. Also, the user / manager can see the results as histogram.

The parameters of the model in this example is: epochs number – 5, batch size – 50,   
training split – 0.6. Those parameters are not default parameters of the system. Note - only the manager can change the parameters of the model in the system.

For summary, in this example the accuracy between the versions is approximately 38% according to the model with data augmentation.

## Compression between third and fourth examples



*Figure 56 – Compression between model without data augmentation   
and model with data augmentation*

## Conclusions

At the experiment stage, we have created many models with different parameters, trying to understand better how the model responds to the various parameter’s values. We learned at the section of learning process that small epoch number gives quite high accuracy and on the other hand, a relatively big epoch number causes over fitting. After deep thinking, we concluded that a different threshold value could contribute to various purposes of the application.

In the future, there is an option to expand the data set of Hebrew bibles, which can assist to improve the classification results. In addition, considering a possibility of a new Hebrew bible version, we suggest to repeating on the training process periodically.

We hope that our program will help to investigate the hypothesis that there have places in the bible that not written by human being.

# REFERENCES

[1] Carlson, K., A. Riddell, and D. Rockmore, *Evaluating prose style transfer with the Bible.*   
Royal Society open science, 2018. **5**(10): p. 171920.

[2] Lopez, M.M. and J. Kalita, *Deep Learning applied to NLP.* arXiv preprint arXiv:1703.03091, 2017.

[3] Cho, K., et al., *Learning phrase representations using RNN encoder-decoder for statistical machine translation.*   
arXiv preprint arXiv:1406.1078, 2014.

[4] Kotsiantis, S.B., I. Zaharakis, and P. Pintelas, *Supervised machine learning: A review of classification techniques.* Emerging artificial intelligence applications in computer engineering, 2007. **160**: p. 3-24.

[5] Alpaydin, E., *Introduction to machine learning*. 2009: MIT press.

[6] Collobert, R. and J. Weston, *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.*

[7] Deng, L., *A tutorial survey of architectures, algorithms, and applications for deep learning.*   
APSIPA Transactions on Signal and Information Processing, 2014. **3**.

[8] Zaremba, W., I. Sutskever, and O. Vinyals, *Recurrent neural network regularization.*   
arXiv preprint arXiv:1409.2329, 2014.

[9] Luong, M.-T., et al., *Multi-task sequence to sequence learning.* arXiv preprint arXiv:1511.06114, 2015.

[10] Abadi, M., et al. *Tensorflow: A system for large-scale machine learning*. in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016.

[11] Papineni, K., et al. *BLEU: a method for automatic evaluation of machine translation*. in *Proceedings of the 40th annual meeting on association for computational linguistics*. 2002. Association for Computational Linguistics.

[12] Jhamtani, H., et al., *Shakespearizing modern language using copy-enriched sequence-to-sequence models.*   
arXiv preprint arXiv:1707.01161, 2017.

[13] Britz, D., et al., *Massive exploration of neural machine translation architectures.*   
arXiv preprint arXiv:1703.03906, 2017.