**Beer Sheva Campus**

**Summary Evaluation Platform**

**Master's Degree Final Project**

**Submitted in the Department of Software Engineering**

**Sami Shamoon College of Engineering**

**Name: Sergey Mordeev**

**ID: 321175879**

**Project Leader: Dr. Marina Litvak**

Contents

[Abstract 3](#_Toc3627070)

[1. Introduction 3](#_Toc3627071)

[2. Text Summarization Task 4](#_Toc3627072)

[3. Summary Evaluation 5](#_Toc3627073)

[4. Metrics 6](#_Toc3627074)

[4.1. Implementation Forewords 7](#_Toc3627075)

[4.2. Rouge 9](#_Toc3627076)

[References 10](#_Toc3627077)

# Abstract

We present here a platform for summary evaluation metrics. The purpose of such platform is to be a useful tool for the researchers in the field of text summaries. Although summary evaluation metrics come with appropriate tools (scripts) and the result could be analyzed within different data manipulation tools as Microsoft Excel or even with languages as R but taking the evaluation process under one umbrella will simplify and automate the analysis of the summary systems by our believing.

# 1. Introduction

For a person working within some field the time has a crucial meaning. No one of us wants spending a time on the same tasks again and again. The purpose of many systems is to automate such the process. For instance, no one expects that bank teller will run SQL queries to update the deposit. As well, no one expects from us working only with command line while solving our problems on computer. Today we have a plenty of tools for automating different kinds of tasks. We believe that researchers in text summarization should not be an exception. Thus, a tool for automating researcher tasks could help her for the better understanding and evaluating the results.

The word “platform” in software world has different meanings. We think that most of us agree that when one says “cross-platform software” her meaning is that the software is not dependent on the current operating system and/or chosen hardware. The expectations in such case are that the software will work similarly with a variety of operating systems or hardware. From the other hand, when one says, for instance “we have an enterprise office platform software” the meaning in such case is not related neither to operating system nor to the APIs of such office software. The meaning is the group of programs for solving a particular business’ issues. As another example, when the programmers say that they chosen a platform that might mean that the meaning of such is a software library or framework. The word “platform” in software is highly context dependent.

To make things less ambiguous, the word platform in our case should be defined. When we say platform, we mean:

1. Concrete GUI application that evaluates several metrics for the text summaries
2. Command line tool if one needs to automate his tasks
3. A software library that could be used by 3rd parties

To be honest, the project had started as the library in mind and latter it morphed into much more massive GUI application. Hence, we believe that it was a good migration since the library and the API were tested by ourselves. We cannot argue that the core library for metric evaluation is one of the best. However, we can argue that for us it was a right vision both for the code management and modeling the process of evaluation itself. The core library will be explained within this text for those who would like to use the platform as the library.

The text will be divided into a couple of part. The first part will describe the task of summary evaluation in general. The second part will give the overview of the program internally. The third part will explain the usage of the GUI application. The last part will be the experiments with the system. Although we are trying to not make the interdependencies within these parts but for better understanding of the system reading it in novel form give the better picture.

# 2. Text Summarization Task

The purpose of this section is to give an overview of what summarization process is from the flying bird point of view. Such need exists because the platform has been developed under several assumptions. No software in the world exists without taking assumptions. This piece of software should not be an exception too. From the one hand each of us wants software for all possible case in life. From the other hand, the practice says that not taking assumptions can finally bring the programming task either to the never-ending story or since it is unclear what restrictions are the software becomes bloated; program features are unclear; the code base in pursuit of generalization becomes unmaintainable and full of bugs. Thus, restrictions and/or assumptions should be taken to produce a valuable software within a given time limit.

Text summarization task depicts itself as taking a summary from the given source(s). More formally the summary could be defined as a *text that is produced from one or more texts, that contains a significant portion of the information in the original text(s), and that is no longer than half of the original text(s)* and text summarization as *the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or user) and task (or tasks)* (Lloret, 2008). Text summarization in general could be divided into the single document or multi document. As well, the language should be considered too when we are talking about automatic text summarization (i.e. produced by machines) (Lloret, 2008). It is also important to understand what formats automatic summarizer can produce and consume. These facts are important since they directly influence on the summary evaluation.

