RankIT!

your best automatic friend!

# Rank IT!

BS-SE-20-49

An automatic metric for summary scoring

Lior Reznik

Under the supervision of Dr. Natalia Vanetik

SCE

המכללה האקדמית להנדסה ע"ש סמי שמעון

מהנדסים לעולם טוב יותר!

בסביבת PROJECT ORIENTED

## Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Dr. Natalia Vanetik for the continuous support of my study and research. For her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research coding and writing of this project.

Besides my advisor, I would like to thank Dr. Marina Litvak, the head of the Data Science track, a track that gave me meaningful knowledge in the ML and data science world. The knowledge that helped me a lot in this final journey towards my B.Sc.

Finally, I could not complete this project without the knowledge that I have gained in the last four years of studying in SCE. For this, my gratitude goes to the Head of Department Dr. Karim Abu Affash and all the academic staff of the department.

Lior Reznik, RankIT!

## Abstract

Today in the era of information, information is everywhere. We are overwhelmed by it, in this era, we can get every single piece of information from the internet: we can find books, articles, news. But how can we obtain all this information, how can we get only the essential pieces from a given text?
The answer is to summarize the text; in other words: condense the text into a smaller representation that expresses the primary keys, facts, and ideas fluently.
Fortunately, there are plenty of automatic summarizers that can do the trick.
Automatic summarizers aim to produce a shorter but concise version of the original text.

The summarization process is not trivial. To produce a good summary,
the summarizer needs to understand the text, its semantics, and main ideas. Eventually, it needs to produce an easy to understand, coherent piece of text.

There are two main summarization evaluation methods, intrinsic and extrinsic.
In the extrinsic methods, the summary gets its score regarding some tasks; this type of evaluation is useful for goal-oriented summaries.
In the intrinsic methods, the summary directly analyzed. The intrinsic methods may involve a direct comparison with the original text(by measuring the coverage of the main topics) or by the closeness between the summary and some golden standard summaries that are written by human-experts.
The problem with the intrinsic methods that rely on golden standard is that they require much workforce and lack consistency.

In this project, I built a system for automatic intrinsic evaluation of English summaries.
I have used a supervised approach in which I have trained deep learning models on pairs of (articles, summaries) and a corresponding score in 0-1 range.
I have taken 16,000 articles from the CORD-19: The Covid-19 Open Research Dataset[37].
I have generated the summaries and the scores in the following way:
(1)Abstract of the article as 1 score (gold standard).(2) Random permutation of the summaries as 0 scores. (3)Concatenation of the first abstract half with noise from news domain, zeros, or duplication of the first half.
I have tried the following deep learning models:
(1)Siamese network of two bi-directional LSTMs (with and without attention) [9] [11] [13] [16],
(2)Siamese network of multi-channeled CNN[14] (to try and capture multiple local contexts),
(3)Siamese network of LSTMs on top of the Siamese network of CNN (to try and capture local contexts alongside with global ones) and (4) the best of the above but with two parallel networks: one for the score, second for the penalty(on duplicated content for Example).

I decided to use MSE cost function as my metric because the domain of the problem and because I wanted to give "high penalties" to higher errors.

With 0.0035 (MSE) on the train set, 0.0247 on the dev set, and 0.0210 on the test set, the LSTM with attention was the best. The results are not bad, but there were two main problems:(1) The network failed to understand when the summary incomplete. (2) our model overfitted! I have tried to deal with the first problem by duplicating the network into one that gives a score and one that gives penalties, this approach needs further analysis.

**Keywords:** Deep Learning, Siamese Network, Metric Learning, Summarization Evaluation.

# Introduction

## Problem definition

Today in the era of information, information is everywhere. We are overwhelmed by it. In this era, we can know what is happening on the other side of the globe just by entering into a news site, we can read about the France revolution or on every other piece of history, we can read some of Shakespeare's poems or some of Machiavelli's writings.

We can get every single piece of information from the internet, and all of that without leaving the couch. I can even get a scientific paper in the same way.
For Example, let us look on the following data-sources:(all the numbers taken from the data-sources itself):
1. Wikipedia (the open encyclopedia) has 40 million articles in 293 different languages. [1]

2. Liinwww (The Collection of Computer Science Bibliographies) contains more than 7 million references to journal articles, conference papers, and technical reports. [2]

3. Arxiv.org (open archive for scholarly articles in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering, and systems science) has 1,632,144 e-prints. [3]

4. Aminer (Knowledge Intelligence) has 270,473,189 publications. [4]

5. Academia (platform for academics to share research papers) has 24 million papers. [5]

One can argue that we are living in a magnificent time, it is true but, this new era holds some challenges.

One of these challenges is the question of how we can obtain all this information or in other words, how can we summarize an essential piece of information?
In superficial view and without getting into specifics, summarization is not a piece of cake, it requires a dissent understanding of the text, it's main facts, and ideas.
The second challenge is how to estimate the summary, how can we know if the summary fits the peace of information.
Today, there are a lot of automated tools that are doing the summarization task; the problem is that there are not enough useful and full automated tools that check the summary. Some tools depend on reference summaries, but these reference summaries are not so easy to find.

## Solution

The solution is a good, fully automated machine that scores a summary concerning the article without any reference summaries or any other piece of information.

## Goal

The goal of this document is to cover summarization techniques, related scoring techniques and works, explain the methodology that applied, tools that I  have used, to show experiments and results and future work and ideas.

## Literature Review

### Technical and theoretical introduction

Before we dive into the summarization world, we need to understand some basic technical and theoretical concepts such as Neural Networks and Natural Language Processing (NLP).

### Artificial Neural Network (ANN) [6][7]

An artificial neural network is a computational model used in machine learning and inspired by the human brain.

When the data flow through the ANN, it learns the features of the data and adjusting itself to generate an excellent solution to the problem.

While the "traditional" programming is all about providing input and algorithm to get output, a neural network is all about feeding the network with input, output, and problem to get the algorithm that is good for solving this type of problem.

Mathematically speaking, a neural network is a function that maps an input to an output.

Every pass between neurons called a signal; there might be more than one signal that connects to a neuron.

Each neuron gets on it turns the signals, processes them, and moves them forward (aka forward propagation).

When the output of the last neurons layer generated, the network starts a reverse process, without going into the math, the network goes back and tries to tune the weights of each signal for each neuron this process is called backpropagation.

Each neuron network composite from:

1. Input layer.
2. At least one hidden layer.
3. An output layer.

Each layer is a collection of neurons.

Every one of the layers makes transformations on their input; our primary focus is on RNNs, CNNs Embeddings, and GCNs.

#### *Embedding Layer [7]*

The embedding layer task is to do the Word Embeddings.

Word Embeddings and Embedding layer are all about mapping the words into the latent space, space in which the similar points are closer to each other.

The main advantage of the embedding layer over pre-trained embeddings is that the weights of each word vector are getting updated continuously as the ANN trains.

### *Graph Convolutional Networks (GCNs) [8]*
GCNs is a type of NN that operates directly on graphs, firstly introduced in 2014.

This network performs convolution in graph domain rather than in the Euclidean domain, which makes the work much harder.

GCN used for feature extraction from each node; the final goal is to encode the features of the graph into lower dimensionality representation.

### *Recurrent Neural Network (RNN) [7] [9] [10]*
We can think about the RNN networks as the learning process of the human being itself.

When we come to a lecture, we do not start all the learning process from the beginning; instead, we are coming with our previous knowledge, knowledge from previous lectures, or even courses.

Infect, we can say that this layer has multiple copies of itself; every cell sends a copy of the information to his son.

In other words, we may consider RNN as cells that have a "memory," which captures or memorizing the information of what has been calculated until now.

We can imagine an RNN layer as a layer that has an internal loop, a loop that gives to data the ability to flow from one step to the next one.

If we untie the loop, we get a sequence that looks like a linked list or array. It is the main reason why RNN is a suitable architecture for text.

RNN is a good idea, but unfortunately, it is not flawless,

Theoretically speaking RNNs can memorize a long arbitrary sequence but in real life, in practice:

1. They are limited to the immediate past only.
2. They have the problem of vanishing an exploding gradient (one of the biggest problems in neural networks), which makes the network unstable.

The solution to this problem is to use LSTM or GRU cells.

### *Long Short-Term Memory(LSTM) [9][11]*
It introduced in the year 1997 by Hochreiter and Schmidhuber.

This type of cell became very popular because it designed specifically to avoid the long-term dependency problem.

LSTM  forms of a chain (as RNN), but unlike RNN, the cells of LSTM are more complicated; while in RNN, we have only one tanH gate.

In the LSTM we have 4 new gates and one more line of data called cell state, without getting into specifics, the LSTM has this new gates and line of data because it gives him the ability to decide which data is relevant for keeping and which not.

### *Gated Recurrent Unit (GRU) [12]*
GRU is a simplification of LSTM, GRU is the newest generation of RNN. It is pretty much like the LSTM architecture, GRU got rid of the cell state and returned to the only state, the hidden state that transfers the information forward throw the layer.  Also, it has only two gates: reset and update.

### Bidirectional GRU/LSTM/RNN [13]

The difference between the Bidirectional layer and the standard one is the direction.

In the standard layer, the data flow direction is forward, while in the bidirectional is forward and backward.

The bidirectional layer is two independent layers stacked together. The input is fed in standard time order (forward) for one network, and in reverse time order for another (backward), the output of the two networks concatenated at each timestamp.

This architecture is good because it can "see" the past and the future at each step of the way.

#### *Convolutional Neural Network(CNN) [14]*

This type of architecture has proven to work well with images ( such as object recognition, image classification, and computer vision).

There are 3 main components in CNNs,( Convolution, Non-Linearity, and sub-sampling) but we are interested only in the first one:

Convolution: this is the primary operation of CNN; in this operation, CNN tries to extract features from the input image by finding local relationships between the pixels in the image. It does this by applying a small 2d squared filter on the image.

But as it turns out, we can apply CNN on the text as well; the only difference is that we are using a 1d filter instead of 2d.

Applying convolution on the text can help us in finding local contexts between words (like n-grams).

## *Sequence to Sequence Model[15]*

A sequence to sequence architecture is getting more and more popular over the years. This type of architecture lies behind numerous systems such as:

- Summarization.
- Translation.
- Video captions.
- Voice recognition, such as Siri, Alexa.
- Question answering.

Google introduced this model in the year 2014. The advantage of this model is the ability to map an input sequence to an output sequence when their length is not the same.

it is essential in tasks like summarization were the output is most likely to be shorter than the input or in tasks like translation, let us look at the following Example:

Translating "Where will you be tomorrow night?" from English to Hebrew has an input of 6 words and output of 4 words (איפה תהיה מחר בלילה?).

For this type of task, we cannot use a regular RNN network.

This model based on LSTM/GRU cells and compound from 3 main parts:

- Encoder
- Latent Space Vector
- Decoder

### Encoder

The encoder is just a stack of several GRU or LSTM cells that gets a sequence as input, collects information about the sequence, and propagates it forward.

### Latent Space Vector

The latent space vector is the final hidden state produced from the encoder. This vector aims to encapsulate the information from all the input elements.

This vector is the initial hidden state of the Decoder.

In general, we can explain the latent space vector in the following way:

We are living in a word full of layers; we cannot see all of them, many of them hidden from us; there are a lot more than meets the eye. We want to learn about the hidden layers; we want "to see" them; to do that, we can take the information and map it into the latent (hidden) space, a space that a similar data points are closer to each other.

In other words, the latent space is space were the features lie.

In the summarization task, we need to find the main topics, these topics lie in the latent space, the meaning is not immediately apparent to the eye, but can be discovered more quickly in the latent space.

### Decoder

The Decoder is just a stack of several GRU or LSTM cells (it can be differing from the Encoder type) that decodes the information; the decoding process depends on the task.

## *Attention[16]*

RNNs on its later forms (LSTM and GRU) still have a problem with long-term dependencies. The truth is that they fail with them, the idea that the last cell (latent space) can hold the information of all the sequence is not working well (at least not all the time).

One of the problems that cause RNNs to fail with such dependencies and sentences is the "vanishing gradient problem."

The core idea of the attention mechanism is that. "not all tokens are burn equal," meaning not all the tokens in the sequence have the same importance for the understanding of the sequence.

The attention mechanism tries to learn the amount of attention that each hidden state needs or the importance of each state.

## *Transformers[17]*

Encoder-Decoder architectures that based on LSTMs with attention mechanism tends to perform well, but still, there are some problems with this type of architectures the main one is that they are complicated, they are time and space consumers.

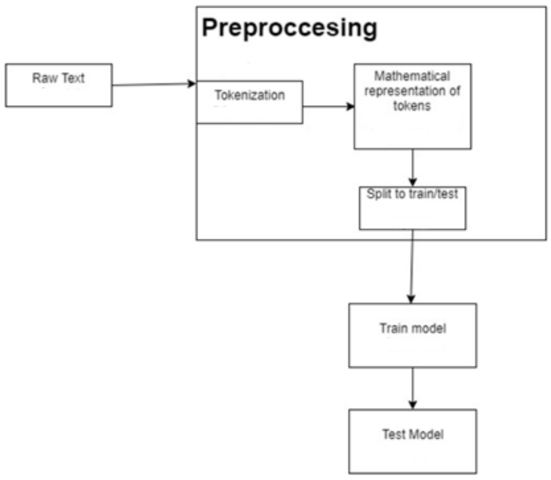
The transformers architectures argue that we can get rid of LSTMs and build suitable architectures that based only on the attention mechanism.

## Natural Language Processing (NLP) and Data Pre-processing

The system that we desire to build is a Natural Language Processing (NLP) system; in this section, let us understand the meaning of NLP and how we can build such a system. [18] [19]

Well, NLP is a subfield of AI and linguistics, this subfield investigating the connection between computers and human languages. In other words: NLP tries to find a way to "teach" the computer how to understand and interact with human language.

The pipeline of a typical NLP system:



The first step of every NLP task is the preprocessing. This step is one of the more time-consuming steps there is.

## Tokenization [7][19]

The first step of every preprocessing task is splitting the text into tokens, or more accurately speaking to smaller but still meaningful units of information. If we think about it, a text is a sequence of chars/words/sentences/paragraphs and even documents.

### *Sentence Splitting and boundary detection*

Usually, the first step is tokenization to the sentence level. Meaning, taking the raw text and breaking it into smaller but still meaningful pieces of information, called a sentence. A sentence is a unit consisting of one or more words that are linked together.

### *Word Level Boundary Detection*

The second step of the preprocessing is splitting the sentence into even smaller, and meaningful units called words.

The step of word-level tokenization may seem easy, but this is not true at all. To understand why, first, we need to understand what is the meaning of the word "word," what is this piece of information.

This question is not as trivial as one can think, and even Linguistics researchers struggle with this question.

One can argue that a word is a sequence of chars that separated only by spaces.

Well, this definition is not so accurate because, by this definition, one can argue that she, she? , and she! is three different words!

We may try and tune this definition by arguing that a word is a sequence of chars that separated by spaces and punctuations.

Looks about right? Well, no. This definition may be true for the English language (except some edge cases) but for other ones not so much for Example:

- In  Japanize and Chinese languages, there are no spaces at all.
- In Hebrew, Arabic, and German, some words are not canonical.

Lucky for us, some shelf products doing the job for us.

### Token Normalization[7]

Sometimes we want to get the same token for a different set of forms of the same word. We want to do this for dim reduction.

For Example, talk, talked, talking, and talks.  Might have the same token talk.

For this, there are two techniques that we can use:

#### *Stemming*

Stemming is the process of removing and replacing of the word's suffixes; we can do it by getting the stem(root) of the word. Usually, to do this, we use heuristic algorithms and regex.

#### *Lemmatization*

Lemmatization is the process of returning the lemma of the word, meaning returning the base (dictionary form) of the word.

We can do it with the help of dictionaries and morphological analysis.

## Mathematical Representation of Tokens

As shocking as it may seem, our computer does not know Hebrew, English, or every other human language, the only language that the computer does know it is the mathematical language.
The computer does not know the meaning of the words, does not know the semantical relationship between words.
Because of this fact, one of the most critical steps in the NLP pipeline is understanding how to represent the raw input mathematically.
Unlike pictures were the representation is clear( two- or three-dimensional matrix), in-text we have ambiguous ways to represent the input.

### *Classical Models*

These models are classic and outdated, but still, we can find them in industrial use.

#### One Hot Encoding [20]

One hot encoding maps every word into a unique ID and then translates that ID into a binary sparse vector, the unique ID than represented by the index that holds in his cell 1, the other cells are zero.

The size of the vector is the size of the vocabulary.

For Example: assuming we have a dictionary sized 5 (meaning, our text consisted only with 5 unique words) and we have a word with ID 3, the vector for this word is [0,0,1,0,0].

Formally, every word is a binary vector that belongs to $w \in \mathbb{R}^n$ vector space.

This model has serval disadvantages:

- The model is naïve and does not try to find ties between the words.
- We must have a pre-defined size for the dictionary.
- If the dictionary gets too big, it cannot fit in the main memory.
- The vector is sparse (may cause Overfitting and too long training).
- We must ask the following question, how we treat words that we never saw before?

#### Bag of Words (BoW) [21][22]

Bow tries to represent text as a vector. However, unlike the one-hot encoder, it tries to avoid sparse representation by assigning the number of appearances of each word in the text to its corresponding cell in the vector.

We can think of BoW as the sum of all One Hot Encoding vectors.

This model has serval disadvantages:

- This model fails to see the semantical and syntactical ties between the words.
- This model loses the word order.
- The model fails to see the similarity between the words.
- The model cannot handle words that do not appear in the vocabulary.

## N-gram [21][22]

Let's assume that we are reading the following sub-sentence "This is a … "and we need to guess what is the next word if we have the following options: {good, did} we as humans will guess "good," because we know the language and because it seems right.

But the computer does not know this, well N-gram is all about that.

The question below is probabilistic: p (good | This is a) =? P (did | This is a)=?

And this we can teach the computer.

Let's take a second and try to understand how we know that the following word should be "good" and not did; it is because we know that P (good | This is a) >P (did | This is a)? Well, yes and no, not a single person uses statistics for this, we know the following word because of our history, because of the books we read and the movies we saw, because all of that we can know what the suitable word is.

The same goes for a computer; by seeing a list of words that frequently used together, the computer can learn and predict which words should follow a specific word.

N-gram model is an N sized window; we are dividing the text corpus into N sized windows and storing them in the vocab.

This model has serval disadvantages:

- This model fails to see the semantical and syntactical ties between the words.
- This model partly keeps track of the word order.
- The vocab is too big.

## Term Frequency-Inverse Document Frequency (TF-IDF) [22]

This model tries to capture the importance of the word to the document. Intuitively speaking, the importance of a word is greater if and only if it appears a lot in the document but barley at the corpus.

