Hanoi University of Science and Technology
School of Information and Communication Technology

Project Report On
Paraphrase Detecting Web App

Project I

IT3910E – 709162

Submitted by

Nguyễn Việt Hoàng. SID: 20194434

Under the guidance of

Assoc. Prof. Lê Thanh Hương

School of Information and Communication Technology

Hanoi, January 2022

Abstract

As the number of academic articles rises at an unprecedented speed, it is becoming more difficult to verify the originality of a submitted document. A number of algorithms, programs and tools have been developed to help human in identifying plagiarism. However, most of them are not accurate nor efficient enough. This project's goal is to build a plagiarism identifying tool that satisfies both the requirement of speed and accuracy. Along with the tool, an interface would also be designed so that the production can be used by common users.

In this project, a Machine learning model is trained to classify whether a sentence is a rephrase of another sentence or not, which achieves an accuracy of higher than 96%. Additionally, a program is developed, which takes in an input document, search for articles online, and compare them in order to check for paraphrasing. An interface is also designed so that the application can be used by common users.

All the project works, along with the guidance to set up the application, are included in the repository: HoangNV2001/Paraphrase-checker (github.com)

Acknowledgements

I would like to send the most sincere appreciation to Assoc. Prof. Lê Thanh Hương. Professor Hương is the one who suggested this topic, and provided many helpful advises throughout the semester so that this project could complete.

Table of contents

I. Introduction

Plagiarism checking is one of the important problem in academic society. Our approach to identify plagiarism is through paraphrase detecting. This project inherits from the paper Paraphrase Identification in Vietnamese Documents by Bach et al. in 2015[1] . In the referenced work, the authors presented the method for text preprocessing for paraphrase identification, and the use of Support Vector Machine model for the classification task. Their research brought in promising results, however, the work is still limited in academic field, not fully developed for practical uses.

The contributions of this project are:

1. From the foundation of the referenced work, we made some additions and modifications in text preprocessing and similarity measures in order to improve the performance of classification model, and speed up the comparing process.
2. Python program is developed. The program takes a document as input, then compare the document with other Internet contents using Google search engine. Inverted index is applied to increase the speed of comparing documents.
3. A web application is created using Flask framework. The web application provides an interface for the user to interact with the Python program.

The rest of the paper is organized as follows. Section II describes the dataset used for training the classification model, the similarity measures, and the model's performance evaluation. Sections III presents the methods applied in the document comparing process, which consists of inverted indexing. The presentation of web application is included in section 4, which contains the diagrams describing the application, some sample snapshots of the interface, and the usage demo. The section 5 provides the analysis on the results. The project is concluded in section 6, where plans for the future are also provided.

II. Classification Model

First, a Machine Learning (ML) model is built to classify paraphrase sentence pairs. The Support Vector Machine is chosen due to its advantages over other ML models[2]. We use the scikit-learn[3] library for creating the Support Vector Classification.

1. Data

The data that is used to train the SVM model is vnPara – a paraphrase corpus for Vietnamese[1]. The corpus has 3000 Vietnamese sentence pairs, corresponding with each pair is the label:

- 0: paraphrase
- 1: non-paraphrase

1500 of the pairs were labeled as 0 (paraphrases), and the other 1500 were labeled as 1 (non-paraphrases). The preprocessing of the data is as follow:

Given a set of n labelled sentence pairs $\{(S_{1,1}, S_{1,2}, y_1),...,(S_{n,1}, S_{n,2}, y_n)\}$, where $(S_{i,1}, S_{i,2})$ is the i^{th} sentence pair, y_i is the label (with value of 0 or 1). Each sentences pair $(S_{i,1}, S_{i,2})$ is converted to a feature vector $\overrightarrow{v_i}$, whose values are scores returned by similarity measures that indicate how similar $S_{i,1}$ and $S_{i,2}$ are. The vectors and the corresponding categories $\{(\overrightarrow{v_i}, y_1), ..., (\overrightarrow{v_n}, y_n)\}$ are given as input to the supervised classifiers, which learns how to classify new vectors $\vec{v}$, corresponding to unseen pairs of sentences (S1, S2).