It should be mentioned that summarization task is vital in current over-information (Giannakopoulos, 2008). For instance, the development of World Wide Web brings us to state where for the person it is not possible to read every page completely (Lloret, 2008). Today “simple” search in web search engine will result in short summary, see Figure 1. (It seems that it is a simplest possible summary, but it is a summary. For Google Scholar it looks like it takes an abstract of an article)



Figure 1.

As it is already said in this section those details are reflected as assumptions and crystalized as requirements in the developed platform directly. For instance, the platform expects that all documents either an original document(s) or the summary itself are a simple text encoded by UTF-8 and separated each one in its file. From the one hand it may require the preprocessing and/or after the summarization but from the other hand simplicity matters. Processing XML or other marked up format will require additional complication even for the end user to prepare documents in the needed format. As well, UTF-8 today is the de-facto standard for simple text. Our opinion that more exotic as marked up language or those that based on text script processing as PDF or office documents (or even rare Unicode encodings) bring complexity both for the end user and text processing.

# 3. Summary Evaluation

In broad term of thinking summary evaluation could be a difficult problem because of human subjectivity about what good summary is. As the result it leads to disagreement whether one summary is better than another (Giannakopoulos, 2008). From the other hand, human judgment is the best possible judgment about the summary since humans eventually should read those summaries and have an idea what they are about.

As in many science fields the evaluation is a score which is given to the object and the method which gives such score is called a metric. From mathematical point of view, it could be represented as a function from text domains (with or without additional parameters) to ℝ domain:

(1)

In many cases score range is restricted to be or normalized to have such range.

It highly dependent on the nature of the metric what expectations are. For the metrics which try show how a summary “good” or “bad” it is expected that these metrics are statistically correlated to the human judgments for those summaries. For instance, it could be a Pearson’s correlation (Lin, 2004). However, one might be interested not only how good or bad a summary is but, say, how good a summarizer is. For us as humans, it is not only important how well the information compressed into summary, but it is also important how this information is presented. Manner and matter for us have a meaning. If it is very hard to read a summary such summary might be useless eventually. Thus, we could be interested in the readability of those summaries. In (Elena Lloret, 2019) such question is asked, and it is implemented by the software platform. The readability metrics’ results are useless without any reference and as a reference for readability metrics of the original documents.

# 4. Metrics

It is already mentioned that a text metric could thought about as a function (1) from text domain to ℝ. However, for the metrics that should decide how much the summary are built on the pattern of comparisons between two texts. Although the text comparison task is interesting, but it is hard and has many applications in Natural Language Processing (Leonidas Tsekouras, 2017). The common pattern requires representing text in intermediate form, Figure 2.



Figure 2.

The second text of comparison in many cases is a human summary. Such summaries are called ‘ideal’ or ‘model’ summaries. Furthermore, because of the subjectivity of what good summary is, several summaries should be considered. As well, it should be considered what the strategy is for the multiple comparisons. For the ROUGE metric(s) (Lin, 2004) there are two strategies: either the average is taken, or the best result is taken. By our observations, the best result is not so popular. Therefore, the only strategy implemented by the evaluation platform is an average among all comparisons.

The real art of those comparisons is the intermediate form and the following algorithms based on these forms. Strictly speaking, the algorithm defines (requires the usage of) such form. Thus, for the platform it has been obligated to define a way to hold and produce various forms in generic manner.

On the other hand, obviously, the automatic summary text comparison with ideal summary is not only the way to have a metric for summary. For example, the platform also proposes variety of readability metrics for automatic summaries. Thus, one can have an overview how much considered system influences on readability. For the platform it was chosen to reflect the difference (i.e. mathematical minus) between the original text metric value and generated summary text metric value. A combination of such simplest form with some visualization techniques gives us truly beautiful results for grasping analysis without deep dive into results.