There are some ways to calculate the TFIDF, but most fundamental is:

$$\mathrm{TF-IDF(t) = TF(t)*IDF(t)}$$

$$\mathrm{TF(t)} = \frac{\text{Number of times term t appears in document}}{\text{Total number of terms in the document}}$$

$$\mathrm{IDF(t)} = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}\right)$$

This method  based on the BoW method, because of it, this method cannot capture:

- semantical and syntactical ties between the words.
- The word orders.

### *Modern models*

The common thing of all these models is the ability to understand semantics.

In these models, tokens with the same or close meaning represented as similar vectors. We are focusing on word and sentence models, but there are serval other models, such as the char model and document model.

### Word Embedding

a trendy set of models that maps words from the vocab into numerical vectors. These models help the computer "understand" the semantic and syntactical similarity between words and the context of the word regarding the document.

The word represented by a dense vector; were similar words represented as similar vectors.

The next question we should ask ourselves is how we can create those vectors?

### Word2Vec [23]

At the year of 2013, Mikolov, a researcher at google labs, suggested a new model. This model is one of the most popular models up to this day.

This model using an unsupervised shallow neural network with one hidden layer.

There are two different model architectures which can be leveraged by Word2Vec to create these word embedding representations:

#### *Continuous Bag of words (CBoW)*

The input of this architecture is a bunch of words (vectors), and the output is one word (as a vector). This architecture tries to understand what the word is by looking at its surroundings; we can look at it as "tell me who is your friends and family and I will tell you who you are."

#### *Skip Gram*

Skip Gram is the opposite of CBoW architecture; in this model, the input is one word, and the output is the surrounding words of this word.

According to Mikolov, those to architectures behave differently, and each one of them has its pros and cons.

The skip-gram model works better with a smaller amount of data and can represent rear words better than CBoW. On the other hand, CBoW is faster and can represent better words that appear more frequently.

### GloVe (Global Vectors)[24]

This method was developed at Stanford University and introduced in the year of 2014.

This method uses unsupervised learning to achieve its goal.

At first, the model builds a (word, context) co-occurrence matrix, each cell or element of the matrix represents how often a word occurs in the context (the context is an n-sequence).

The second step is to apply matrix factorization to a matrix of context and matrix of words.

Now because of this factorization, we can find the word by its soundings and vice versa.

GloVe argued to perform better than Word2Vec.

### FastText [25] [26]

Facebook firstly introduced this model in the year 2016 as an extension of Word2Vec. The main improvement over Word2Vec is the use of subword information, meaning, while Word2Vec tries to find a unique vector for each word, FastText tries to find a vector for sequences of chars within each word. Each word becomes the sum of the sub-word vectors.

### Sentence level embeddings (BERT and SCI-BERT models) [27] [28]

BERT is the state-of-the-art encoder-decoder architecture that uses a transformer for learning contextual relations between words in a given text.

The essential use of  BERT is sentence-level embedding.

### Graph Embeddings

#### Node2Vec[29]

Node2Vec is an embedding algorithm used for creating a dense representation of nodes and edges; the algorithm introduced at Stanford in 2016.

The Node2Vec tries to take the graph relationship information and embed it into vector space; nearby nodes should be closer.

Node2Vec is based on Word2Vec and builds the vector-like Word2Vec, but as we recall, Word2Vec gets a sequence of words (sentence), and Node2Vec gets graphs, any kind of graphs, so the question is:  how does Node2Vec operate like Word2Vec? The answer is that Node2Vec uses random walks to generate a group of graphs that looks like a sequence or a linked list or, in other words, to generate a bunch of sentences.

#### Structural Deep Network Embedding (SDNE) [30]

SDNE takes a different approach then Node2Vec, SDNE does not perform random walks.  This approach works with ANN and preserves the first (local pairwise similarity between nodes that are linked by edges and second (similarity between node's neighborhood structures, meaning if two nodes share many neighbors that they are similar) proximity.

In other words, it captures the local and global network structure.

## summarization Techniques[31][32]

Finally, after we talked about the technical and theoretical concepts, we can talk about summarization techniques.

### Summarization Types (taxonomy)

While the summarization work is unambiguous and well described, we still have more than one way to make the summary and even categorize the types of summaries that we can find out there.

But before we jump into this discussion, we need to understand the meaning of summarization:

Summarization is the process of shortening a text that holds the main topics/points of the original text.

There are 6 main ways to categorize a summarization:

- Number of Sources: Single or Multi-domain.
- Approaches: Abstractive or Extractive.
- Details: Indicative of Informative.
- Information Content: Generic or Query-based.
- Limitation: Domain Dependent, Genre Specific, or Independent.
- Language: Mono or Multi-Lingual.

#### *Single/Multi-Domain*

One of the lines which we can draw between the summarization systems that are out there is the input of the system. The input might be a single document or multi documents.

While it seems l, not such an important fact, it is far from being accurate, the input dimension can influence the quality of the output (summary).

In the case of single document input, the main topic might be harder to find then in a multi-document input system, were the input contains a set of documents that the main idea is the same at all of them.

#### *Abstractive/Extractive*

The second line that we can draw between the summarization types is the approach line; there are two main approaches:

1. Extractive summarization is formed by reusing subsets of the original text, subsets such as words, sentences, or even paragraphs. In this approach, this subset remains unharmed and unchanged in its original form.
2. On the other hand, in Abstract Summarization, the original text is being rewritten to a shorter version resulting in an interpretation of the original text.
   The abstractive method goals  achieved by one of the two sub-methods:
   - Extraction of parts from the original sentences (also known as compressive summarization).
   - Generation of new sentences by using a different set of words.

### *Informative/Indicative*

Another way to distinguish between the summarization methods is by looking at them as informative or indicative.

- Informative Summarization: this technique aims to cancel the reader's need in the original text. When the reader reads this summary, he is not required to have the original text; the summary should describe all the main ideas from the original text.
- Indicative Summarization: Summaries that generated with this kind of method do not claim to replace the original text; they meant to help the reader to decide if he wants to read the original work or not.

### *Generic/Query-based*

Another line that drawn is a line based on the content of the original text:  generic vs. query-based.

- Query Based: The query-based summary is all about creating a short representation of the text concerning some queries that entered; in this approach, parts of the original text have different levels of importance.
- Generic Summarization: In this type of summaries, the user does not need to have any previous understanding of the text; the underlying assumption is that different types of users may use the summary, so all the information treated with the same level of importance.

### *Domain Dependent, Genre Specific, or Independent*

Another way to look at summaries or summarization systems  based on some limitations that made on the input, there are three different types that we can look at:

- Domain Dependent: the summarizer system experts on an only domain, it can work and summarize texts which their subject defined within the domain of the system.
- Independent: these systems do not assume any limitation on the input.
- Genre Specific:  these type of summarization systems accepts particular types of text as input, the limitation is on the template of the text.

### *Mono or Multi-Lingual*

The last line that we are drawing is the line of language, some systems accept documents in different languages, and the user can choose the desired summary language. Some systems can accept or work only with documents of one language.

## summarization Evaluation Methods [31] [32] [33][34][35][36][38]

quality of a summary can differ from one human to another and from model to model.

While on the other hand, we may see different summaries from different sources that are equally good.

For Example, we all agree that:

- A summary should be easy to read and understand; a summary that hard to read is a bad summary.
- A good summary is a summary that grammatically correct.
- A good summary should convey all the essential information.
- A lousy summary is a summary that repeats on the same information more than one time.

But these examples are not enough; we need to understand how we can quantify the quality of a summary.

There are two main summarization evaluation methods, intrinsic and extrinsic.

In the extrinsic methods, the summary gets its score regarding some tasks; this type of evaluation is useful for goal-oriented summaries.

In the intrinsic methods, the summary directly analyzed. The intrinsic methods may involve a direct comparison with the original text(by measuring the coverage of the main topics) or by the closeness between the summary and some golden standard summaries that are written by human-experts.

The problem with the intrinsic methods that rely on golden standard is that they require much workforce and lack consistency.

Also, the intrinsic methods can be divided into content evaluation methods and text quality evaluation methods.

Another way to look at summarization evaluation methods is to look at them as manual-direct, manual in-direct, and automatic methods.

### Manual direct methods

In this type of method, the summary is getting a manual score from people based on subjective criteria.

### Manual indirect methods

One of the most known methods of this type is Pyramid; in this method, humans are reading the summaries of an article, the human needs to understand what is the Summary Content Units (SCUs) meaning, what is the ideas that lie beneath the text.

The summaries are being categorized into these units by the ideas that they have managed to capture.

Each unit is getting a weight representing its importance to the text.

In the last step, each summary gets its score from the units it is associated with; if the summary associated with high ranked units, it will be highly scored.

## Automatic methods

For now, the manual methods are more accurate then automated once, but they have their problems:

1. They are costly; they require a lot of human resources.
2. They lack consistency.

Because of those problems, in recent years, there is an effort to build an excellent automatic method.

### *ROUGE (Recall Oriented Understudy for Gisting Evaluation) [31][32]*

ROUGE is a family of automated methods that is most popular these days, initially introduced by Chin-Yew, Lin. At 2004.

- ROUGE-N calculates overlapping N-grams between the candidate summary and a reference summary. N usually stands for a number between 1-4.
- ROUGE-L that evaluates the quality of the summary by finding LCS between the reference text and the summary.
- ROUGE-W, which measures the length of the longest weighted common subsequence and differentiates subsequences by their length.
- ROUGE-S, which calculates the overlap of skip-bigrams between a candidate summary and a set of reference summaries.
- ROUGE-SU - overlaps of both skip-bigrams and unigrams.

### *MeMoG [31][32]*

The MeMoG method was introduced by Giannakopoulos, George, and Vangelis Karkaletsis in 2011.

MeMoG is a method that evaluates summaries with the help of N-grams and graphs. This system builds a char-based N-gram graph for the generated summary and the model summary and, afterward, compares the graphs.

### *APES [35]*

APES is a method that was developed in Ben-Gurion University of the Negev, by Eyal, Matan, Tal Baumel, and Michael Elhadad in 2019.

the basic idea is to see if the summary has all the main points of the article by trying to answer questions with the help of the summary.

The system has two main parts:

1. Automated Deep learning QA system.
2. A mechanism that checks the number of answers was answered correctly by the QA system. The more questions that are correctly answered, the higher the score of the summary.

### *BLEU[36]*

It initially introduced by Papineni, Kishore, Roukos Salim, Todd Ward, and Wei-Jing Zhu in 2002.

BLEU is another type of automatic method; this method calculates N-grams and precision between a candidate and a set of references.

At first, the BLEU initially designed as an evolution method of machine translation; the main idea was to check how close machine translation to a professional human translation.

## System Description

The purpose of this section is to provide a debriefed view of the requirements, specifications, and design of the system.

The intended audience of this section is anyone with some programming, NLP, and machine learning familiarly.

This section of the document includes:

- Description of the product, the terminology, and the technology that in use.
- Description of the Interface, the functional and non-functional requirements.

### Terminology

| Term | Description |
|---|---|
| User | Anyone who is interacting with the software |
| System | The package of all the components of the project |
| Deep learning | A machine learning sub-field that tries to solve problems by simulating the neural network in the human brain and the process of thinking. |
| Python | A high level, general-purpose programming language, one of its uses is in the field of scientific, statistical and |
| TensorFlow | A differential computing library mainly used for designing of Neural Networks. |
| Pre-processing | The process of transforming raw data into  data that  used as input to the model |
| NLP | Natural Language Preprocessing. |
| Tokenizer | A program that breaks a sequence into significant pieces such as words, sentences. |
| Decontraction | The process of expanding the word from a contracted state. |
| Lemmatization | Lemmatization is the process of transforming different forms of a single word into its root form. |
| Stop words | common words in a language (e.g., a, an, the, not, nor). |
| Embedding | A numerical representation of tokens and the similarity between them. |
| GUI | Graphical User Interface. |
| Server | A strong computer with GPU for faster training |

## Main Technology

All the programming is done with the Python 3 Programming language because it is a mature programming language for general, statistical, and machine learning proposes.

There are some frameworks of neural networks in the market; the project works with TensorFlow 2.

### TensorFlow

TensorFlow is an open-source ML and NN library developed by Google and written in Python; this framework is trendy. TensorFlow gains its popularity  from:

- Scalability.
- Support for production on various devices and platforms.
- Training and network visualization that helps to spot problems.
- Hot model swap support.
- Distributed training support.
- TensorFlow community is broad; there is a massive amount of content such as code, tutorials, and support.

Although it is advantages, there are some minuses with TensorFlow, but the main one was:

- The computation graph is static and needed to be defined and compiled before running the model. (from ver. 2 of TensorFlow there is an option for dynamic graphs).

### Bert As a Service

A server that gives us the ability to encode our text (on the sentence level) into the embeddings with the help of a pre-trained Bert model (sciBert in this project).

### Stanza

Stanza is a Python natural language analysis package. It contains tools, which used to convert human-readable text into lists of sentences and words, to lemmatize the words much more.

The project uses stanza for tokenization of the text into sentence level.

### NLTK

Another Pythonic natural language analysis package, the project uses this package to check if the text is in English (thaw we could do this with Naive Bayes) and to get a stop words list.

## Product Perspective

the system contains 4 main components:

- Laptop or a workstation that sends the data to the model.
- Deep Learning model that gives a score to the summary regarding its much to the article.
- Server to (re)train our deep learning model.
- User-end GUI based software for scoring the summary concerning the article.

### Data

The model trained on 16,000 articles taken from the CORD-19: The Covid-19 Open Research Dataset. [38]

The summaries and its ranks generated in the following way:

- 1 score for the Abstract of the article.
- 0 score for random permutation of the summaries.
- 0.5 score for concatenating of the first half of the abstract with some noise from news domain, zeros, or with duplication of the first half.

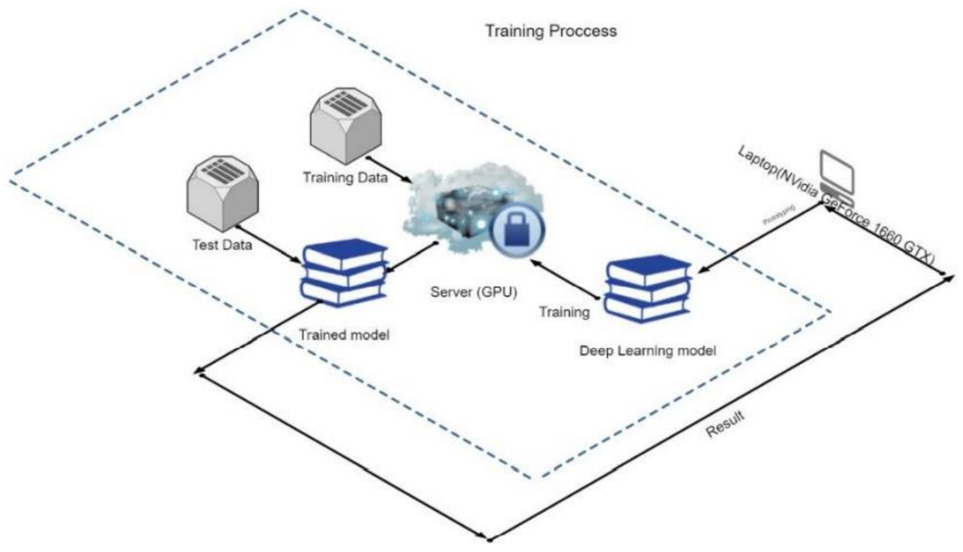
### Training Process and High-level Diagram

This sub-section covers the preparation of the data and the training of the model.

- The first step is to read the data as pairs of (article, abstract) with label 1 (good summary).
- The second step is the data pre-processing; this process is done for both articles and summaries.
  For each article and summary:
  - Decontraction.
  - Stop words removal except no, not, and nor (optional).
  - Tokenization into the sentence level.
  - Sentence level encoding (Sentence Embeddings).
  - Generation of bad summaries as described in the data sub-section.
  - Padding

  And finally:

  - Data splitting (into train, dev, and test sets).
  - Data serializing.
- Model training and testing.



## Basic Functionality

In this sub-section, we introduce the essential functions of this product and a brief description of them.

A more detailed one, as well as functional and non-functional requirements, is introduced in later sub-sections.

| ID | Function | Description |
|---|---|---|
| 1 | Rank pair from the screen. | Rank one pair of data |
| 2 | Train Model. | Train a model. |
| 3 | Retrain Model. | Retrain the model with new data. |
| 4 | Save full results. | Save full results into a file. |
| 5 | Show results. | Show results on the screen. |

## User stories

1. As a user, I want to copy-paste a pair of (article, summary) to the screen so the system can rank it.
2. As a user, I want the ability to re-train the model with new articles so that the model will work better.
3. As a user, I want the ability to train a new model with a configuration file for the hyperparameters.
4. As a user, I want the ability to see the results of the (re)train or the ranking in the file For future use.
5. As a user, I want to see the highlighted results on the screen.

| | |
|---|---|
| ID | 1 |
| Use Case | Rank pair from the screen. |
| Description | Rank one pair of data |
| Actor | User |
| Trigger | The user clicks on the “RankIT!” button. |
| Primary Scenario | - The user selects the “eval” option in the menu bar.<br>- User inputs the article and the summary into corresponding text boxes.<br>- The user clicks on the “RankIT!” button.<br>- The system preprocessing the article and the summary:<br>Decontraction.<br>(Optional) Stop words removal.<br>Sentence level tokenization.<br>Sentence level embedding.<br>Padding.<br>-The system passes the data into the model for evaluation.<br>- The system outputs the results. |
| Exceptional Scenarios | -The folder is empty.<br>-The folder does not contain the configuration file.<br>-The data does not appear in the right format.<br>-Data is not in English<br>-Users' system does not strong enough. |

| | |
|---|---|
| ID | 2 |
| Use Case | Train Model. |
| Description | Train a model. |
| Actor | User |
| Trigger | The user clicks on “TrainIT!” button. |
| Primary Scenario | - The user selects “Train” option in the menu bar.<br>- The user indicates a path to directory with the data.<br>- The user clicks on “TrainIT!!” button.<br>- The system reads the data.<br>- The system preprocessing  the data:<br>  for each article and  summary:<br>    Decontraction.<br>    (Optional) Stop words removal.<br>    Sentence level tokenization.<br>    Sentence level embedding.<br>  Generation of bad summaries.<br>  Padding.<br>  Splitting into train, dev, and test sets.<br>  Data Sterilization.<br>- The system reads the configuration file.<br>- The system defines a model.<br>- The system passes the data into the model for training.<br>- The system outputs the results. |
| Exceptional Scenarios | -The user did not entered text into one or both of the boxes .<br>-The data is not in English<br>-Users' system does not strong enough. |

| | |
|---|---|
| ID | 4 |
| Use Case | Save full results. |
| Description | Save full results into a file. |
| Actor | System |
| Trigger | The model ends the (re)train or rank process. |
| Primary Scenario | - The model ends the (re)train or rank process.<br>- The system writes the results into the file. |
| Exceptional Scenarios | -there enough space in the machine. |

| | |
|---|---|
| ID | 5 |
| Use Case | Show results. |
| Description | Show results on the screen. |
| Actor | System |
| Trigger | The model ends the (re)train or rank process. |
| Primary Scenario | - The model ends the (re)train or rank process.<br>- The system Shows the results on the screen. |

| | |
|---|---|
| ID | 3 |
| Use Case | ReTrain Model. |
| Description | Retrain the model with new data. |
| Actor | User |
| Trigger | The user clicks on the “reTrainIT!” button. |
| Primary Scenario | - The user selects the “reTrain” option in the menu bar.<br>- The user indicates a path to directory with the data.<br>- The user clicks on the “reTrainIT!” button.<br>- The system reads the data.<br>- The system preprocessing the data:<br>  for each article and summary:<br>    Decontraction.<br>    (Optional) Stop words removal.<br>    Sentence level tokenization.<br>    Sentence level embedding.<br>  Generation of bad summaries.<br>  Padding.<br>  Split into a train, dev, and test sets.<br>  Data Sterilization.<br>- The system loads the model.<br>- The system passes the data into the model for training.<br>- The system outputs the results. |
| Exceptional Scenarios | -The user did not enter text into one or both of the boxes.<br>-Data is not in English<br>-The user's system does not strong enough. |

## Operating Environment

We have the following minimum specification for the (re)train process:

- OS: Windows, Mac, BSD, or Linux distribution.
- Processor: Multi-core Intel CPU.
- Memory:  8 GB of RAM.
- 1TB SSD  (at least 512 GB swap/page file).
- Nvidia (Cuda enabled) GPU with at least 12 GB of RAM.
- Python 3.6.