2. Similarity measures

Below are the nine similarity measures that are applied to each sentence pair (S1, S2):

a. Cosine similarity

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x}.\vec{y}}{\|\vec{x}\| \times \|\vec{y}\|}$$

With $\vec{x}$, $\vec{y}$ are vectors of binary values. i.e. x_i and y_i are 1 or 0, depending on whether or not the corresponding word occurs in the first or the second sentence, respectively.

b. Scaled Damerau-Levenshtein distance

While orthodox Damerau–Levenshtein distance operates on character level, the implementation in this project is on word level. Furthermore, after the calculation of the Damerau-Levenshtein distance, it is then scaled so that it would have the range of 0 to 1 by the following formula:

$$Scaled\_Damerau\_Levenshtein(S_1, S_2) = \frac{Damerau\_Levenshtein(S_1, S_2)}{\max\left(l(S_1),\ l(S_2)\right)}$$

With $l(S_i)$ is the number of word in the sentence i.

c. Jaro-Winkler distance

The implementation of Jaro-Winkler distance would be similar in the work Paraphrase Identification in Vietnamese Documents by Bach et al.

d. Scaled Manhattan distance

Given a sentence pair (S_1, S_2), for each sentence S, a set of words and its number of occurrence in that sentence is created. Let the set X correspond to S_1, set Y correspond to S_2, which means each x_i in X, y_i in Y shows how many times each one the distinct words occurs in one of two sentences. The Manhattan distance and Scaled Manhattan distance is calculated as follows:

$$Manhattan\_distance(X, Y) = \sum_{i=1}^{n} |x_i - y_i|$$

$$Scaled\_Manhattan(X, Y) = \frac{Manhattan\_distance(X, Y)}{\sum_{i=1}^{n} \max\left(x_i\ ,\ y_i\right)}$$

e. Scaled Euclidean distance

Similarly to Manhattan distance, assume we have a sentence pair (S_1, S_2), and two corresponding sets (X,Y). Then the formula for the Euclidean and Scaled Euclidean distance would be:

$$Euclidean\_distance(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$Scaled\_Euclidean(X,Y) = \frac{Euclidean\_distance(X,Y)}{\sqrt{\sum_{i=1}^{n}\max(x_i\ ,\ y_i)^2}}$$

f. Scaled Bi-gram

These are some notable properties of this implementation of distance using Bi-gram:

- Works on word level, not character level.
- In each sentence, every n adjacent words compose a n-gram
  - ⇨ Which means a bi-gram consists of two adjacent words in the sentence.
- Treating bi-grams as words, the Scaled bi-gram distance is the same as Scaled Manhattan distance

g. Scaled Tri-gram

The only different between Tri-gram and Bi-gram is that each Tri-gram consists of three words, instead of two.

h. Dice coefficient

Let X and Y be the sets of unique words of two sentences S_1 and S_2, correspondingly. Then the Dice coefficient for the sentence pair (S_1 ,S_2) would be calculated as:

$$Dice\_coef = \frac{2\ \times\ |X\ \cap Y|}{|X| +\ |Y|}$$

i. Jaccard coefficient

With the same (S_1,S_2) and (X,Y) as in Dice coefficient, we have:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

3. Evaluation

The dataset is shuffled then divided into 5 folds. Afterward, the 5-fold cross validation test is performed. The performance metrics are accuracy score and f1 score. The two scores are averaged over the five folds.

In the paper Paraphrase Identification in Vietnamese Documents by Bach et al., the author have built several SVM models, with different preprocessing methods, which had different performances. The best model's result from their work to was selected to compare with the performance of our trained SVM model.

	Accuracy (%)	**F1**
Referenced paper's best model	89.10	86.77
Our SVM model	**97.34**	**97.39**

Table 1: Comparison of models' performances

It can be seen that our model has significant better result than the referenced model.

III. Methodology

Detailed information about the implementations of packages, classes, and modules will not be included in this report. Instead, the methodologies that are applied in order to enhance the performance of the application would be discussed.

Inverted Indexing

The naïve approach of comparing each sentence in the input document to every other scraped sentences results in very high computational cost as well as

low efficiency, due to the fact that most of the scraped sentences are not actual paraphrases.

With the intention of only comparing the sentence pairs where the two sentences share some words in common, we decided to apply the Inverted Index method.

Each document, both the inputted and the scraped, would have an inverted index table, which is in the form of a Python dictionary[4]. For each document's inverted index table, each key represents a word that appears in the document, and it's value is a list that contains the indices of the sentences that contain the word.

Sample document
Fuzzy Wuzzy was a bear. Fuzzy Wuzzy had no hair.