The next sections will give a brief overview of the chosen metrics for implementation by our platform. The one might wish as much as possible metrics; however, such effort would require much time with negative consequences on other planned features.

## 4.1. Implementation Forewords

While it seems a bit early to dive into the implementation details, but we should give an overview picture about implementation details to make the following text clearer. Within our platform we decided to implement several metrics for summary evaluation. Among them there are ROUGE metrics. Originally, ROUGE metrics were implemented in Perl language.

We decided to implement our platform in Java language. Specifically, it is of version 1.8. The decision was dictated firstly due to high popularity of this language. See Table 1 for top 5 languages on the time of writing this text according to TIOBE index (TIOBE, 2019). As well, according to Figure 3, this language is a leadership for almost two decades.

| **Mar 2019** | **Mar 2018** | **Change** | **Programming Language** | **Ratings** | **Change** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 |  | Java | 14.880% | -0.06% |
| 2 | 2 |  | C | 13.305% | +0.55% |
| 3 | 4 | change | Python | 8.262% | +2.39% |
| 4 | 3 | change | C++ | 8.126% | +1.67% |
| 5 | 6 | change | Visual Basic .NET | 6.429% | +2.34% |

Table 1.

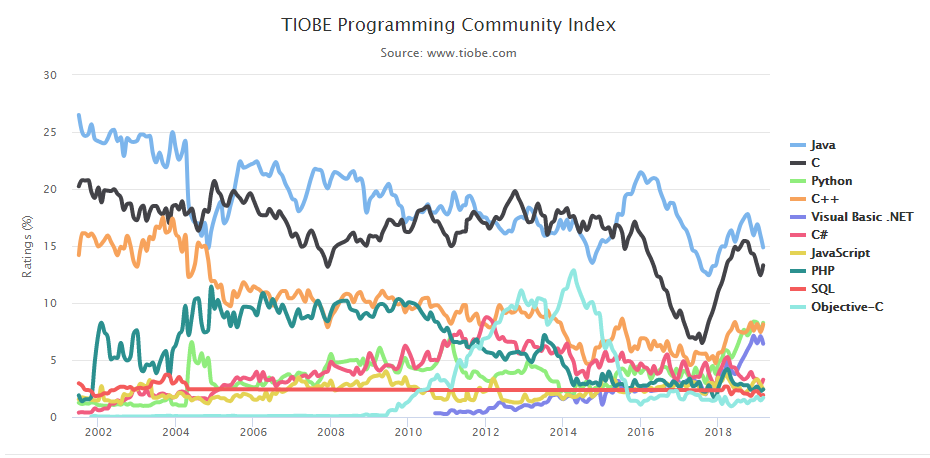


Figure 3.

The second reason for choosing the Java language was a fluent knowledge of it by authors because of professional activity of its authors for a decade. The third reason is that used by authors NLP libraries are written in Java. More specifically, for NLP processing it was chosen Stanford CoreNLP (Stanford, 2019). Fourthly, despite of common believing about the slowness of this languages, today this language (specifically, HotSpot JVM (Oracle, 2019)) is the one of fastest available choices (Debian, 2019).

Thus, the platform almost completely written in Java language. As it was mentioned, the reference implementation of ROUGE metrics is written in Perl. Therefore, we needed to decide how to integrate it within the platform. The decision might have seen non-standard, but we decided to port it to Java language. From the one hand it seems time consuming and, maybe, even useless. From the other hand, many NLP projects today start with Java or even when they do not start with, integrate it with Java is not an issue because of huge availability of integration tools and large community. Additionally, Perl by today standards is outdated language. Understanding the Perl code is known to be an issue by itself. We believe that the port will help for future researchers/implementors better understanding of the source code of ROUGE metrics.

Furthermore, possibly it does not seem to be an issue, but Perl is times slower than Java (Debian, 2019). One might say that it is not an issue for today’s machines (according to (Debian, 2019) it has similar measurements to the Python language). However, we still believe that time matters, and faster solutions are better.