We have the following minimum specifications for the ranking process:

- OS: Windows, Mac, BSD, or Linux distribution.
- Memory: 16 GB of RAM.
- Python 3.6.

## Design and Implementation Constraints

- We need massive processing power for running the train and re-train processes.
- We need a considerable amount of RAM to pre-process all the data.
- We need a massive dataset so we can better generalize the model.

## Functional Requirements

In this sub-section, we dive deep into the main functional requirements and features and extend our understanding of the use-cases.

### Rank Summary

One of the cores of our system is the “rank summary” functionality, this functionality depends on various functions, but without the following, it fails to achieve its goal, here is the primary “must-have “functionality”:

- The system must provide text parser functionality that divides the text into tokens.
- The system should clean any noise and unnecessary words.
- The system should encode the text into sentence level embeddings.
- The system should provide a pre-trained deep learning model that ranks the summary concerning the article.

### Re-train Model

One of the cores of our system is the “re-train model” functionality, this functionality depends on various functions, but without the following, it fails to achieve its goal, here is the primary “must-have “functionality”:

- The system must provide text parser functionality that divides the text into tokens.
- The system should clean any noise and unnecessary words.
- The system should encode the text into sentence level embeddings.
- The system should provide a pre-trained deep learning model to re-train on.

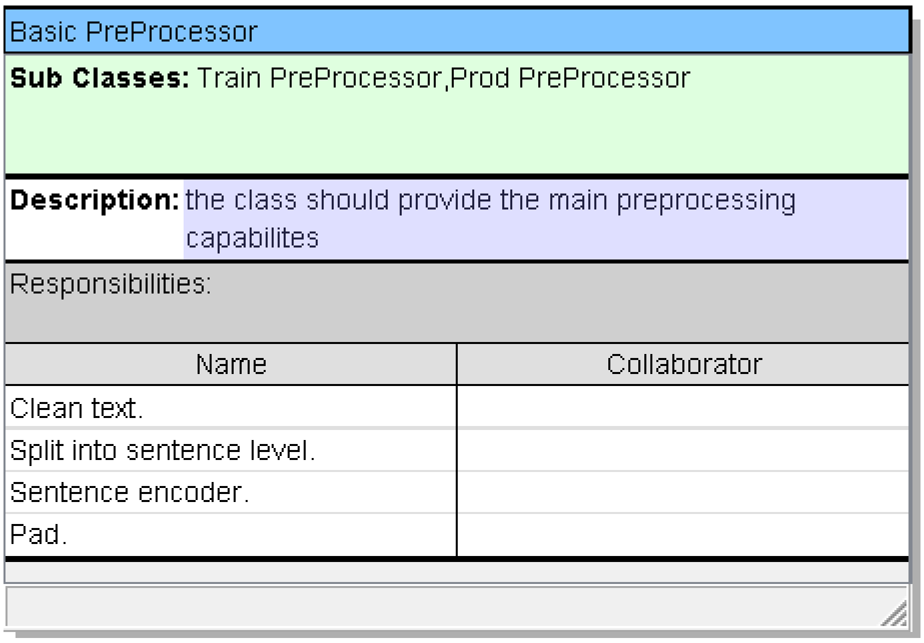
### Train Model

One of the cores of our system is the “re-train model” functionality, this functionality depends on various functions, but without the following, it fails to achieve its goal, here is the primary “must-have “functionality”:

- The system must provide text parser functionality that divides the text into tokens.
- The system should clean any noise and unnecessary words.
- The system should encode the text into sentence level embeddings.

## Modeling and Description

### Class Responsibility Collaborator (CRC)



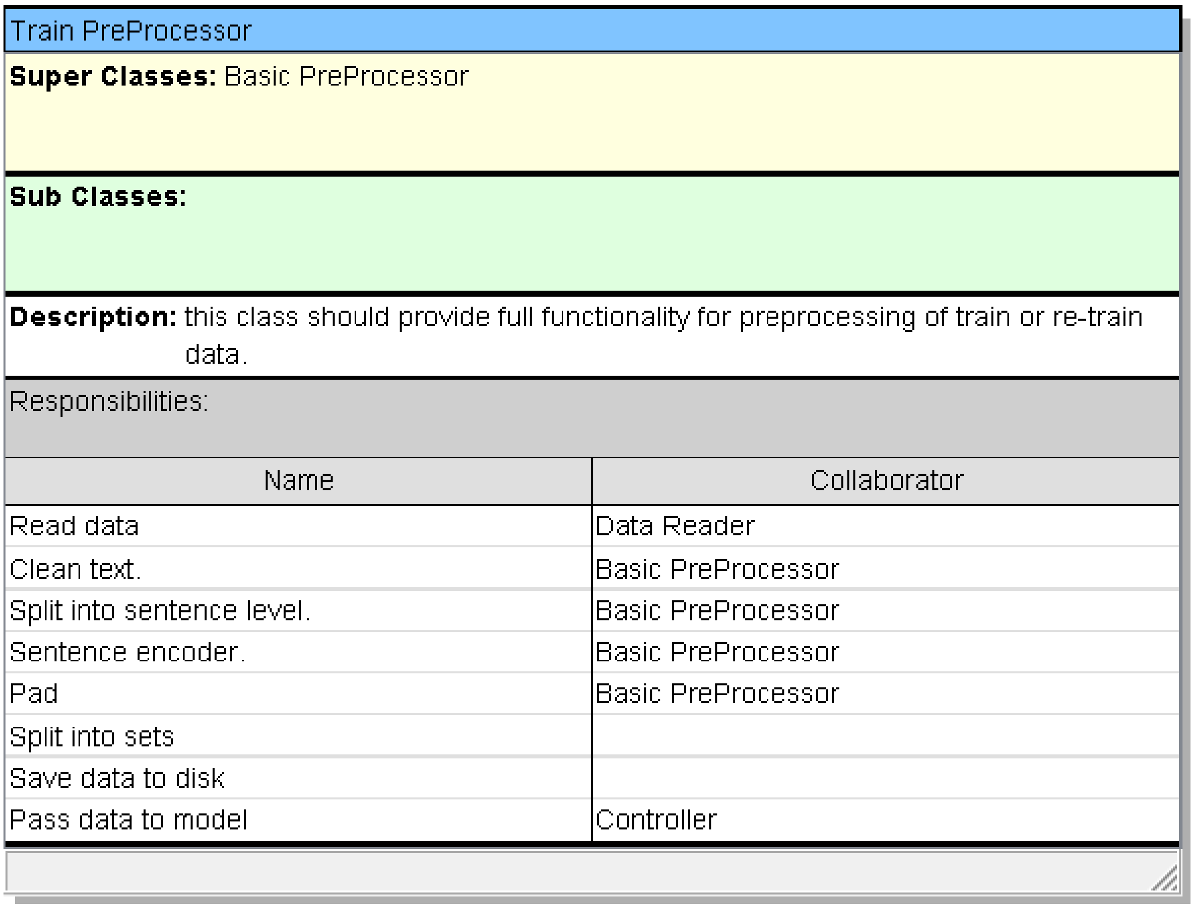
Basic PreProcessor

**Sub Classes:** Train PreProcessor,Prod PreProcessor

**Description:** the class should provide the main preprocessing capabilites

Responsibilities:

| Name | Collaborator |
|---|---|
| Clean text. | |
| Split into sentence level. | |
| Sentence encoder. | |
| Pad. | |



Train PreProcessor

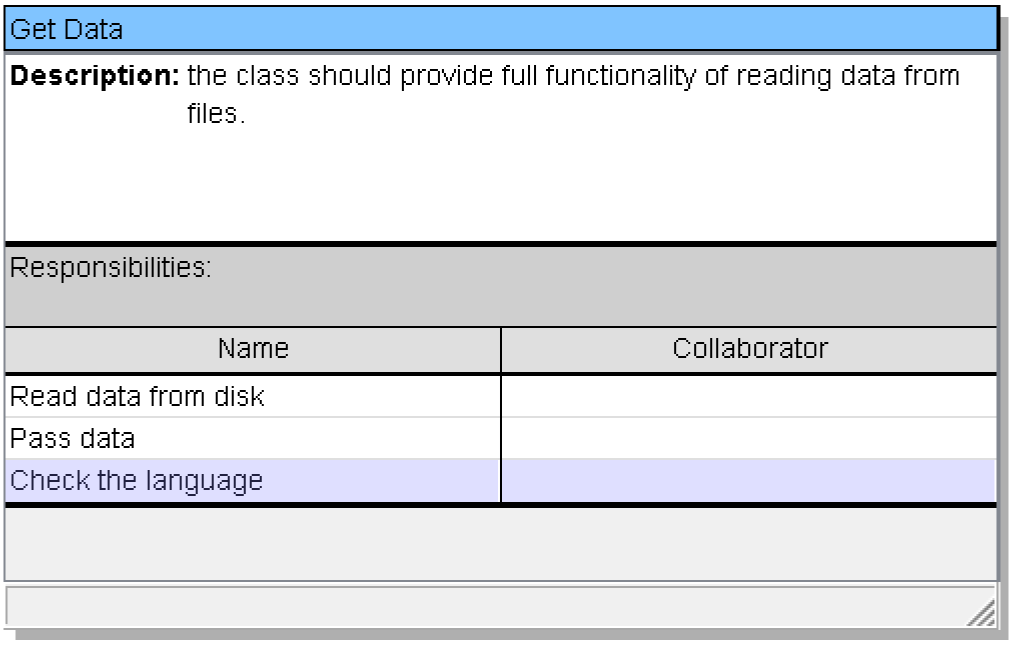
**Super Classes:** Basic PreProcessor

**Sub Classes:**

**Description:** this class should provide full functionality for preprocessing of train or re-train data.

Responsibilities:

| Name | Collaborator |
|---|---|
| Read data | Data Reader |
| Clean text. | Basic PreProcessor |
| Split into sentence level. | Basic PreProcessor |
| Sentence encoder. | Basic PreProcessor |
| Pad | Basic PreProcessor |
| Split into sets | |
| Save data to disk | |
| Pass data to model | Controller |

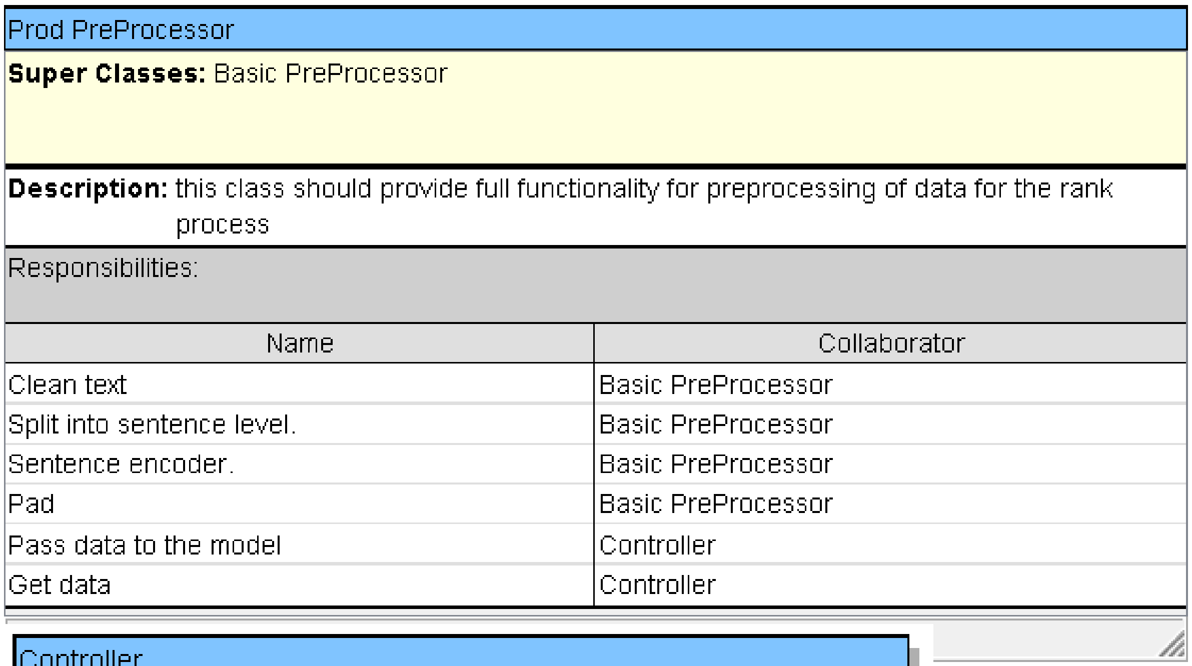


Get Data

**Description:** the class should provide full functionality of reading data from files.

Responsibilities:

| Name | Collaborator |
|---|---|
| Read data from disk | |
| Pass data | |
| Check the language | |



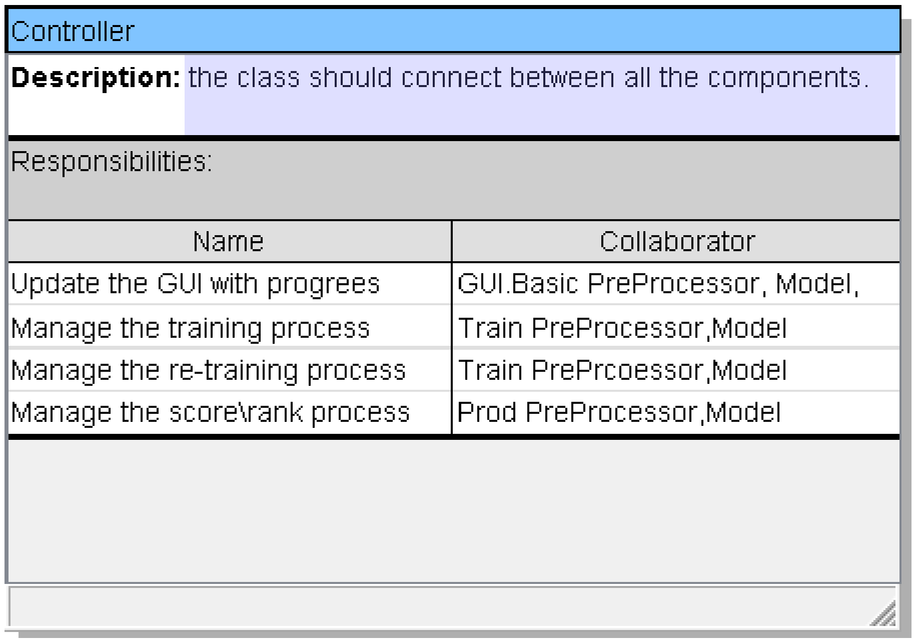
Prod PreProcessor

**Super Classes:** Basic PreProcessor

**Description:** this class should provide full functionality for preprocessing of data for the rank process

Responsibilities:

| Name | Collaborator |
|---|---|
| Clean text | Basic PreProcessor |
| Split into sentence level. | Basic PreProcessor |
| Sentence encoder. | Basic PreProcessor |
| Pad | Basic PreProcessor |
| Pass data to the model | Controller |
| Get data | Controller |

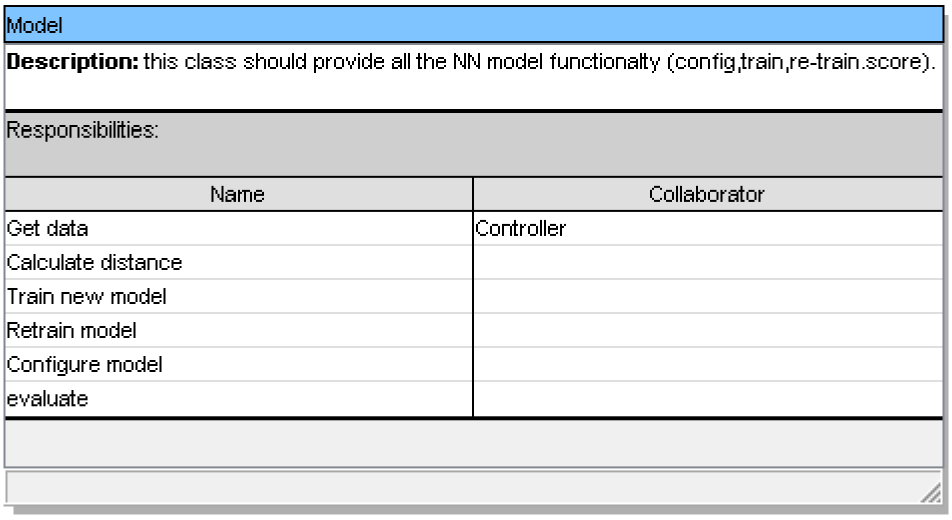


Controller

**Description:** the class should connect between all the components.

Responsibilities:

| Name | Collaborator |
|---|---|
| Update the GUI with progrees | GUI.Basic PreProcessor, Model, |
| Manage the training process | Train PreProcessor,Model |
| Manage the re-training process | Train PrePrcoessor,Model |
| Manage the score\rank process | Prod PreProcessor,Model |