Word	Sentence
a	1
bear	1
fuzzy	1,2
had	2
hair	2
no	2
was	1
wuzzy	1,2

Figure 1: an example of Inverted Index table.

In our new strategy, the sentences in the input document would only be compared to the scraped sentences that have more than 4 words in common.

IV. Web Application

After creating a classification model and constructing related modules, the next step is to build an application that takes an input, which consists of a document that needs to be checked for paraphrasing, the keyword for the document. The output of the application would be a report that comprises of a list of sentence pairs that have high change of paraphrasing of each other, and a pie chart that shows the proportion of the input document that are identified as paraphrase from other sources. The framework that we use to develop the web application is Flask[5].

1. Sequence diagram

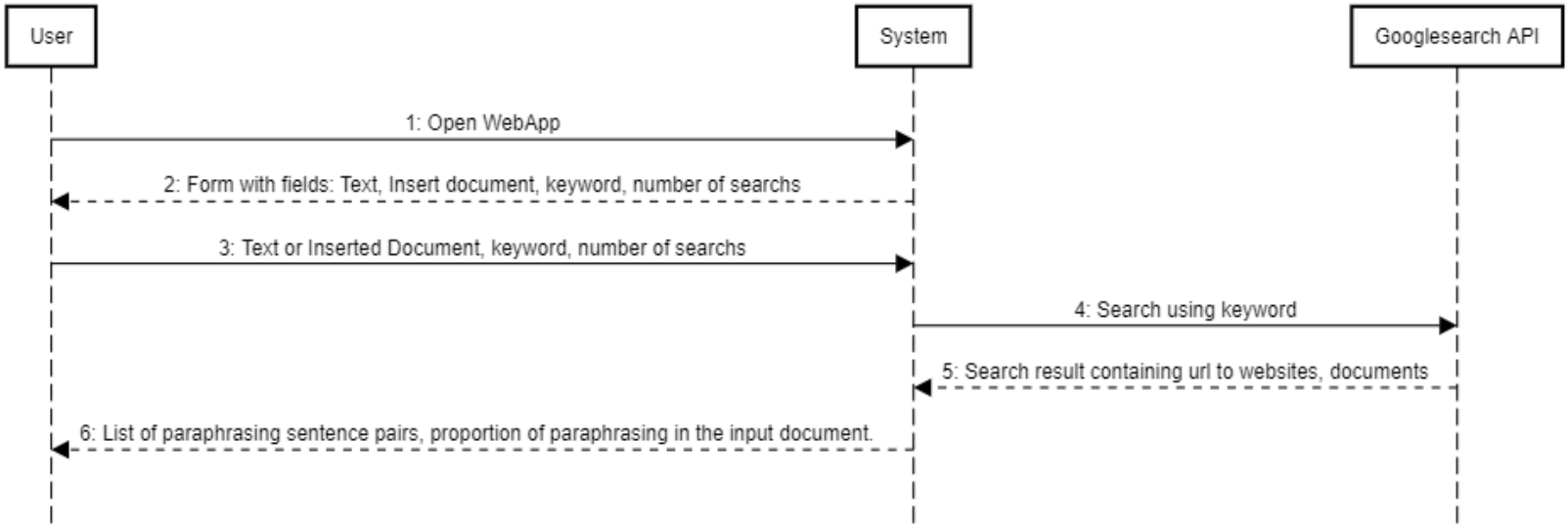

Figure 2: Sequence diagram of the Web application.