With all said, we assume that the reader is familiar with one of C derived modern language as Java, C#, etc. If it will be required, the code snippets will be written in Java language. However, this knowledge is not a must.

## 4.2. Rouge

Since it has seen the light, the ROUGE metrics have a wide acceptance in the community due to it statistical correlation with human judgements. The ROUGE methods use statistical measures of similarity based on n-gram of words. The intuition behind the metric is that if two text considered having the similar meaning then they must share similar words or phrases (Giannakopoulos, 2008). One can find detailed explanation of the metrics in (Chin-Yew Lin, 2003). From our side, we should mention that all these metrics share the same “score” structure. Each metric has a precision, recall and F-measure. What precision and recall exactly meant by each metric to be the reader can find also in (Chin-Yew Lin, 2003).

For example, for the ROUGE-N metric between two text the number of common occurrences of n-grams (which tokenized words) is calculated:

**int** nGramHits(Map<String, Integer> peerGrams, Map<String, Integer> modelGrams) {  
 **int** hits = 0;  
 **for** (String modelGramToken : modelGrams.keySet()) {  
 **if** (!**"\_cn\_"**.equals(modelGramToken) && peerGrams.get(modelGramToken) != **null**) {  
 **int** peerHits = peerGrams.get(modelGramToken);  
 **int** modelHits = modelGrams.get(modelGramToken);  
 hits += Math.*min*(peerHits, modelHits);  
 }  
 }  
 **return** hits;  
}

Where peerGrams dictionary is the token n-gram against the number of its occurrences in the peer (i.e. automated summary generated by some system). Analogously, modelGrams is the n-grams against the number of its occurrences of the model. The model in this context is the human summary. Thus, the calculated value is the number of n-gram tokens shared by two texts. (The “\_cn\_” token in both mappings is an internal token indicating the total number per text on n-grams.) In order to calculate the precision, we need to divide the number of shared n-grams on the number of tokens in the peer (automated summary). Accordingly, the recall is the number of shared occurrences divided by total number of n-grams in model (human summary).

(2)

(3)

As well, by default the platform (and original the ROUGE metrics) produce the harmonic mean, i.e. the measure. It is defined as (4) with .

, (4)

Internally, the platform uses the α-based measure (Sasaki, 2007). I.e. . The equation (4) becomes:

(5)

Thus, if one wishes to calculate different F-measures she should give an appropriate α. For instance, for β=0 (, i.e. precision) α=1; for β=2 () α=0.2, etc. However, the ability to change the α parameter is only available when the platform is used as a library.

We should mention that we have ported all ROUGE metrics to the Java language: RougeN, RougeS, RougeW, RougeL. We are not familiar with the similar effort. We believe it

# References

Chin-Yew Lin, E. H. (2003). Automatic Evaluation of Summaries Using N-gram.

Debian. (2019). *The Computer Language Benchmar Game*. (Debian) Retrieved March 2019, from https://benchmarksgame-team.pages.debian.net/benchmarksgame/

Elena Lloret, T. V. (2019). Are Better Summaries also Easier to Understand? Analyzing Text Complexity in Automatic Summarization. In *Multilingual Text Analysis: Challenges, Models And Approaches* (pp. 337-371). World Scientific.

Giannakopoulos, G. K. (2008). Summarization System Evaluation Revisited: N-Gram Graphs.

Leonidas Tsekouras, I. V. (2017). A Graph-based Text Similarity Measure That Employs Named Entity Information.

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries.

Lloret, E. (2008). *Text summarization: an overview.*

Oracle. (2019). *Java SE HotSpot at a Glance*. (Oracle) Retrieved March 2019, from https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136373.html

Stanford. (2019). *Stanford CoreNLP*. (Stanford) Retrieved March 2019, from https://stanfordnlp.github.io/CoreNLP/

TIOBE. (2019). *TIOBE Index*. (TIOBE) Retrieved March 2019, from https://www.tiobe.com/tiobe-index/