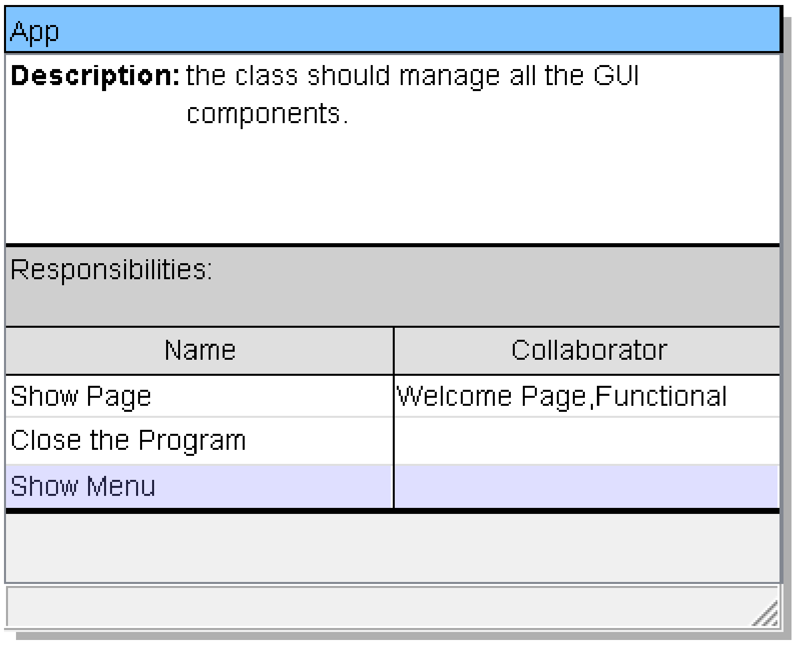


Model

**Description:** this class should provide all the NN model functionalty (config,train,re-train.score).

Responsibilities:

| Name | Collaborator |
|---|---|
| Get data | Controller |
| Calculate distance | |
| Train new model | |
| Retrain model | |
| Configure model | |
| evaluate | |

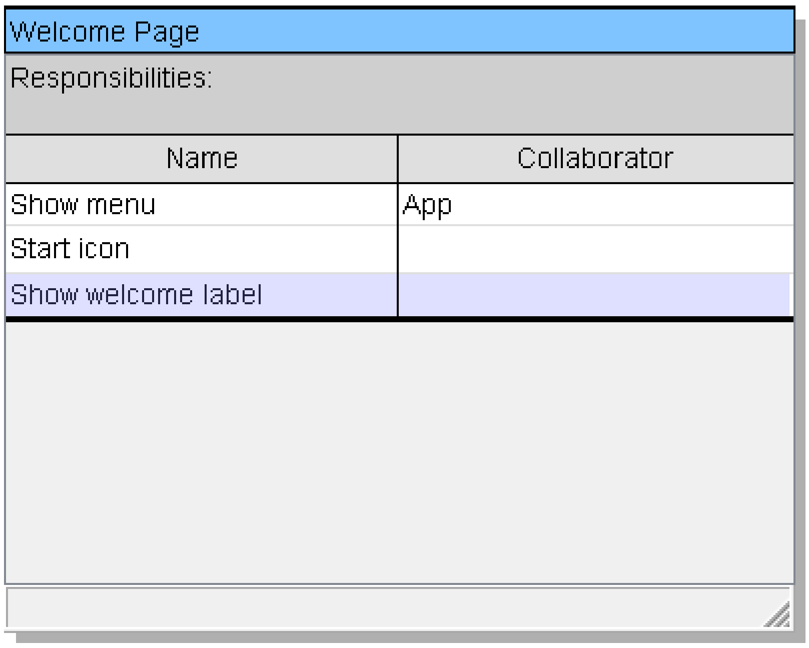


App

**Description:** the class should manage all the GUI components.

Responsibilities:

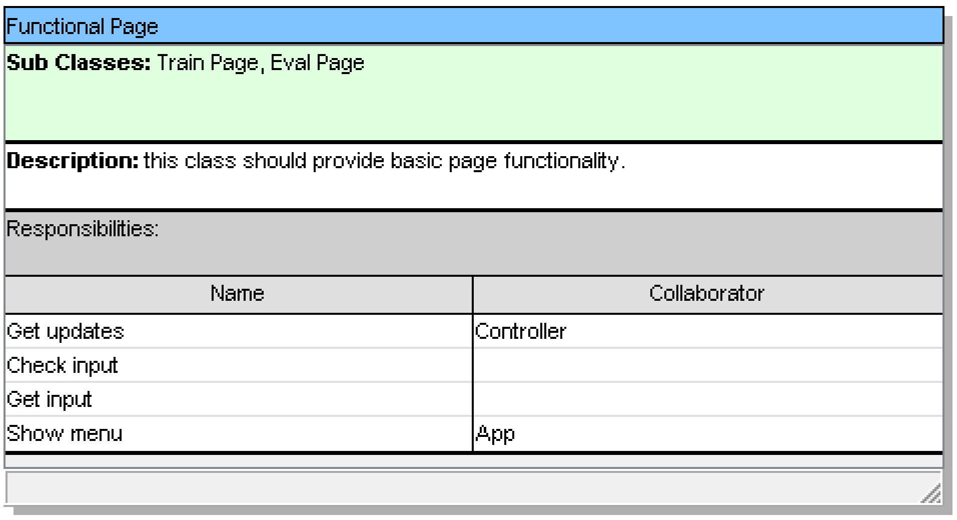
| Name | Collaborator |
|---|---|
| Show Page | Welcome Page,Functional |
| Close the Program | |
| Show Menu | |



Welcome Page

Responsibilities:

| Name | Collaborator |
|---|---|
| Show menu | App |
| Start icon | |
| Show welcome label | |



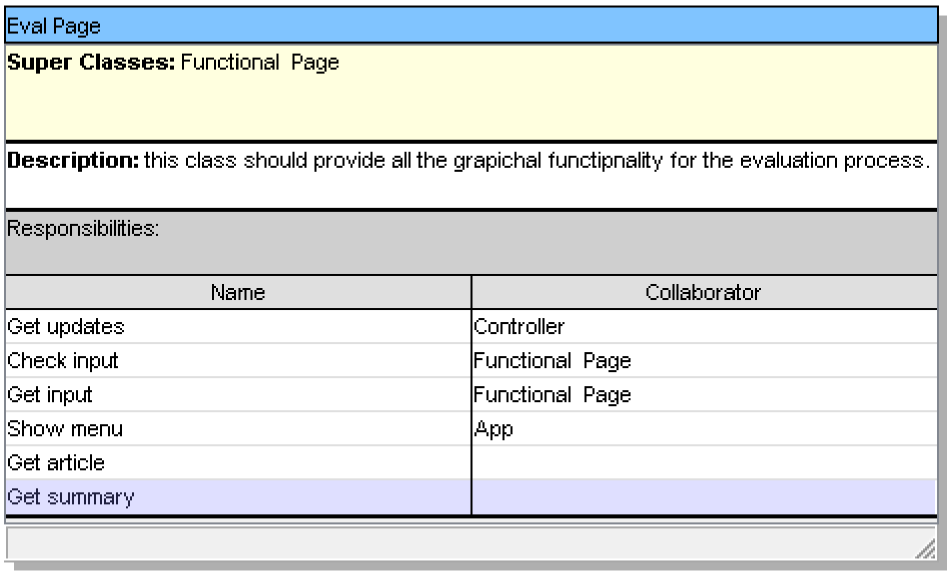
Functional Page

**Sub Classes:** Train Page, Eval Page

**Description:** this class should provide basic page functionality.

Responsibilities:

| Name | Collaborator |
|---|---|
| Get updates | Controller |
| Check input | |
| Get input | |
| Show menu | App |



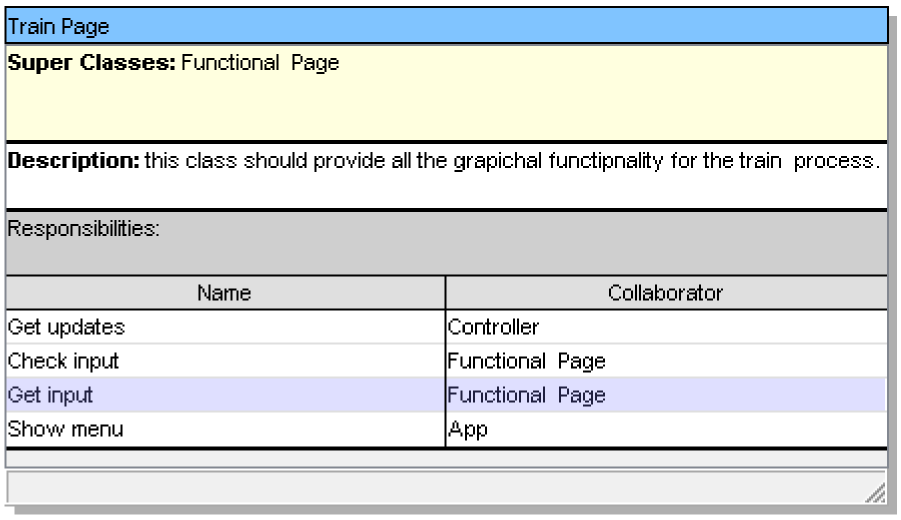
Eval Page

**Super Classes:** Functional  Page

**Description:** this class should provide all the grapichal functipnality for the evaluation process.

Responsibilities:

| Name | Collaborator |
|---|---|
| Get updates | Controller |
| Check input | Functional  Page |
| Get input | Functional  Page |
| Show menu | App |
| Get article | |
| Get summary | |



Train Page

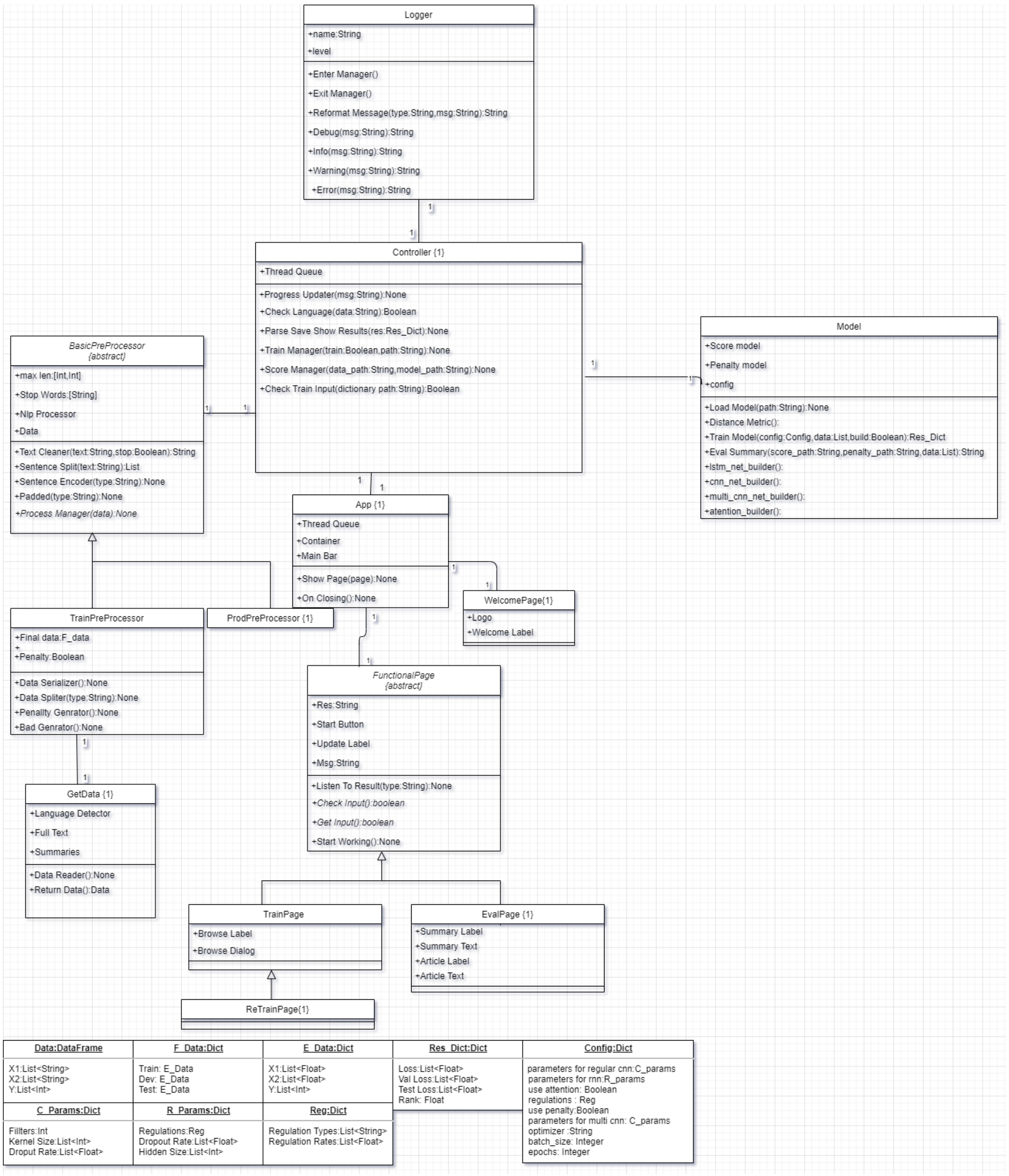
**Super Classes:** Functional  Page

**Description:** this class should provide all the grapichal functipnality for the train  process.

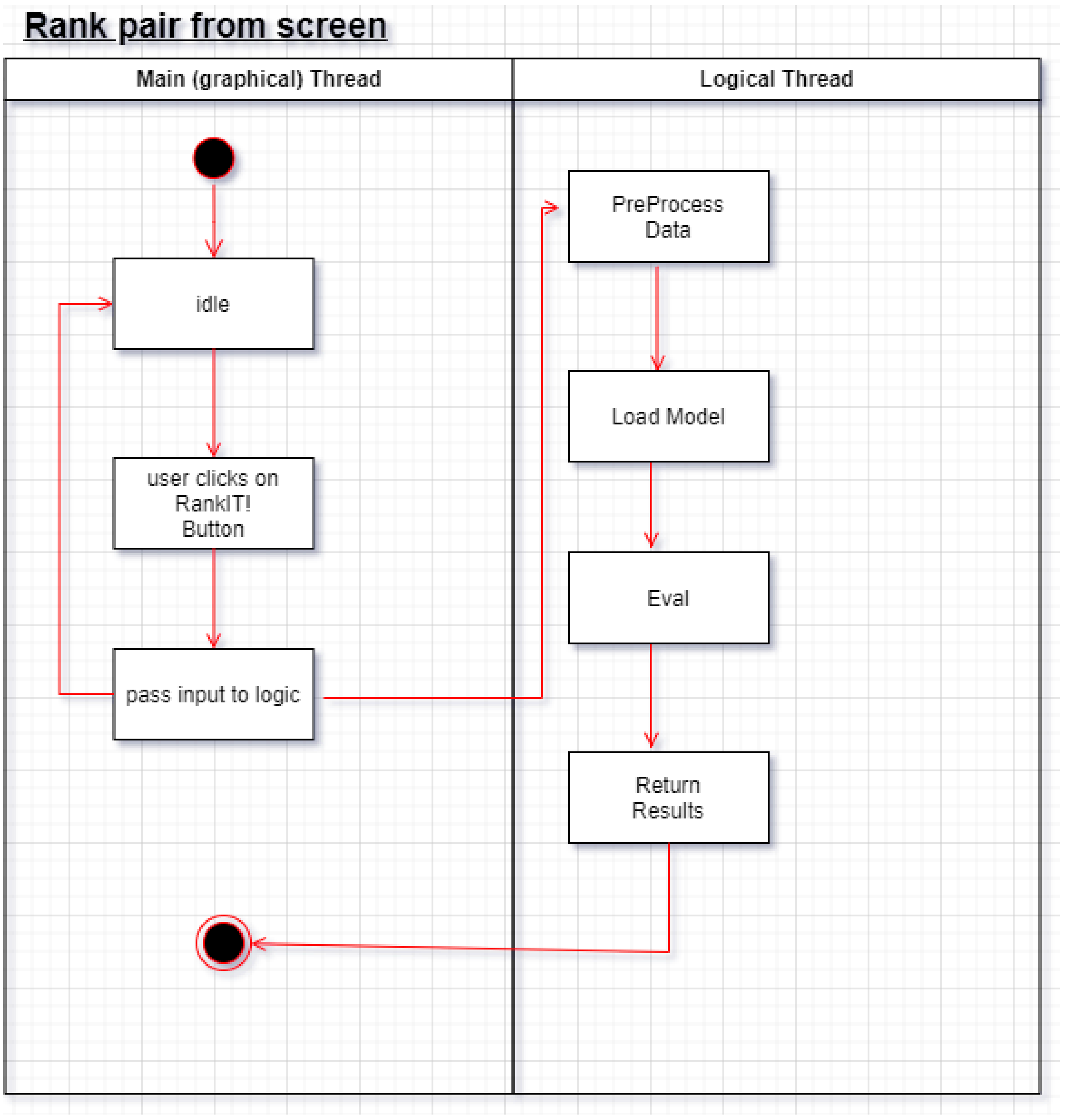
Responsibilities:

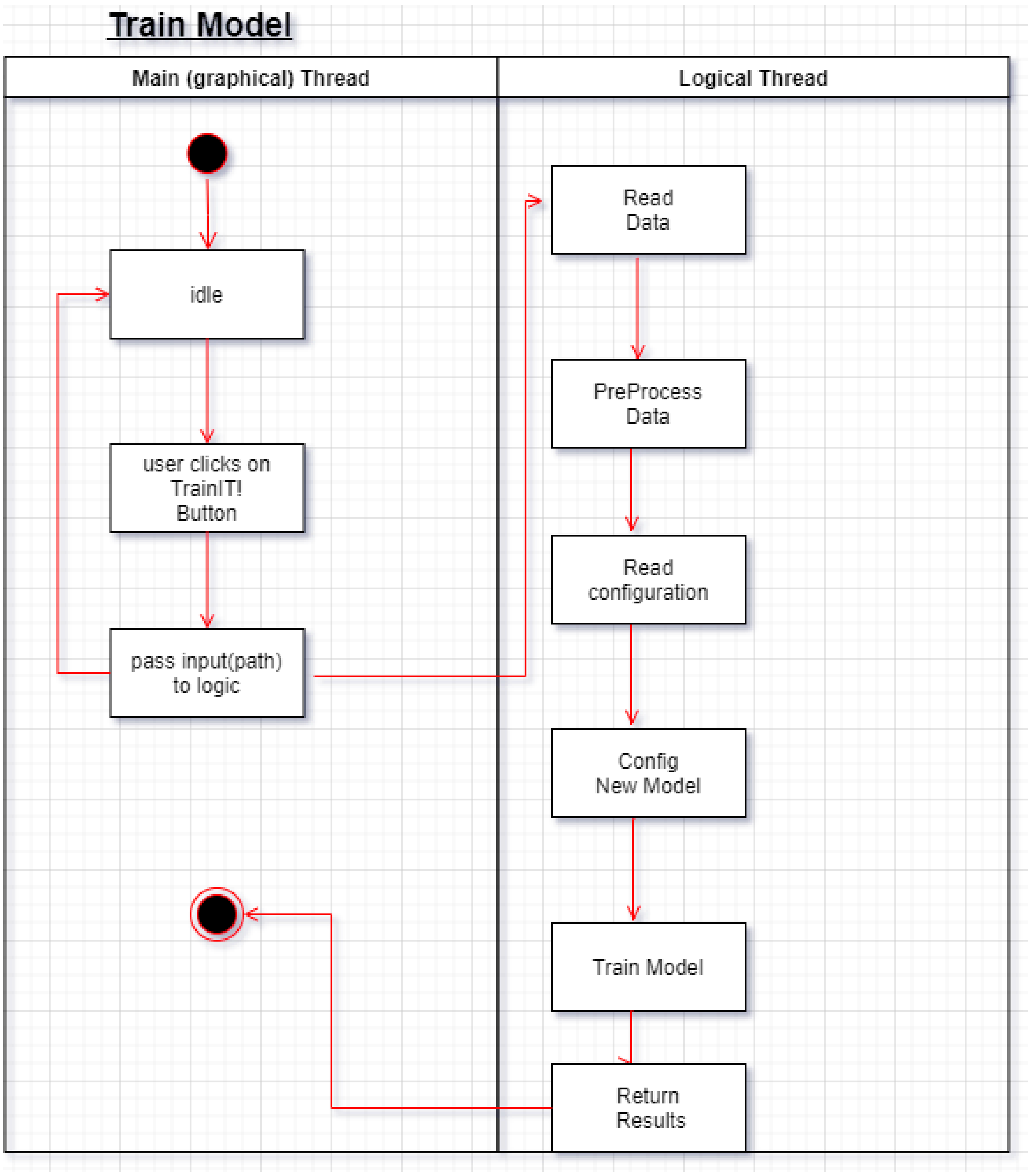
| Name | Collaborator |
|---|---|
| Get updates | Controller |
| Check input | Functional  Page |
| Get input | Functional  Page |
| Show menu | App |