2. Activity diagram

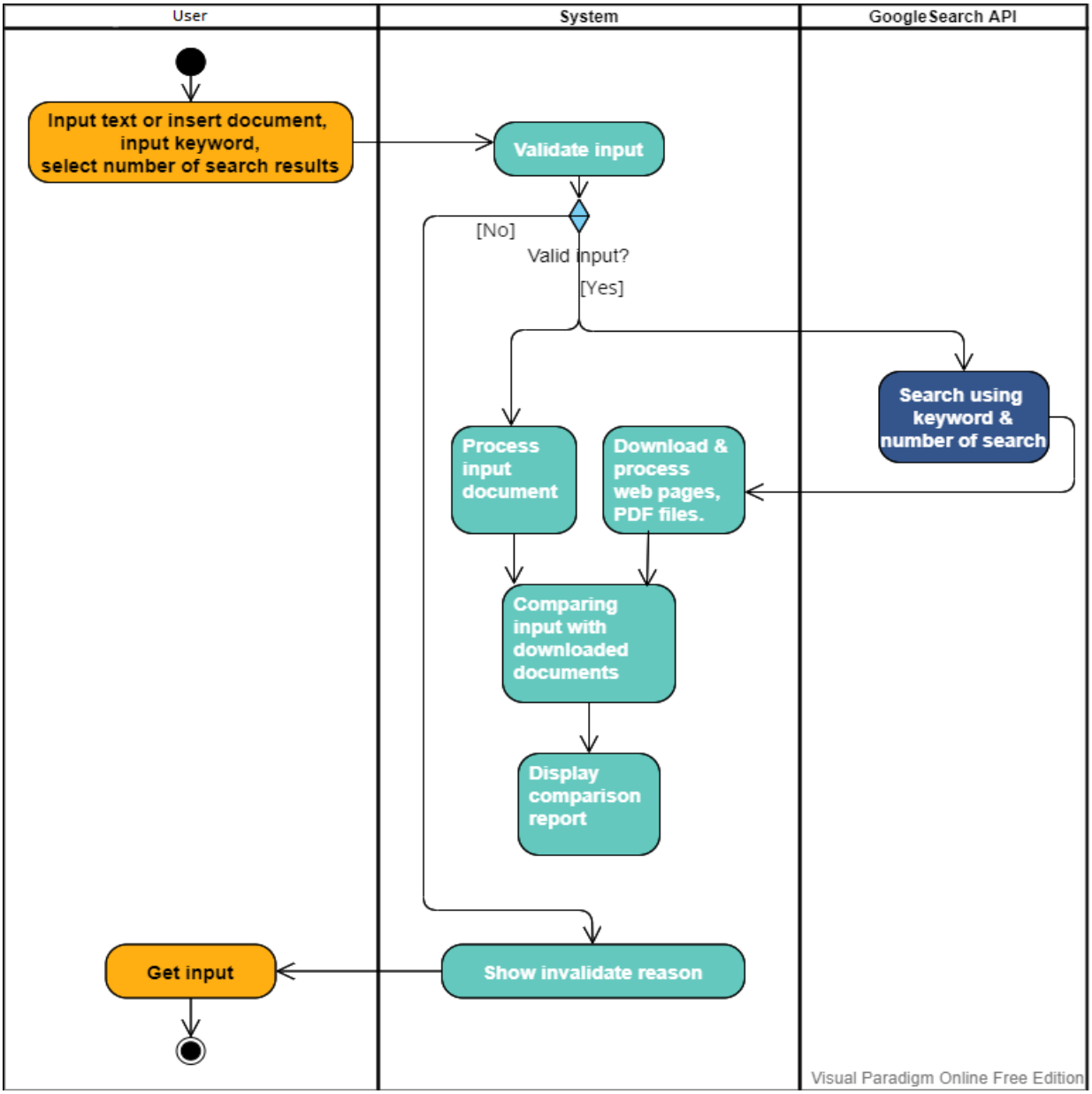

Figure 3: Activity diagram of the Web Application.

3. Interfaces & demo

- Below is the main page interface, which will appear when user open the web page:

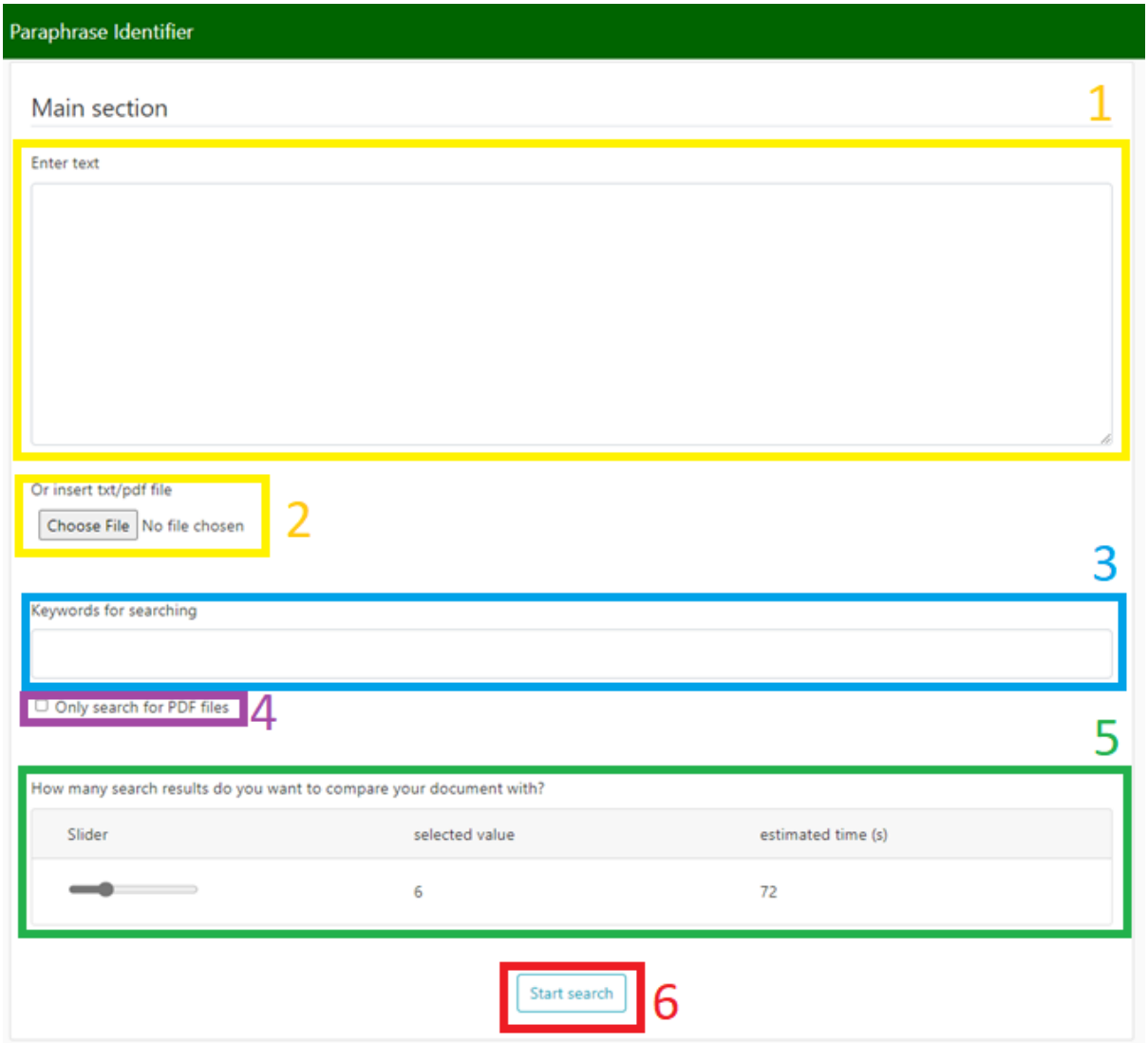

Figure 4: The main page.

- User input the document that needs to be check by either enter the text into Section 1, or insert the document through the button in Section 2.

  Note: Section 1 and 2 cannot be both filled nor empty.

- Next, the user enter the keyword that best summarize/ present the document content into Section 3. This keyword will be used to search for documents/webpages on Google search.

  Note: Section 3 cannot be empty.

- If the user want to only compare with PDF files (which mean to exclude websites from the search), tick the check box in Section 4.
- Downloading and comparing with 1 document on average takes 12 seconds. The number of default value for number of search results to download and compare is 6, with the maximum value is 20. This value can be changed by the user by using the slider in Section 5.
- When the form-filling is completed, press the "Start search" button in Section 6 to signal the system to start take in and process the input.
- Afterward, the waiting interface will appear as below:

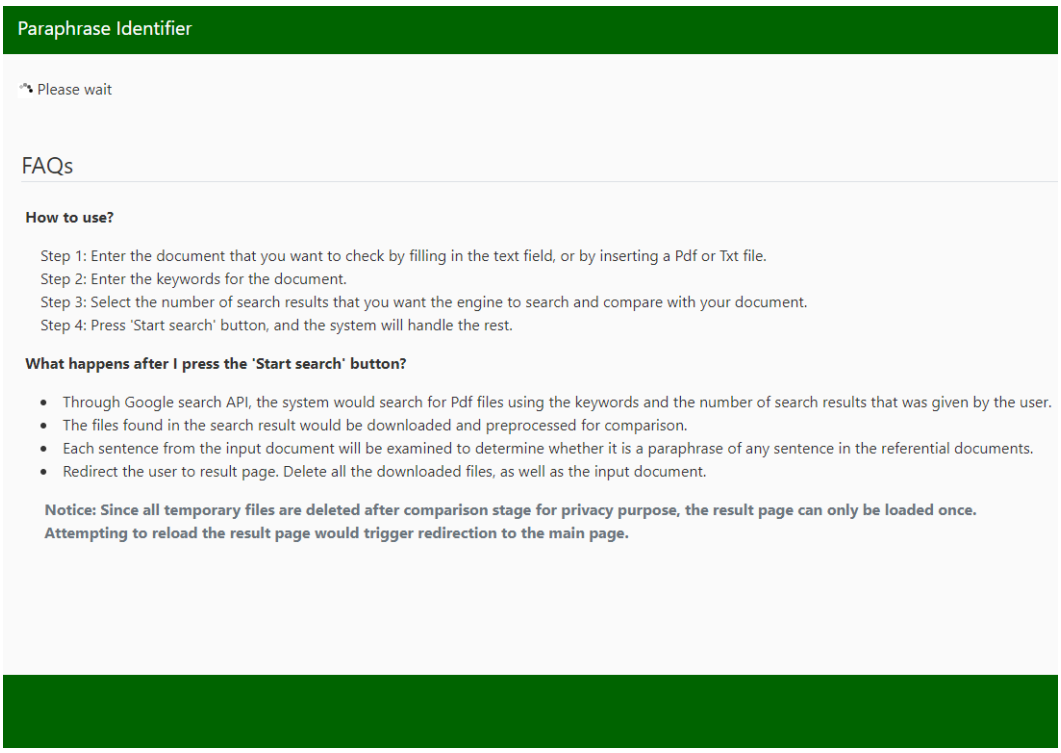

Figure 5: Waiting interface.

- When the processing is completed, the webpage will be redirected to the result page. One example of the result page is as follow:

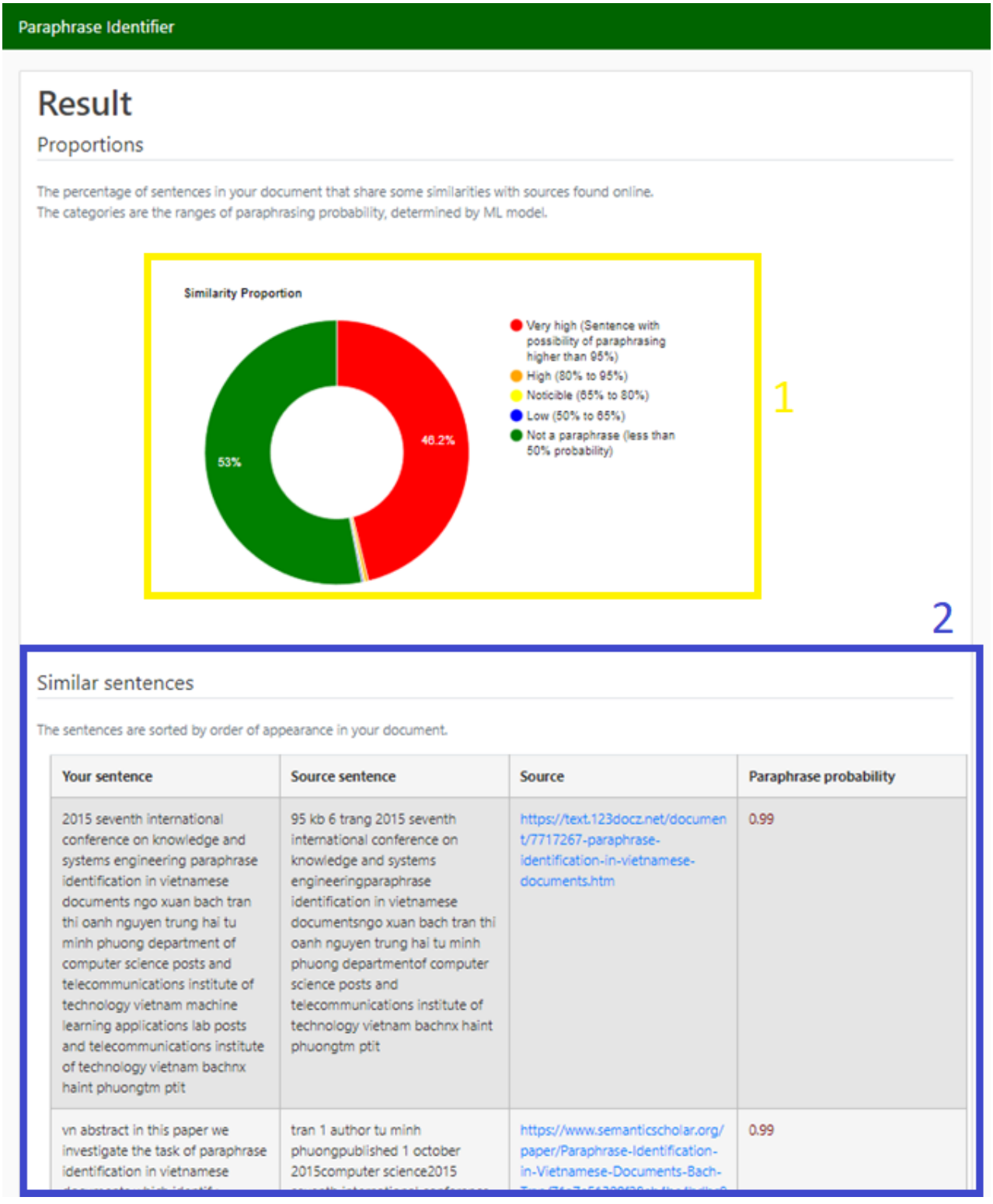

Figure 6: Result page.

There are 2 main sections in the result page:

- Section 1 show the percentage of input document that has high probability of paraphrasing from an online source.
- Section 2 is a table, in which each row is a sentence pair that has high chance of paraphrasing - the first column contains the sentence in the input document, the second column contains the sentence found from the search, in the third column is the link to the webpage/document that contains the source sentence, the fourth columns shows value of the probability of paraphrasing determined by the classification model.

V. Experimental Results

Below is a part of the comparison report. The proportional pie chart and the source URLS field is not included, since we focus on the classification of each sentence pair.

No.	Your sentence	Source sentence	Paraphrase probability
1	in the 1980s knowledge became the focal point of ai research	knowledge based systems and knowledge engineering became a major focus of ai research in the 1980s	0.98
2	850 million was invested in the fifth generation computer project	the money returns the fifth generation project edit in 1981 the japanese ministry of international trade and industry set aside 850 million for the fifth generation computer project	0.59
3	their goal at the time was to make machines think like	their objectives were to write programs and build machines	0.94

	human which included programs or machines that could hold conversations translate languages understand pictures and do logical reasoning like human beings	that could carry on conversations translate languages interpret pictures and reason like human beings	
4	the us s response was to triple the investment in ai between 1984 and 1988	darpa responded as well founding the strategic computing initiative and tripling its investment in ai between 1984 and 1988	0.85

Table 2: Some sentence pairs that are classified as paraphrases.

We can see that the model classifies sentence pairs that are on paraphrasing-borderline as paraphrases. In personal opinion of the author of this paper, whose focus is on the model's recall rather than precision, this behavior is expected and acceptable.

VI. Conclusions

In this project, from the foundation of the work by Bach et al., we have made some additions and modifications in text preprocessing and similarity measures, which resulted in improvement in the performance of classification model. Also, we have developed a Python program that takes an input document, then compare the document with other Internet contents using Google search engine . Inverted index has been applied to increase the speed of comparing documents. Finally, a web application has been created.

In the future, to improve the performance of out method, we plan to include semantic information from synonym dictionaries. Another approach that we would like to experiment on is to implementing deep learning models for paraphrase classification. On the other hand, the size of the training dataset used in this project

is relatively small, with only 3000 sentence pairs. We believe that enriching the training dataset would enables the use of more complex models, providing better generalization, and ultimately better performance on real test cases.

References

[1] Xuan Bach, Ngo & Tran, Oanh & Hai, Nguyen & Phuong, Tu. (2015). Paraphrase Identification in Vietnamese Documents. 174-179. 10.1109/KSE.2015.37.

[2] David Meyer, Friedrich Leisch, Kurt Hornik, The support vector machine under test, Neurocomputing, Volume 55, Issues 1–2, 2003, Pages 169-186, ISSN 0925-2312, https://doi.org/10.1016/S0925-2312(03)00431-4.

[3] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[4] Dictionary Objects — Python 3.10.1 documentation

[5] Flask Documentation (2.0.x) (palletsprojects.com)