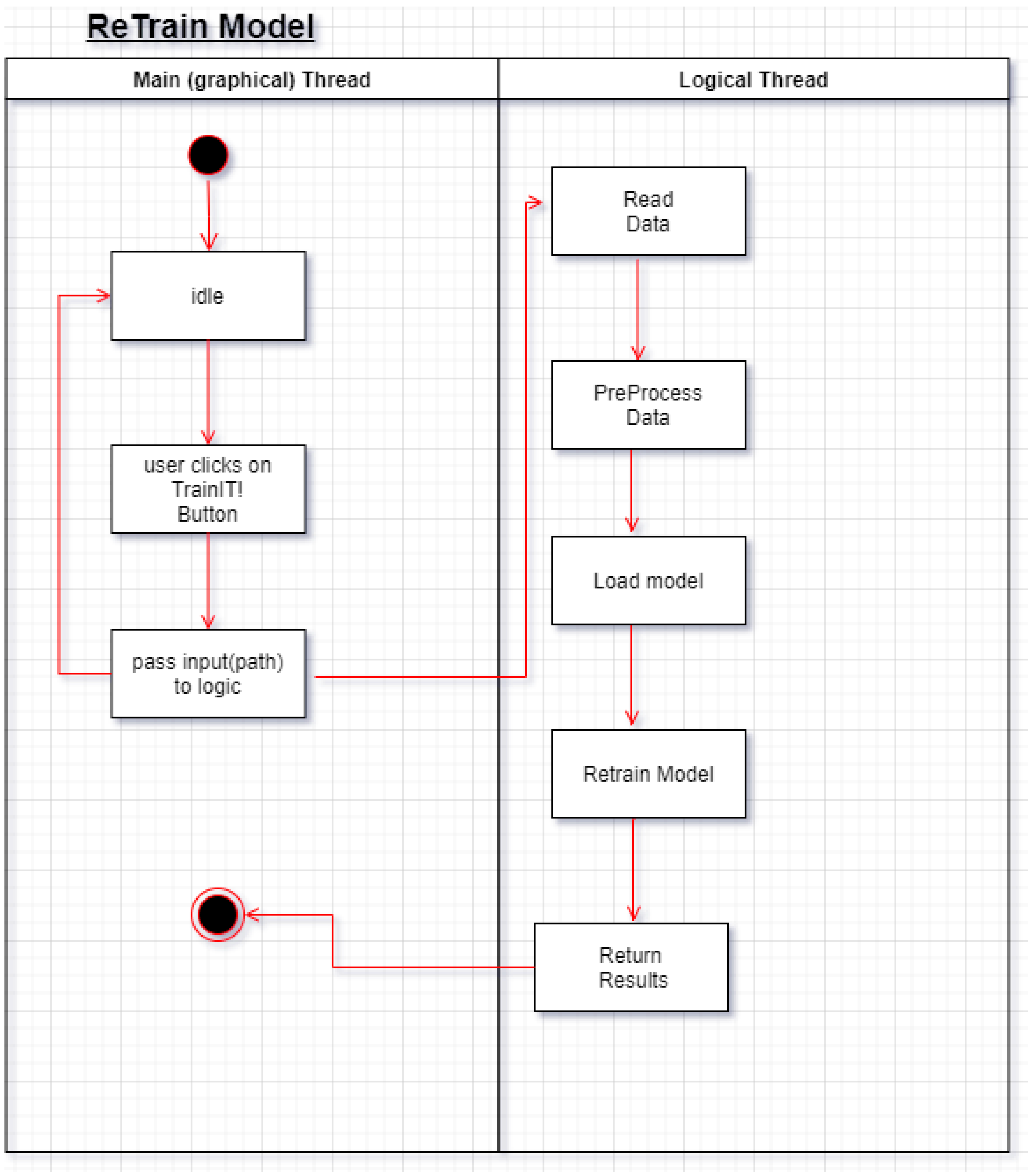
## Class diagram



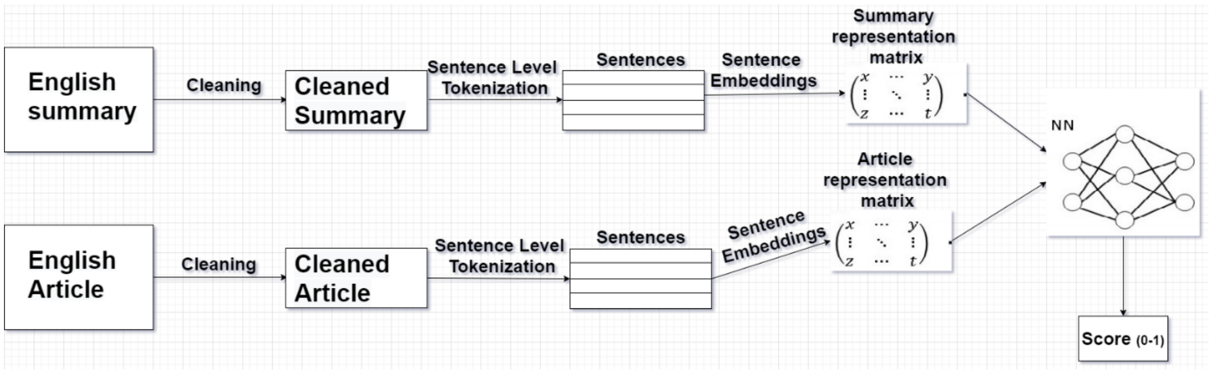
## Activity diagram




Train Model
Main (graphical) Thread
Logical Thread
idle
user clicks on TrainIT! Button
pass input(path) to logic
Read Data
PreProcess Data
Read configuration
Config New Model
Train Model
Return Results


ReTrain Model
Main (graphical) Thread
Logical Thread
idle
user clicks on TrainIT! Button
pass input(path) to logic
Read Data
PreProcess Data
Load model
Retrain Model
Return Results

## Full pipeline

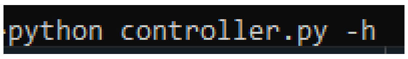


## User Interface

There are two interfaces GUI and CLI.

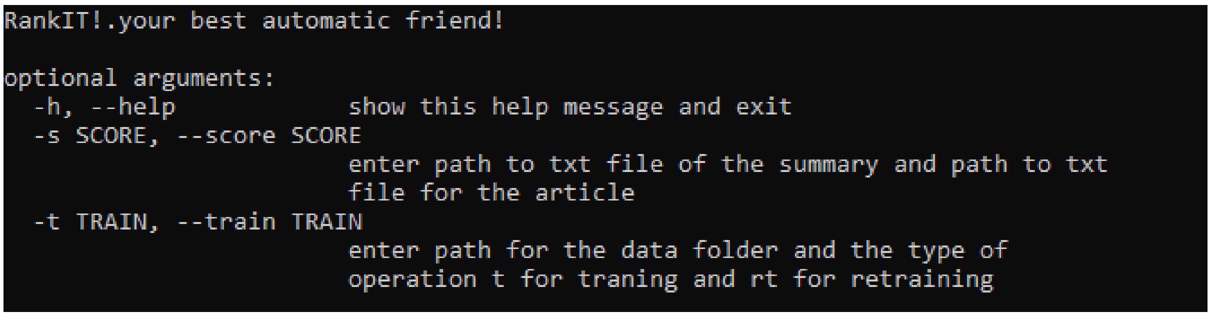
### Command-line interface

The first option to engage with our program is the command-line interface, to work with this interface all we need to do to type Python PATHTO/controller.py and one of the options (-h,-t,-s).

For example, to get help, we need to enter 

```
python controller.py -h
```

And we get the help menu:

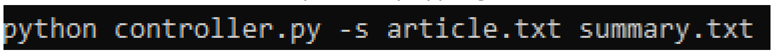


```
RankIT!.your best automatic friend!

optional arguments:
  -h, --help            show this help message and exit
  -s SCORE, --score SCORE
                        enter path to txt file of the summary and path to txt
                        file for the article
  -t TRAIN, --train TRAIN
                        enter path for the data folder and the type of
                        operation t for traning and rt for retraining
```

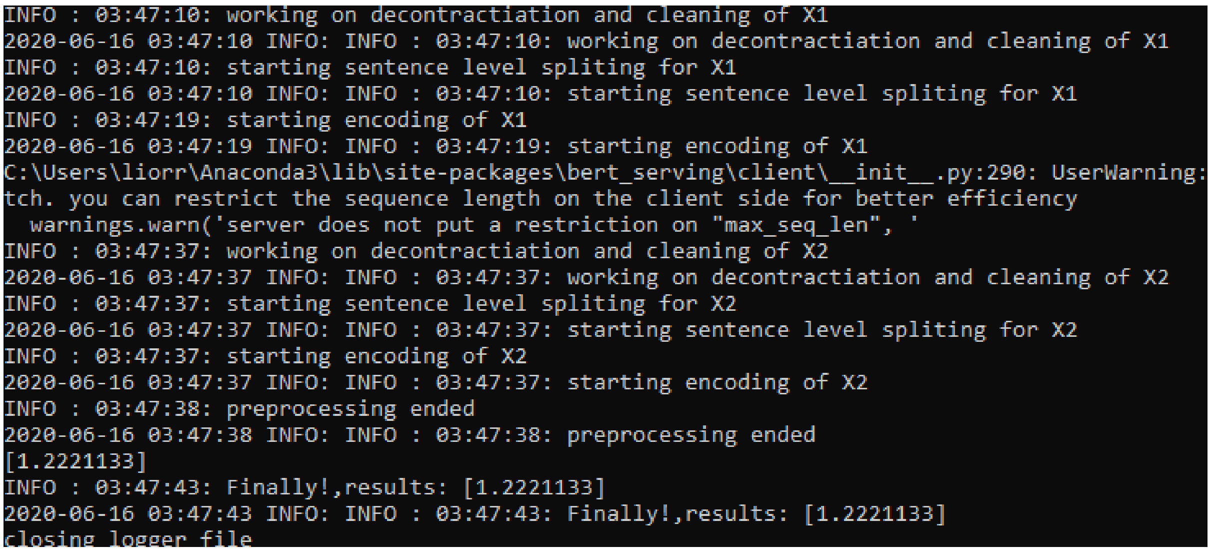
As we can see, there are two main options: The score for scoring a summary and t for the training new model or retraining the best model.

Let us look at the score option, by typing



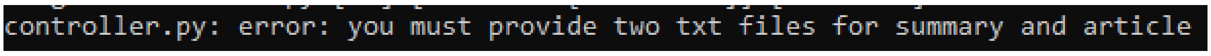
```
python controller.py -s article.txt summary.txt
```

We can see progress messages and in the end the score :

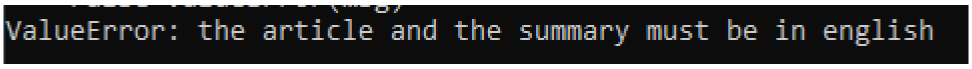


```
INFO : 03:47:10: working on decontractiation and cleaning of X1
2020-06-16 03:47:10 INFO: INFO : 03:47:10: working on decontractiation and cleaning of X1
INFO : 03:47:10: starting sentence level spliting for X1
2020-06-16 03:47:10 INFO: INFO : 03:47:10: starting sentence level spliting for X1
INFO : 03:47:19: starting encoding of X1
2020-06-16 03:47:19 INFO: INFO : 03:47:19: starting encoding of X1
C:\Users\liorr\Anaconda3\lib\site-packages\bert_serving\client\__init__.py:290: UserWarning:
tch. you can restrict the sequence length on the client side for better efficiency
  warnings.warn('server does not put a restriction on "max_seq_len", '
INFO : 03:47:37: working on decontractiation and cleaning of X2
2020-06-16 03:47:37 INFO: INFO : 03:47:37: working on decontractiation and cleaning of X2
INFO : 03:47:37: starting sentence level spliting for X2
2020-06-16 03:47:37 INFO: INFO : 03:47:37: starting sentence level spliting for X2
INFO : 03:47:37: starting encoding of X2
2020-06-16 03:47:37 INFO: INFO : 03:47:37: starting encoding of X2
INFO : 03:47:38: preprocessing ended
2020-06-16 03:47:38 INFO: INFO : 03:47:38: preprocessing ended
[1.2221133]
INFO : 03:47:43: Finally!,results: [1.2221133]
2020-06-16 03:47:43 INFO: INFO : 03:47:43: Finally!,results: [1.2221133]
closing logger file
```

By the way! If you enter wrong input, you get an exception (it applies for GUI as well), for example:



```
controller.py: error: you must provide two txt files for summary and article
```

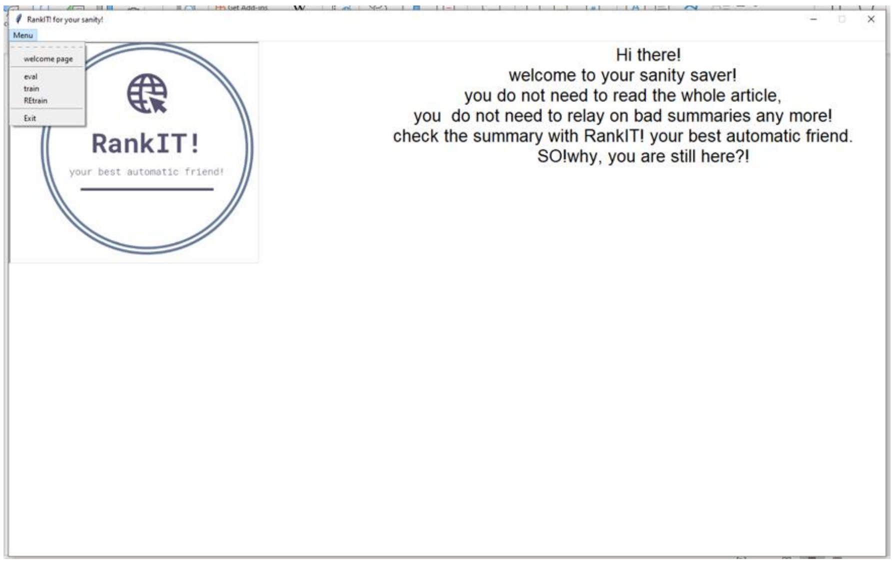


```
ValueError: the article and the summary must be in english
```

## Graphical interface

The second option is the graphical one, to enter the graphical one; all we need to do is to start the GUI.py script.

The main window:



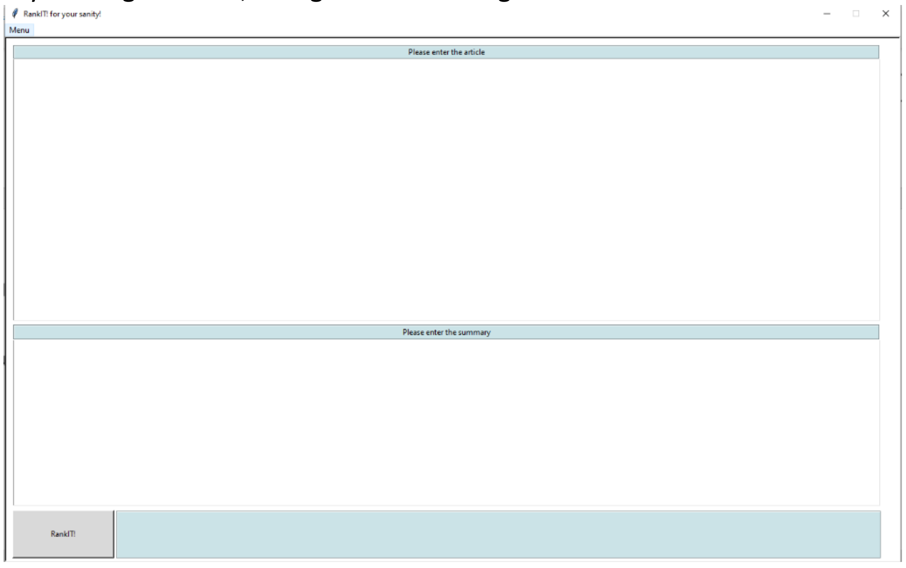
Hi there!
welcome to your sanity saver!
you do not need to read the whole article,
you  do not need to relay on bad summaries any more!
check the summary with RankIT! your best automatic friend.
SO!why, you are still here?!

As we can see, there are three main options:

- Eval for evaluation (scoring) of a summary.
- Train for training a new model.
- Retrain for retraining an existing model.

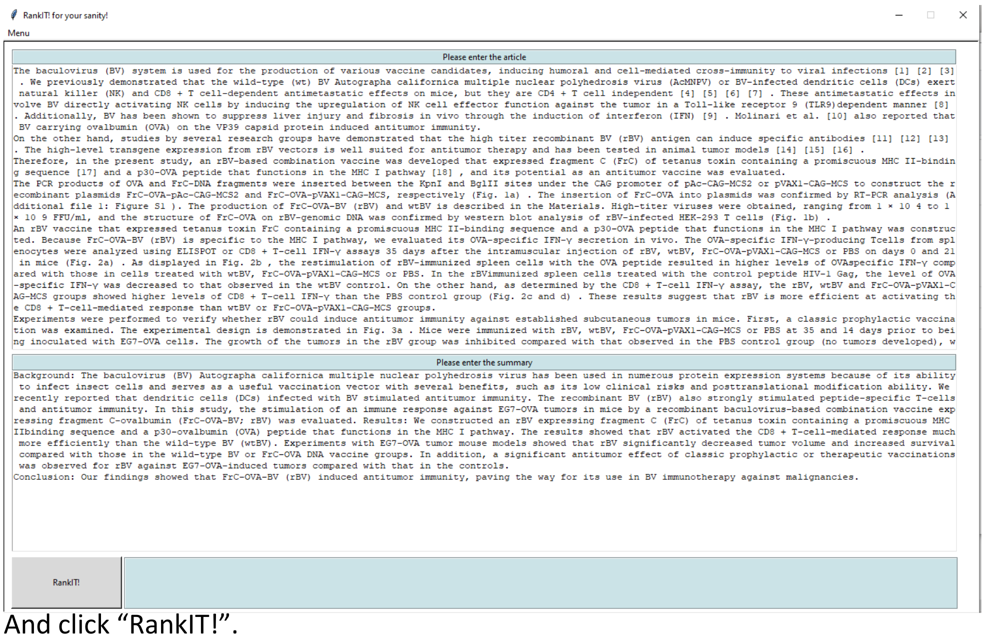
Let us look at the scoring process.

By clicking on “eval,” we get the following window:



Lior Reznik, RankIT!

Let us put an article and summary.

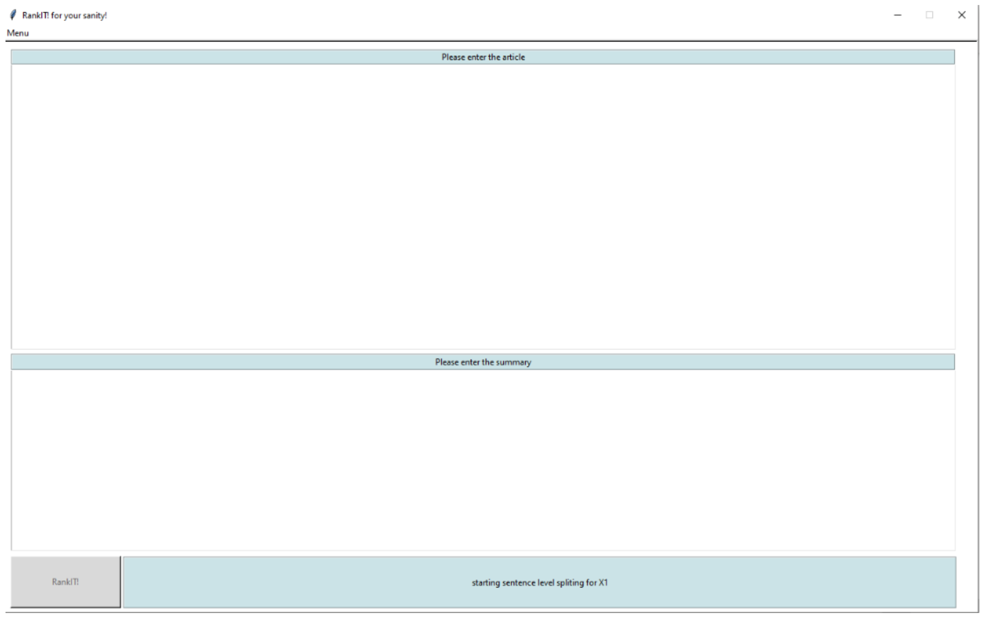


```
The baculovirus (BV) system is used for the production of various vaccine candidates, inducing humoral and cell-mediated cross-immunity to viral infections [1] [2] [3]
 . We previously demonstrated that the wild-type (wt) BV Autographa californica multiple nuclear polyhedrosis virus (AcMNPV) or BV-infected dendritic cells (DCs) exert
 natural killer (NK) and CD8 + T cell-dependent antimetastatic effects on mice, but they are CD4 + T cell independent [4] [5] [6] [7] . These antimetastatic effects in
volve BV directly activating NK cells by inducing the upregulation of NK cell effector function against the tumor in a Toll-like receptor 9 (TLR9)dependent manner [8]
. Additionally, BV has been shown to suppress liver injury and fibrosis in vivo through the induction of interferon (IFN) [9] . Molinari et al. [10] also reported that
 BV carrying ovalbumin (OVA) on the VP39 capsid protein induced antitumor immunity.
On the other hand, studies by several research groups have demonstrated that the high titer recombinant BV (rBV) antigen can induce specific antibodies [11] [12] [13]
. The high-level transgene expression from rBV vectors is well suited for antitumor therapy and has been tested in animal tumor models [14] [15] [16] .
Therefore, in the present study, an rBV-based combination vaccine was developed that expressed fragment C (FrC) of tetanus toxin containing a promiscuous MHC II-bindin
g sequence [17] and a p30-OVA peptide that functions in the MHC I pathway [18] , and its potential as an antitumor vaccine was evaluated.
The PCR products of OVA and FrC-DNA fragments were inserted between the KpnI and BglII sites under the CAG promoter of pAc-CAG-MCS2 or pVAX1-CAG-MCS to construct the r
ecombinant plasmids FrC-OVA-pAc-CAG-MCS2 and FrC-OVA-pVAX1-CAG-MCS, respectively (Fig. 1a) . The insertion of FrC-OVA into plasmids was confirmed by RT-PCR analysis (A
dditional file 1: Figure S1 ). The production of FrC-OVA-BV (rBV) and wtBV is described in the Materials. High-titer viruses were obtained, ranging from 1 × 10 4 to 1
× 10 9 FFU/ml, and the structure of FrC-OVA on rBV-genomic DNA was confirmed by western blot analysis of rBV-infected HEK-293 T cells (Fig. 1b) .
An rBV vaccine that expressed tetanus toxin FrC containing a promiscuous MHC II-binding sequence and a p30-OVA peptide that functions in the MHC I pathway was construc
ted. Because FrC-OVA-BV (rBV) is specific to the MHC I pathway, we evaluated its OVA-specific IFN-γ secretion in vivo. The OVA-specific IFN-γ-producing Tcells from spl
enocytes were analyzed using ELISPOT or CD8 + T-cell IFN-γ assays 35 days after the intramuscular injection of rBV, wtBV, FrC-OVA-pVAX1-CAG-MCS or PBS on days 0 and 21
 in mice (Fig. 2a) . As displayed in Fig. 2b , the restimulation of rBV-immunized spleen cells with the OVA peptide resulted in higher levels of OVAspecific IFN-γ comp
ared with those in cells treated with wtBV, FrC-OVA-pVAX1-CAG-MCS or PBS. In the rBVimmunized spleen cells treated with the control peptide HIV-1 Gag, the level of OVA
-specific IFN-γ was decreased to that observed in the wtBV control. On the other hand, as determined by the CD8 + T-cell IFN-γ assay, the rBV, wtBV and FrC-OVA-pVAX1-C
AG-MCS groups showed higher levels of CD8 + T-cell IFN-γ than the PBS control group (Fig. 2c and d) . These results suggest that rBV is more efficient at activating th
e CD8 + T-cell-mediated response than wtBV or FrC-OVA-pVAX1-CAG-MCS groups.
Experiments were performed to verify whether rBV could induce antitumor immunity against established subcutaneous tumors in mice. First, a classic prophylactic vaccina
tion was examined. The experimental design is demonstrated in Fig. 3a . Mice were immunized with rBV, wtBV, FrC-OVA-pVAX1-CAG-MCS or PBS at 35 and 14 days prior to bei
ng inoculated with EG7-OVA cells. The growth of the tumors in the rBV group was inhibited compared with that observed in the PBS control group (no tumors developed), w
```

```
Background: The baculovirus (BV) Autographa californica multiple nuclear polyhedrosis virus has been used in numerous protein expression systems because of its ability
 to infect insect cells and serves as a useful vaccination vector with several benefits, such as its low clinical risks and posttranslational modification ability. We
recently reported that dendritic cells (DCs) infected with BV stimulated antitumor immunity. The recombinant BV (rBV) also strongly stimulated peptide-specific T-cells
 and antitumor immunity. In this study, the stimulation of an immune response against EG7-OVA tumors in mice by a recombinant baculovirus-based combination vaccine exp
ressing fragment C-ovalbumin (FrC-OVA-BV; rBV) was evaluated. Results: We constructed an rBV expressing fragment C (FrC) of tetanus toxin containing a promiscuous MHC
IIbinding sequence and a p30-ovalbumin (OVA) peptide that functions in the MHC I pathway. The results showed that rBV activated the CD8 + T-cell-mediated response much
 more efficiently than the wild-type BV (wtBV). Experiments with EG7-OVA tumor mouse models showed that rBV significantly decreased tumor volume and increased survival
 compared with those in the wild-type BV or FrC-OVA DNA vaccine groups. In addition, a significant antitumor effect of classic prophylactic or therapeutic vaccinations
 was observed for rBV against EG7-OVA-induced tumors compared with that in the controls.
Conclusion: Our findings showed that FrC-OVA-BV (rBV) induced antitumor immunity, paving the way for its use in BV immunotherapy against malignancies.
```

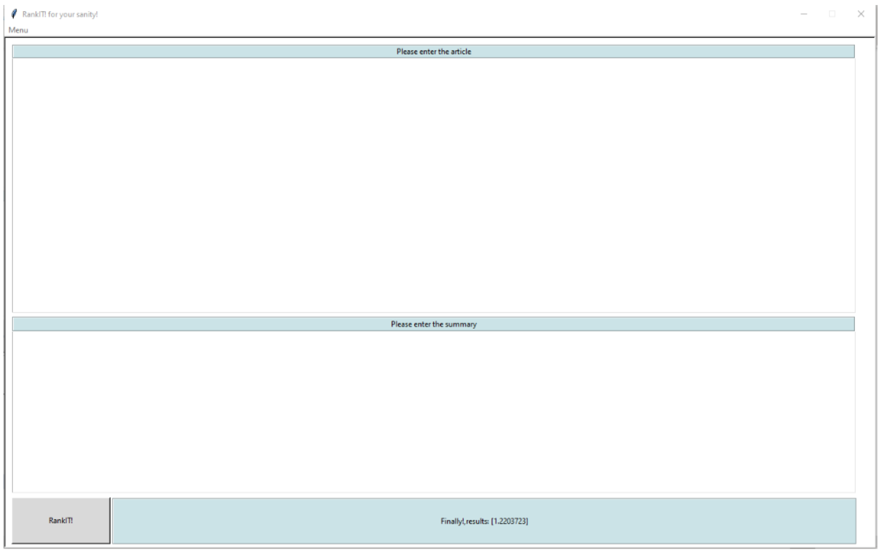
And click “RankIT!”.

We can see updates on the process at the bottom.



starting sentence level spliting for X1

And finally, the results:

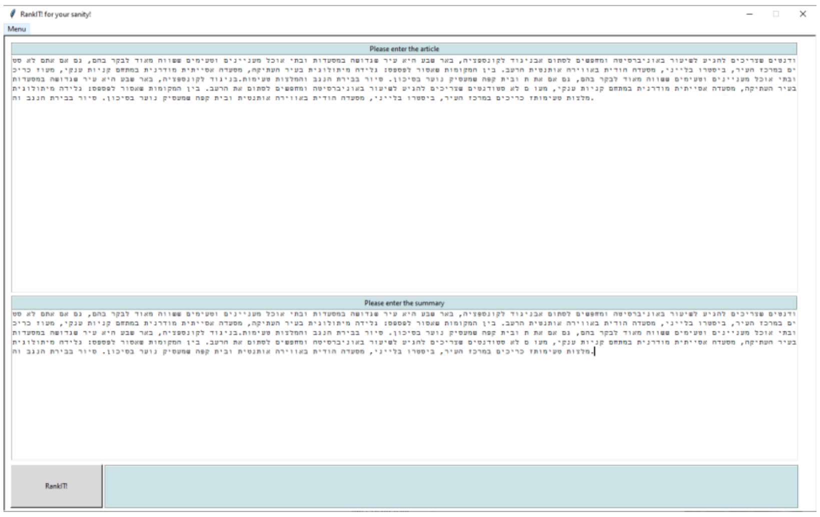


Finally!, results: [1.2203723]

Wow! Look at that. This is a perfect summary; you should read it!

By the way! Beware of your input! For example:

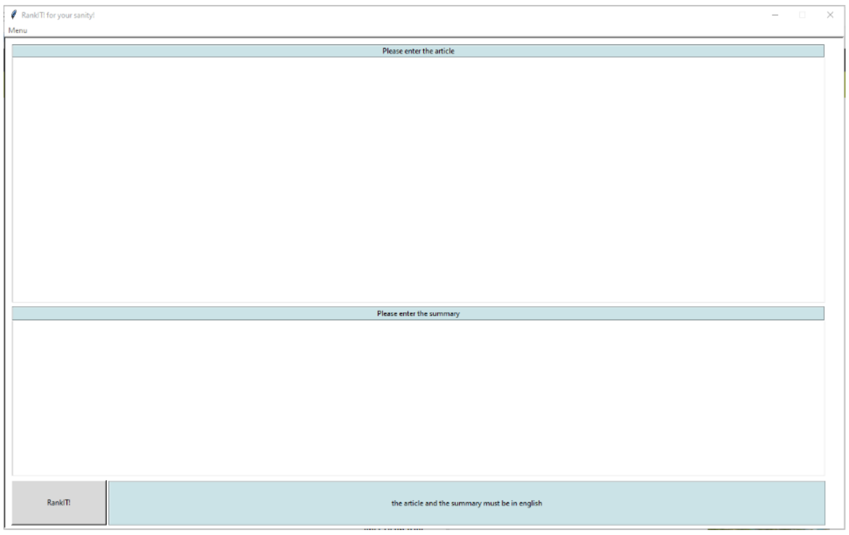
If you try to  put a Hebrew article and summary:



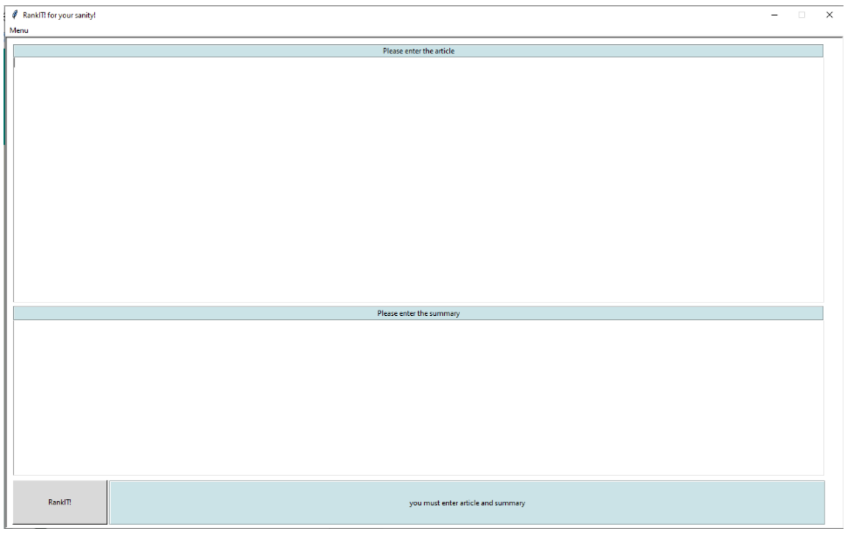
ודנטים שצריכים להגיע לשיעור באוניברסיטה ומחפשים לסתום אבניגוד לקונספציה, באר שבע היא עיר שגדושה במסעדות ובתי אוכל מעניינים וטעימים ששווה מאוד לבקר בהם, גם אם אתם לא סט
ים במרכז העיר, ביסטרו בלייני, מסעדה הודית באווירה אותנטית הרעב. בין המקומות שאסור לפספס: גלידה מיתולוגית בעיר העתיקה, מסעדה אסייתית מודרנית במתחם קניות ענקי, מעוז כריכ
ובתי אוכל מעניינים וטעימים ששווה מאוד לבקר בהם, גם אם את ת ובית קפה שמעסיק נוער בסיכון. סיור בבירת הנגב והמלצות טעימות.בניגוד לקונספציה, באר שבע היא עיר שגדושה במסעדות
בעיר העתיקה, מסעדה אסייתית מודרנית במתחם קניות ענקי, מעו ם לא סטודנטים שצריכים להגיע לשיעור באוניברסיטה ומחפשים לסתום את הרעב. בין המקומות שאסור לפספס: גלידה מיתולוגית
מלצות טעימותז כריכים במרכז העיר, ביסטרו בלייני, מסעדה הודית באווירה אותנטית ובית קפה שמעסיק נוער בסיכון. סיור בבירת הנגב וה.

ודנטים שצריכים להגיע לשיעור באוניברסיטה ומחפשים לסתום אבניגוד לקונספציה, באר שבע היא עיר שגדושה במסעדות ובתי אוכל מעניינים וטעימים ששווה מאוד לבקר בהם, גם אם אתם לא סט
ים במרכז העיר, ביסטרו בלייני, מסעדה הודית באווירה אותנטית הרעב. בין המקומות שאסור לפספס: גלידה מיתולוגית בעיר העתיקה, מסעדה אסייתית מודרנית במתחם קניות ענקי, מעוז כריכ
ובתי אוכל מעניינים וטעימים ששווה מאוד לבקר בהם, גם אם את ת ובית קפה שמעסיק נוער בסיכון. סיור בבירת הנגב והמלצות טעימות.בניגוד לקונספציה, באר שבע היא עיר שגדושה במסעדות
בעיר העתיקה, מסעדה אסייתית מודרנית במתחם קניות ענקי, מעו ם לא סטודנטים שצריכים להגיע לשיעור באוניברסיטה ומחפשים לסתום את הרעב. בין המקומות שאסור לפספס: גלידה מיתולוגית
מלצות טעימותז כריכים במרכז העיר, ביסטרו בלייני, מסעדה הודית באווירה אותנטית ובית קפה שמעסיק נוער בסיכון. סיור בבירת הנגב וה.

you will get the following error:



the article and the summary must be in english for empty input, you will get:



you must enter article and summary

## Test Cases and Scenarios

| # Test Scenario | Test Scenario Description | Test Cases |
|---|---|---|
| **1** | Check the CLI score option. | 1. Check system behavior when an invalid path entered for a summary.<br>2. Check system behavior when an invalid path entered for an article.<br>3. Check system behavior when the article file is empty.<br>4. Check system behavior when the summary file is empty.<br>5. Check system behavior when the article file is empty.<br>6. Check system behavior when the article is not in plain English.<br>7. Check system behavior when the summary is not in plain English.<br>8. Check system behavior when the article and summary are valid, check that it is doing its all functionality, and return a score. |
| **2** | Check the GUI score option. | 1. Check system behavior when the article entered, but the summary does not.<br>2. Check system behavior when a summary is entered but not an article.<br>3. Check system behavior when an article, summary, or both is not in plain English.<br>4. Check system behavior when the article and summary that entered are valid. |
| **3** | Check CLI (re)train option | 1. Check system behavior when only the data path passed.<br>2. Check system behavior when only the type of operation entered.<br>3. Check system behavior when the entered path is not valid.<br>4. Check system behavior when the data folder does not contain valid files (.json).<br>5. Check system behavior when the config file is not present (in case of t operation).<br>6. Check system behavior when all the passed params are valid. |
| **4.** | Check GUI (re)train option | 1. Check system behavior when the entered folder is empty.<br>2. Check system behavior when the data folder does not contain valid files (.json).<br>3. Check system behavior when the config file is not present.<br>4. Check system behavior when all the contents of the folders are valid. |
| **5.** | Check basic preprocessor functionality. | 1. Check that the sentence splitting works fine (returns a list of sentences).<br>2. Check that the encoding works fine, that it returns an array of floats.<br>3. Check that penalty and bad summaries generator works fine and returns the desired scores. |

| | | |
|---|---|---|
| | | 4. Check that the padding process pads into the following shape (? 650,768). |
| **6.** | Check the reading processes. | 1. Check that the reading process (for the train) returns a data frame with the following columns [“X1”,” X2”,” Y”].<br>2. Check that the reading process for (score) works fine, that the text that it gets in the GUI window is the same text that passed into the preprocessor.<br>3. Same as two but for the CLI files. |
| **7.** | GUI tests | 1. Check that the system shuts down the start button when the logic is in work.<br>2. Check that the start button enabled when the system is in IDLE mode.<br>3. Check that the GUI works fine in Linux \ BSD \mac and windows.<br>4. Check all the GUI elements for its size, position, width, and length.<br>5. Check that the window is not resizable.<br>6. Check that the GUI is visible ok(that you can still read all the text) at different screen resolutions. |

## NN Modeling

### Metrics and Measurements

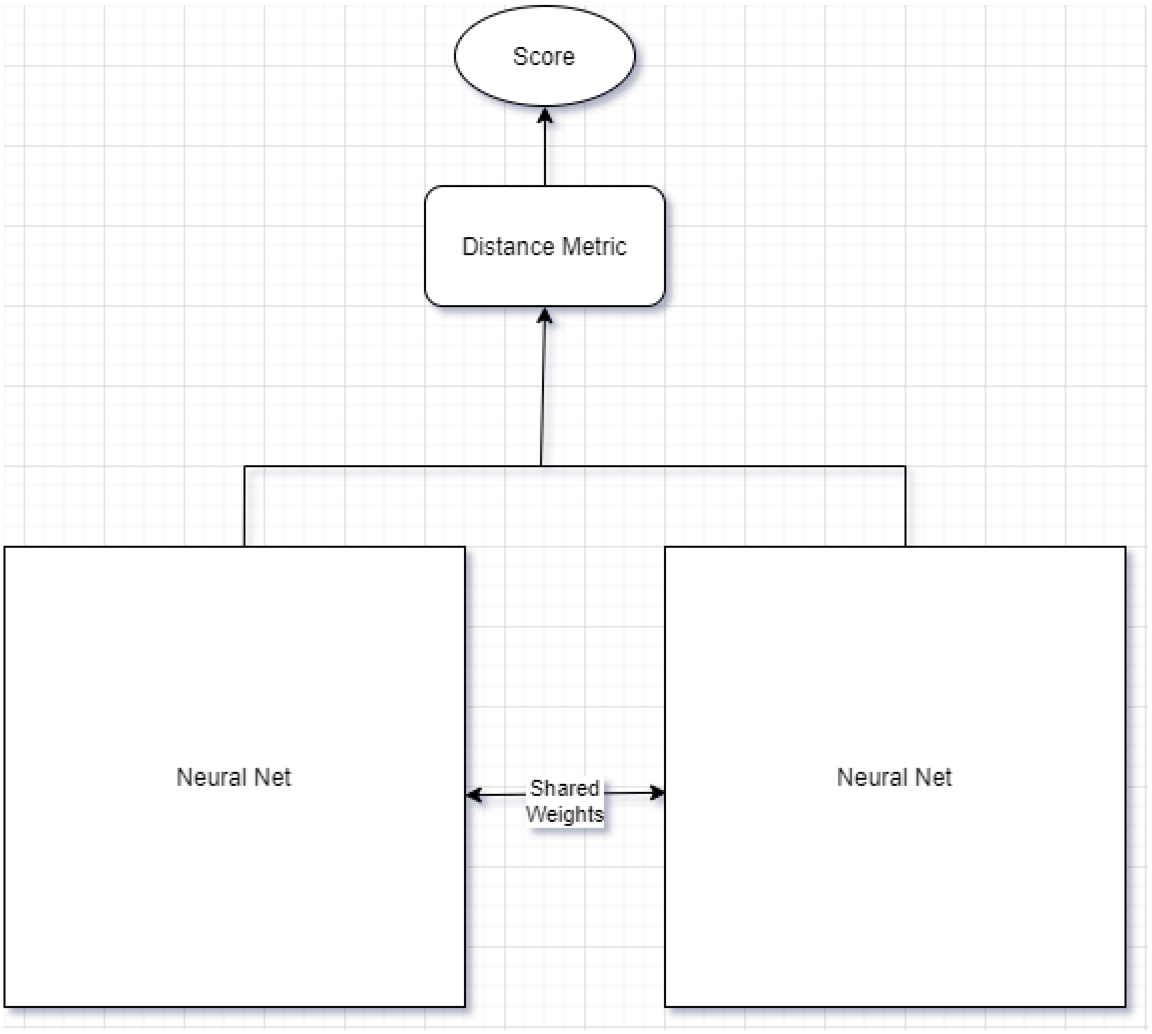
guidelines:

1. I  am taking a supervised learning approach.
2. The model needs to predict a continuous variable, so we are talking about a regression domain.
3. Because of Memory limitations, the data set divided into 2 chunks of 8,000 articles each.
4. For each article, we have generated 0,0.5 and 1 score, as described above. So, there are 24000 samples in each chunk.
5. For model estimation, each chunk divided into a train, dev, and test (17,340,3060,3600) sets.
6. I am using the save best model approach; I am saving the best model concerning val_loss.
7. In some of the experiments because of lack of time, I have used early stopping.
8. Because of lack of time: only the baseline and the model that showed improvement trained on the second chunk.

Metrics:

1. I decided to use MSE cost function as my metric because the domain of the problem and because I wanted to give "high penalties" to higher errors.
2. Additionally, I checked (on the best models only) how the model deals with damaged summaries that cut in the middle and concatenated with duplicated lines or zeros.

## Siamese Networks

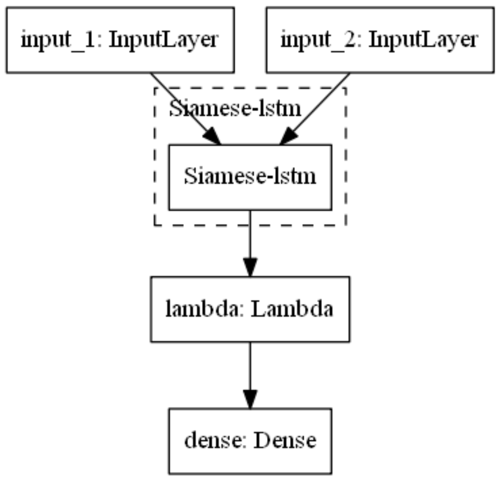


All the experiments based on Siamese networks; the basic idea behind these networks is to learn the same mapping function for the two inputs.

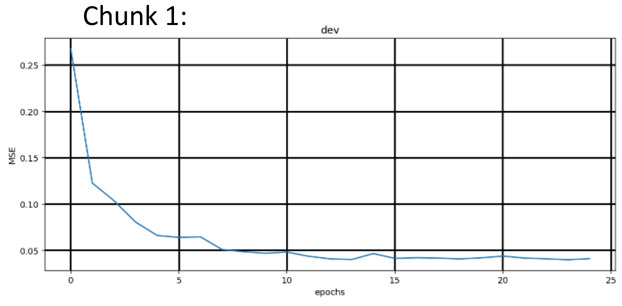
As such, the input of the summary passed to the same architecture that the article passed in.

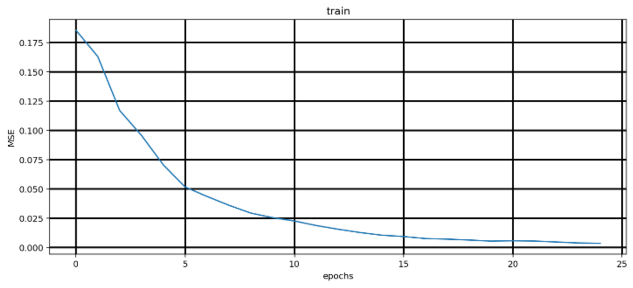
## Baseline

1. Siamese LSTM:
   Bidirectional LSTM with 250 hidden states.
   Bidirectional LSTM with 250 hidden states.
2. Taxi driver distance.
3. Linear Dense.

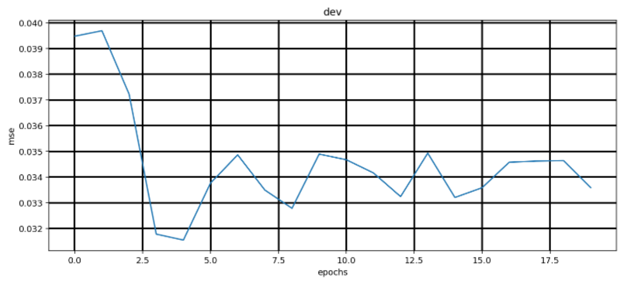


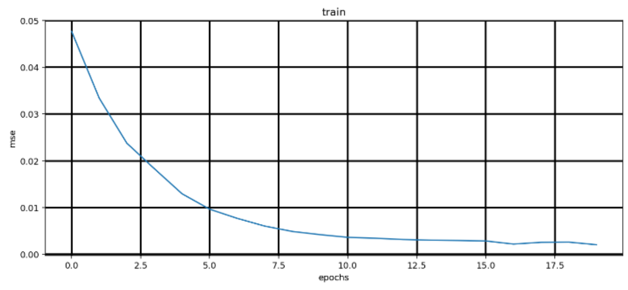
Chunk 1:





Chunk 2:





Mean :

1. Best loss: 0.0028
2. Best dev loss: 0.0358

On duplicated lines of summary: 1767 out of 3600 was with a high score (>0.5).
On concatenation with zeros: 1833 out of 3600 were with a high score (>0.5).
Conclusions:

1. The model overfitted.
2. It fails on damaged data.

## Main experiments

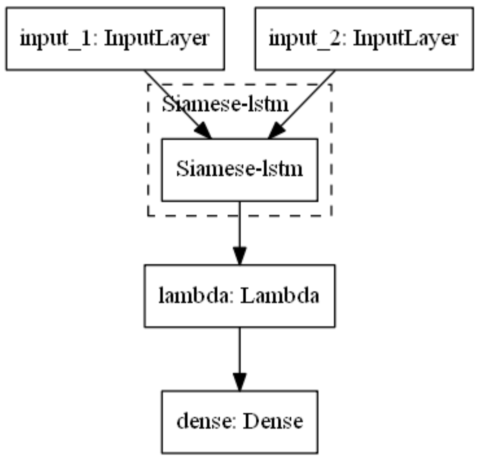
### *The first experiment: dropout regularization and simplification of the model*

to reduce overfitting, I decided to:

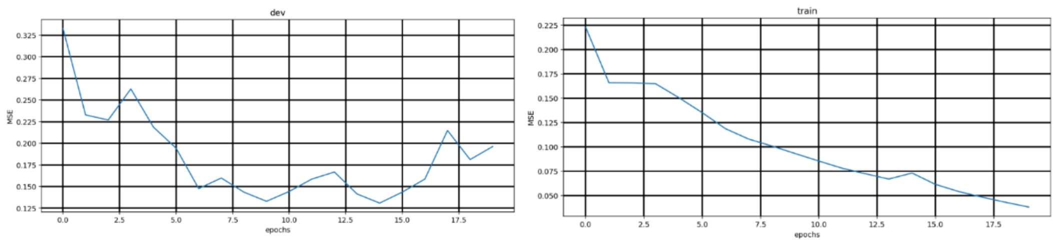
1. Add dropout on the output on the second LSTM layer.
2. To cut the neurons of the second LSTM layer by half.

the architecture:

1. Bidirectional LSTM with 250 hidden states.
2. Bidirectional LSTM with 125 hidden states.
3. Dropout of 0.2.
4. Taxi driver distance.
5. Linear Dense.



Chunk 1:

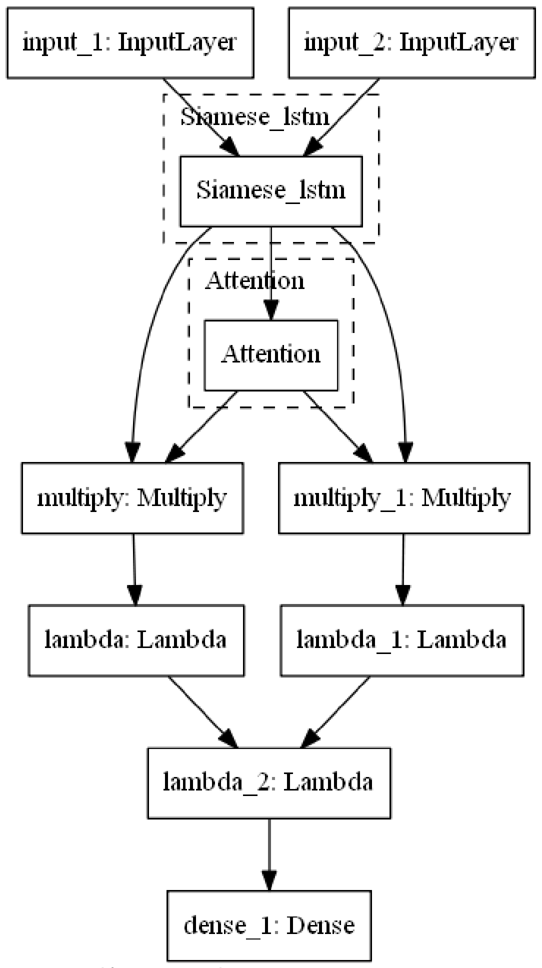


Conclusions: Unfortunately, we cannot see any improvement, quite the opposite.

## *The second  experiment: add attention*

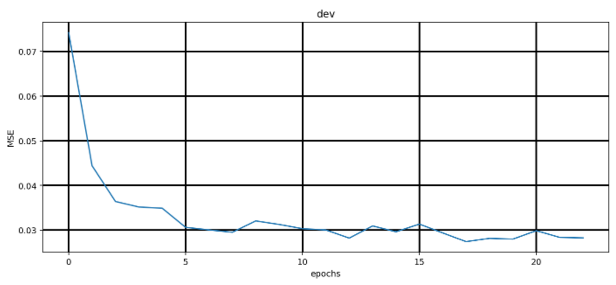
The articles are long! And well, besides this fact, not all the sentences burn equal("All animals are equal, but some animals are more equal than others"); some sentences are essential than others. To help the model to focus, I have decided to add an attention mechanism.

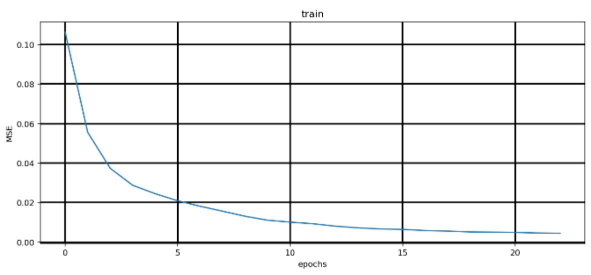
The architecture:



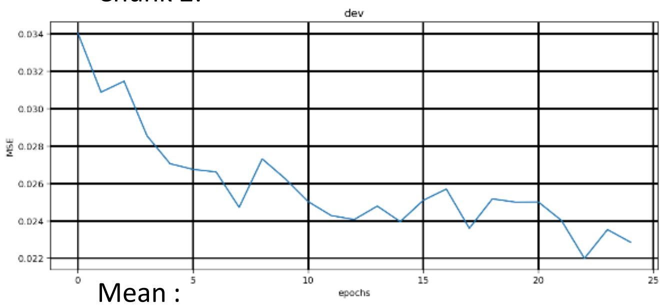
1. Siamese LSTM:
   Bidirectional LSTM with 250 hidden states.
   Bidirectional LSTM with 125 hidden states.
   Dropout 0.2.
2. Siamese Attention
3. Sum(Mul ( Siamese Attention, Attention input))
4. Taxi driver distance.
5. Linear Dense.

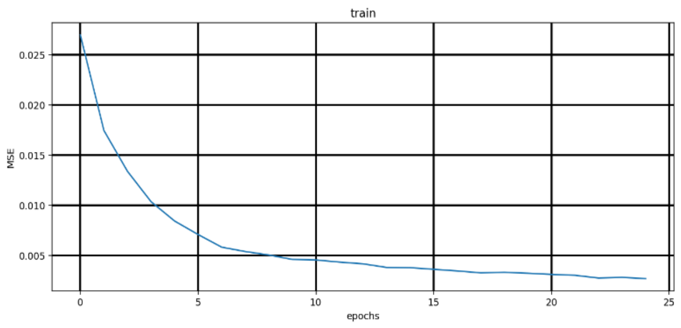
Chunk 1:





Chunk 2:





Mean :

1. Best loss: 0.0035
2. Best dev loss: 0.0247

On duplicated lines of summary: 2 out of 3600 was with a high score (>0.5).
On concatenation with zeros: 1882 out of 3600 was with a high score (>0.5).
Conclusions:

- MSE: we can see improvement. But there is still overfitting.
- On duplicated lines of summary, we can see an improvement; maybe it is because of the importance scoring approach of the attention.
- But unfortunately, the problem with zero concatenated summaries remains, maybe because of the padding with zeros, our model gets confused.

Maybe it will be a good idea to try hyper params tuning and regularization(not only dropout) on this architecture.

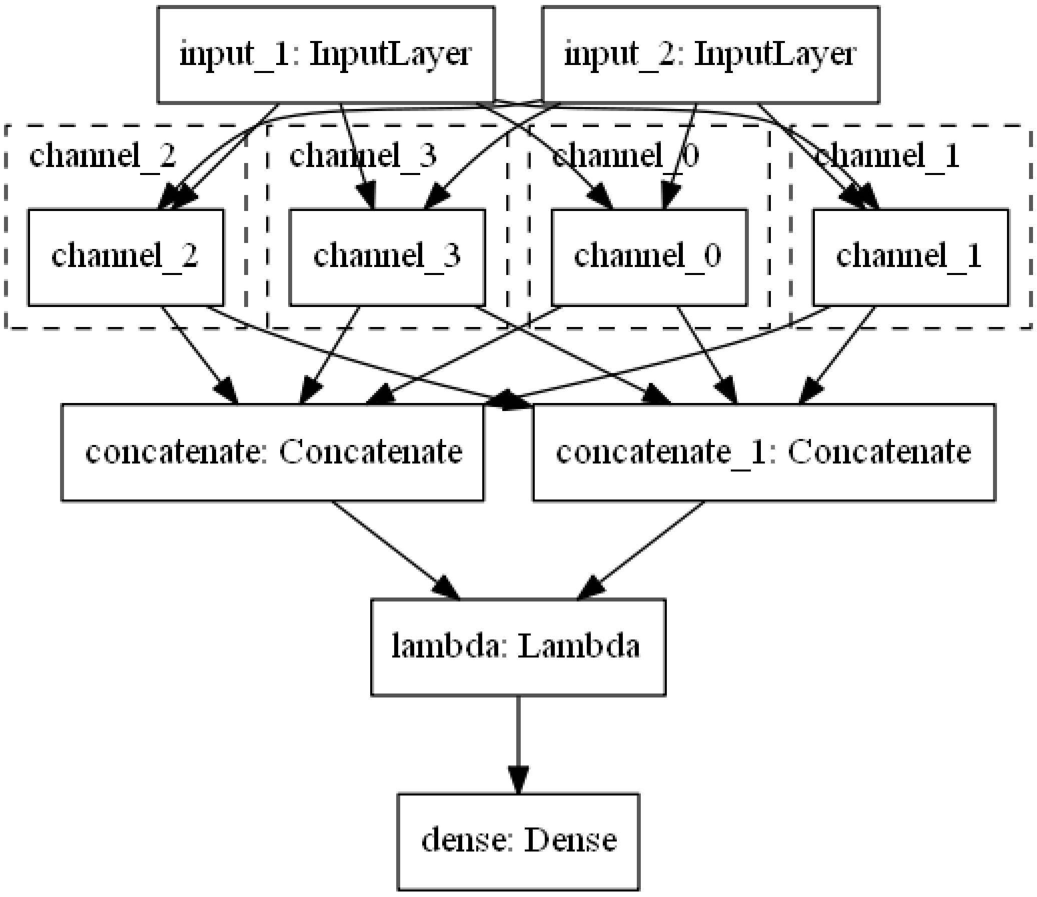
### *The third experiment: multi-channel CNN*

CNN's generally used in computer vision tasks, but in recent years CNN been applied on NLP tasks as well.

CNN is known for its ability to do feature extraction and to grasp local context between features, in our case, between sentences.

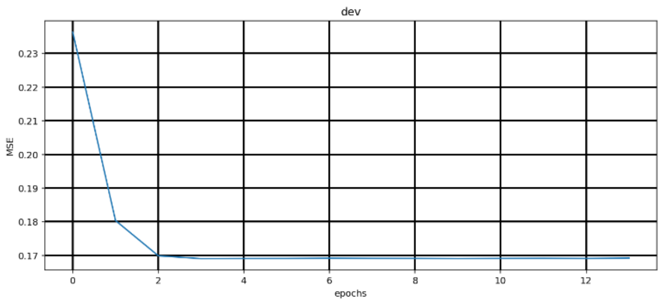
In this experiment, I have tried to apply multi-channel CNN (we can think on each channel as N-gram than been learned).

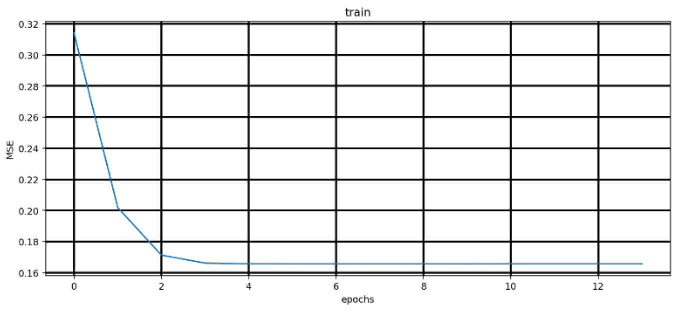
The architecture:



1. <u>4 channels, each channel with 100 filters.</u>
   First, 2-sized kernel.
   Second, 3-sized Kernel.
   Third, 4-sized kernel.
   Fourth, 5-sized Kernel.
2. A combination of the channels
3. Taxi driver distance between the two combinations.
4. Linear Dense.

Chunk 1:





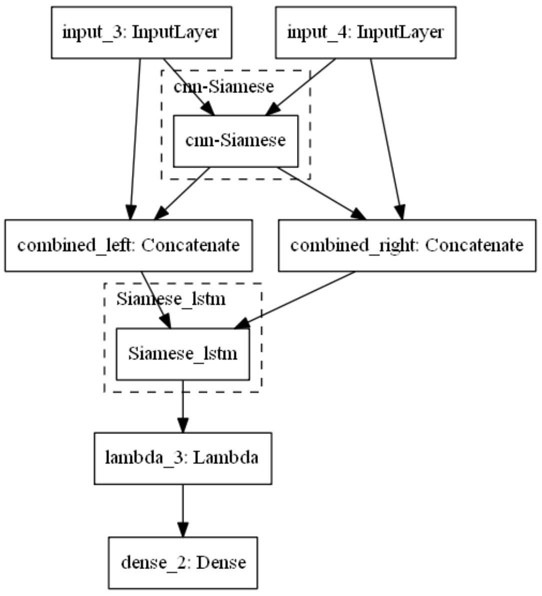
Conclusions:

unfortunately,  this approach did not show improvement at all, quite the opposite. But maybe it will be a good idea to check another combination of different kernels size and different amounts of filters.

### *The fourth experiment: LSTM on top of CNN*

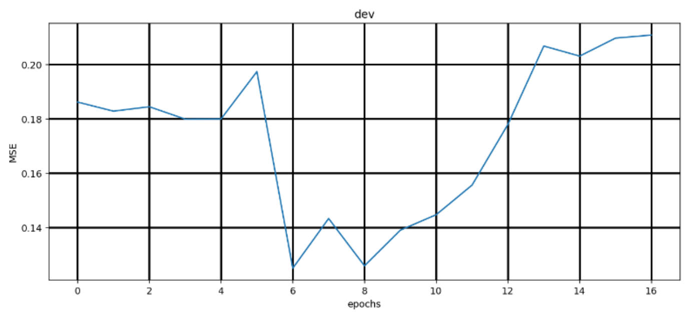
The fourth experiment tries to get advantages of LSTM alongside with advantages of CNN; in this experiment, the input flow through a CNN Siamese to capture the local context, afterward it is concatenated with the original input and then flows through an LSTM Siamese to try and capture the global context.

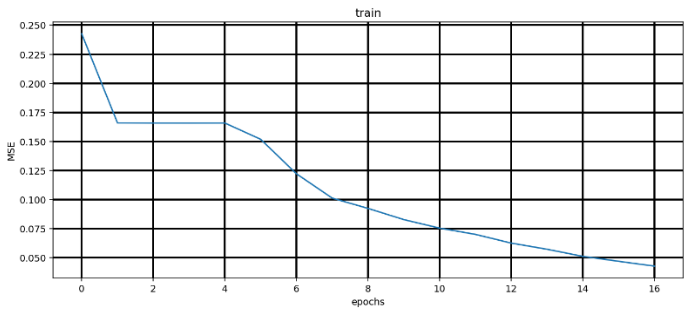
The architecture:



1. <u>Siamese CNN</u>
   Kernel size of 5 and 128 filters.
   Dropout 0.2.
2. <u>Siamese LSTM</u>
   Bidirectional LSTM with 250 hidden states.
   Bidirectional LSTM with 125 hidden states.
   Dropout 0.2.
3. Taxi driver distance.
4. Linear Dense.

Chunk 1:



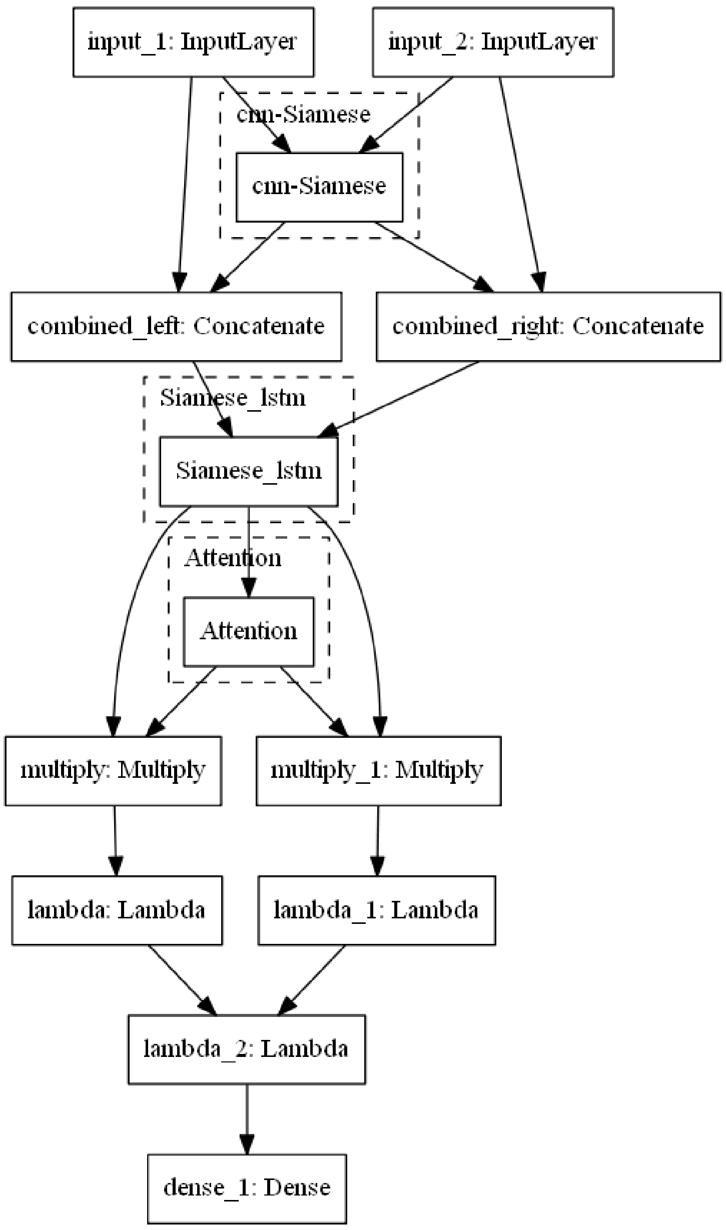


Conclusions: we cannot see any improvement.

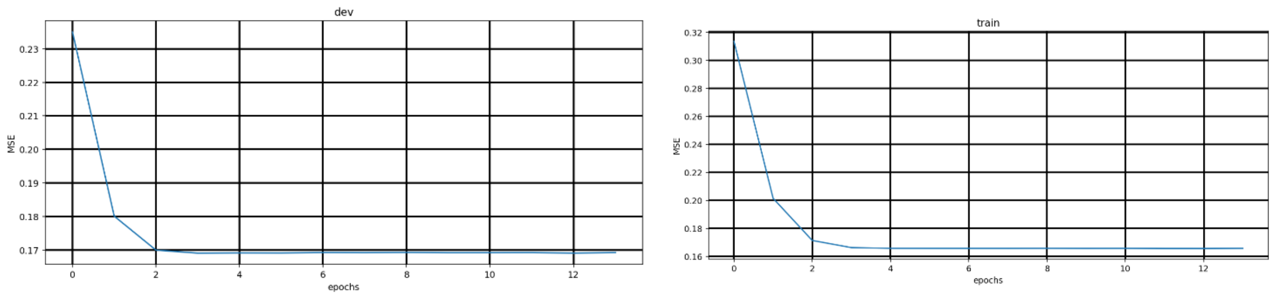
### *The fifth experiment: LSTM on top of CNN with attention*

The same as the fourth but with attention on the LSTM.

The architecture:



Chunk 1:



Conclusions: still without improvement.

## *The last experiment: reward-penalty network*

Till now, I tried to get better MSE results, but I  neglected the damaged summaries problem; in this experiment, I took the best architecture(LSTM with attention) and, based on this, trained a duplicate network. I have trained this network on one chunk and generated the following scores:

- Score 1 for permutations of the golden summaries.
- Score 0.5 for golden standard summaries that cut in the middle and padded with zeros.
- Score 0.3 for golden standard summaries that cut in the first 1\3 of the summary and padded with zeros.
- Score 0 for golden standard summaries.

So, in prediction mode, the system passes the input through the scoring network that gives rewards, and through a penalty network that gives penalties for damaged structure, the final score is a subtraction between the two results.

The model was trained on 23120 samples, validated on 4080 samples, and tested on  4800 samples:

The results of the penalty network:

Best train loss: 0.0068.

Best dev loss: 0.0120.

On test set: 0.0147.

The results are not bad, but we do have overfitting, and it might be better.

We may say that the penalty network understands what damaged summary is.

But, unfortunately, the combination between the reward (second experiment) and the penalty network did not give the expected improvement.

A follow-up experiment is taking place to get the best of this idea.

## Conclusions and future ideas

Just to recap, some of the experiments gave decent results; the best experiment was the LSTM-attention, its MSE cost on the test was:

On chunk1: 0.0216  On chunk2: 0.0203   mean: 0.02095

Not bad! But it can be better.

Some things can be done:

1. Train on more data and better clean the data.
2. Train on more epochs, different batch sizes, and another optimizer.
3. Grid-search for Hyper-params tuning.
4. Put Regularizations on the weights (in addition to the Dropout regularization).
5. Change the architecture completely. The truth is that we have 2 major problems with our architecture. The first: we are manually giving scores to bad summaries; this approach is problematic because the scores are an approximation.
   And the second problem is that distance is a relative thing.
   For example: when I am in class, I am much closer to my classmates than to my family, but when I am at home, I am much closer to my family, the same logic applied with text, some summaries are closer to the article than others.
   The solution to these problems is to change the architecture to a triplet-based one.

link for main experiments: models and results

## Bibliography and References

[1] Taken from https://en.wikipedia.org/wiki/Wikipedia:Size_comparisons

[2] Taken from https://liinwww.ira.uka.de/bibliography

[3] Taken from https://arxiv.org

[4] Taken from https://www.aminer.cn

[5] Taken from http://www.academia.edu/about

[6] N, Anastasios Venetsanopoulos, and Nicolaos Karayiannis. *Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications.* Springer Science & Business Media, 1992.

[7] Goldberg, Yoav. *Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies).* A Publication in the Morgan & Claypool Publishers series, 2017.

[8] Max Welling; Thomas N. Kipf." Semi-Supervised Classification with Graph Convolutional Networks." *Published as a conference paper at ICLR 2017*..2017 .

[9] Schmidt, Robin M. "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview." 2019.

[10] Jeffrey L. Elman." Finding Structure in Time." *Cognitive Science*.1990 .

[11] Schmidhuber, Jürgen, and Sepp Hochreiter . "Long Short-Term Memory." *Neural Computation*, 1997: 1735-1780.

[12] Bengio, Yoshua, KyungHyun Cho, Caglar Gulcehre, and Junyoung Chung. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[13] Schuster, Mike, ıKuldip Paliwal." Bidirectional recurrent neural networks." *IEEE Transactions on Signal Processing*,2673-2681 ,1997.

[14] Yoon, Kim. "Convolutional Neural Networks for Sentence Classification." EMNLP, 2014.

[15] Sutskever ,Ilya, Oriol Vinyals, ıLe Quoc." Sequence to sequence learning with neural networks." *NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*.3104-3112 :2014 .

[16] Bahdanau, Dzmitry; KyungHyun, Cho; Bengio, Yoshua. "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE." ICLR'15 Conference,2015.

[17] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz; Polosukhin, Illia. "Attention Is All You Need". NIPS'17 31st Conference on Neural Information Processing Systems, 2017.

[18] Kodratoff, Yves. "Knowledge discovery in texts: a definition, and applications, Foundations of Intelligent Systems, " *Springer*, 1999.

[19] Litvak, Marina, and Natalia VanetikMulti-lingual*al Text Analysis: History, Tasks, and Challenges.* World Scientific, 2019.

[20] Potdar, Kedar, Taher Pardawala, and Pai Chinmay. "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers." i*nternational Journal of Computer Applications (0975 – 8887) Volume 175 – No.4, October 2017*, 2017.

[21] Manning, Christopher, Hinrich Schuetze. *Foundations of Statistical Natural Language Processing.* The MIT Press, 1999.

[22] Manning, Christopher, Prabhakar Raghavan, Hinrich Schutze. *Introduction to information retrieval.* Cambridge University Press, 2008.

[23] Mikolov, Tomas, Quoc Le, ıIlya Sutskever." Translation, Exploiting Similarities among Languages for Machine." *http://arxiv.org/abs/1309.4168*,2013 .

[24] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation." *Association for Computational Linguistics*, 2014: 1532–1543.

[25] Bojanowski, Piotr, Grave Edouard, Joulin Armand, ıTomas Mikolov." Enriching Word Vectors with Subword Information." *https://arxiv.org/abs/1607.04606v1*,2016 .

[26] Bojanowski, Piotr , Grave Edouard , Joulin Armand , Tomas Mikolov. "Bag of Tricks for Efficient Text Classification." *https://arxiv.org/abs/1607.01759v1*, 2016 .

[27] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina." BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *https://arxiv.org/pdf/1810.04805.pdf* ,2018.

[28] Betlagy,lz; Lo Kyle; Cohan Arman. SciBERT: A Pretrained Language Model for Scientific Text " *https://arxiv.org/abs/1903.10676*,2019.

[29] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable Feature Learning for Networks." *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[30] Daixin, Wang, Peng Cui, and Wenwu Zhu. "Structural Deep Network Embedding." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016: 1225-1234.

[31] Gholamrezazadeh, Saeedeh, Mohsen Amini Salehi, and Bahareh Gholamzadeh. 2009. "A Comprehensive Survey on Text Summarization Systems." *2009 2nd International Conference on Computer Science and its Applications.*

[32] Veksler, Yael, Marina Litvak, and Natalia Vanetik. 2019. "EASY-M: Evaluation System for Multilingual Summarizers." *Proceedings of Multiling.*

[33] Chin-Yew, Lin. 2004. "ROUGE: A Package for Automatic Evaluation of Summaries." *Text Summarization Branches Out* 74-81.

[34] Giannakopoulos, George, and Vangelis Karkaletsis. 2011. "AutoSummENG and MeMoG in Evaluating Guided Summaries." *TAC 2011 Workshop.*

[35] Eyal, Matan , Tal Baumel, and Michael Elhadad. 2019. "Question Answering as an Automatic Evaluation Metric for News Article Summarization." *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* 3938-3948.

[36] Papineni, Kishore, Roukos Salim, Todd Ward, and Wei-Jing Zhu. 2002. "BLEU: a Method for Automatic Evaluation of Machine Translation." *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002* 311-318.

[37] Wang, Lucy Lu, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Michael Kinney, Ziyang Liu, William. Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. "CORD-19: The Covid-19 Open Research Dataset." ArXiv (2020): n. pag.

[38] Steinberger, Josef, and Karel Jezek. "Evaluation Measures for Text Summarization." Comput. Informatics 28 (2009): 251-275.

www.sce.ac.il

**קמפוס באר שבע**
ביאליק 56, באר שבע 84100

**קמפוס אשדוד**
ז'בוטינסקי 84, אשדוד 77245