UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
SAIMA SULTANA

DÉTECTION DE PHRASES SIMILAIRES : UTILISATION DES N-GRAMMES DE CARACTÈRES

JUIN 2017

UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

Cette thèse a été dirigée par :

Ismaïl Biskri	Université du Québec à Trois-Rivières
Directeur de recherche, Maîtrise.	Institution à laquelle se rattache l'évaluateur

Jury d'évaluation de la thèse :

Ismaïl Biskri	Université du Québec à Trois-Rivières
Directeur de recherche, Maîtrise	Institution à laquelle se rattache l'évaluateur
Boucif Amar Bensaber	Université du Québec à Trois-Rivières
Professeur, Maîtrise	Institution à laquelle se rattache l'évaluateur
Mhamed Mesfioui	Université du Québec à Trois-Rivières
Professeur, Maîtrise	Institution à laquelle se rattache l'évaluateur

Thèse soutenue le Avril 28, 2017

ACKNOWLEDGEMENTS

I would like to devote my cordial thanks to the people who have given me their full support and contribution for developing my thesis paper as well as accomplishing my whole academic work.

I also want to thank my supervisor Ismail Biskri, Professor in the Department of Applied Mathematics and Computer Science for his time, effort as well as availability throughout the completion of this research work. Special thanks to the Professor Éric Poirier in the Department of Modern Languages and Translation for providing me an opportunity to explore myself into professional work which will be very beneficial in my future carrier.

I would like to thank my beloved family for their support and care throughout my studies. Finally, I would like to thank those Professors who helped me a lot for correcting as well as evaluating my thesis paper.

RÉSUMÉ

De nos jours, la capacité à détecter des phrases similaires s'avère une caractéristique fondamentale des applications de lecture et d'analyse de textes, car celle-ci permet de catégoriser, détecter, résumer et traduire l'information présente à l'intérieur de ceux-ci. Parallèlement, les médias sociaux et internet jouent un rôle de plus en plus important en tant que sources d'information, mais les limites technologiques présentes ne permettent pas une utilisation optimale de ceux-ci.

Plusieurs approches ont été proposées pour améliorer l'analyse, la lecture et la récupération d'information. Parmi ces multiples options, certaines s'avèrent basées sur des modèles symboliques et logiques, tandis que d'autres sont fondées sur des cooccurrences numériques et empiriques. Finalement, plusieurs autres possèdent un modèle de reconnaissance reposant sur l'utilisation de formules algébriques. De façon générale, toutes ces approches utilisent des outils spéciaux pour dynamiser leur modèle afin de faciliter la détection de similarités telles que les règles grammaticales, la syntaxe, les dictionnaires en ligne et les moteurs de recherche. Néanmoins, l'utilisation de ces outils peut complexifier l'usage du modèle, le rendant ainsi difficile d'utilisation. De plus, ces approches peuvent parfois voir leurs performances réduites lors de la mesure de similarités dépendant du contexte dans lequel les données se retrouvent.

Dans ce manuscrit, nous présentons une nouvelle méthode afin de détecter les similarités entre deux textes contenant un nombre élevé de mots. Cette méthode met en relief l'utilisation des n-grammes de caractères, la distance entre n-grammes de caractères ainsi que la mesure de la cooccurrence avec les coefficients Jaccard, Dice, Overlap, Cosine et Simple Matching.

Dans notre méthode, les dictionnaires en ligne et les moteurs de recherche n'ont pas été utilisés. Nous proposons une approche simple afin de manipuler les similarités entre deux textes contenant un nombre élevé de caractères, tout en ne suivant aucune règle de syntaxe ou de grammaire, contribuant ainsi à l'obtention d'une méthode fonctionnant indépendamment des langues. La procédure complète de notre méthode s'avère être la suivante :

1. Assemblage de deux bases de données contenant un nombre élevé d'information provenant de sources distinctes telles que des pages internet, des journaux, des blogues ou des fichiers informatiques de type pdf, doc et/ou text.
2. Analyse et modification de ces bases de données en éliminant les caractères spéciaux, la ponctuation et les mots vides.
3. Production de deux ensembles de caractères n-grammes avec les deux bases de données. Au courant de notre recherche, nous avons utilisé tous les types de n-grammes pour finalement déterminer que les trigrammes présentaient les meilleurs résultats.
4. Mesure de la distance entre deux ensembles de n-grammes de caractères en éliminant toutes les valeurs négatives des deux matrices de distance.
5. Calcul des scores de similarité et dissimilarité en utilisant les cinq mesures de cooccurrence : Jaccard, Dice, Overlap, Cosine et Simple.
6. Établissement d'un seuil fixé à $\alpha \geq 0.3$ étant donné les valeurs variables des cooccurrences.

La présente méthode fut implantée à l'aide des logiciels C# et Windows. De plus, la complexité de notre algorithme est de temps quadratique et dénotée par ($\boldsymbol{N}^2$), indiquant ainsi que la performance de notre algorithme est directement proportionnelle au carré de la grosseur de la base de données analysée. De façon globale, la précision de notre méthode a été évaluée a 86.67% & 94.74% pour les deux bases de données analysées. Les principaux résultats de notre recherche s'avèrent être les suivants:

- Capacité à analyser des phrases aléatoires sans prendre en compte les règles de syntaxe ou de grammaire.
- Analyse indépendante des dictionnaires en ligne, des pages internet telles que Wikipédia et des moteurs de recherche comme Google et Digg.com. Ceci contribue de façon générale à une augmentation de l'efficacité et de la versatilité de notre méthode tout en favorisant l'accroissement de la véracité au niveau des résultats obtenus.
- Implantation dans notre algorithme d'un mécanisme d'apprentissage indépendant.

- Implantation d'une fonction d'analyse multilingue permettant la détection de similarités et dissimilarités entre les phrases traduites, par exemple de l'anglais ↔ français.
- Développement d'une méthode versatile permettant la détection de similarités et dissimilarités entre toutes les langues répertoriées telles que l'anglais, le français, l'arabe et le mandarin.

En guise de conclusion, nous croyons fortement que les caractéristiques forgeant notre algorithme pourraient éventuellement incorporer des éléments d'intelligence artificielle afin de renforcer ses capacités d'analyse. Dans un futur proche, nous aimerions intégrer cette fonctionnalité afin de pouvoir analyser davantage de langues, ainsi que des textes plus volumineux.

ABSTRACT

Nowadays, detecting similar sentences can play a major role in various fundamental applications for reading and analyzing texts like information retrieval, categorization, detection of paraphrases, summarizing, translation etc. In this work, we present a novel method for the detection of similar sentences. This method highlights the using of units of n-grams of characters. In this method, the online dictionary as well as any search engine are not being used. Hence, this idea leads our method a simplest and optimum way to handle the similarities between two largest texts. Besides, the grammar rules as well as any syntax have not been used in our method. That is why, we expect that we can use this method for detecting similarities of any languages. We analyze and compare a range of similarity measures with our methodology. Meanwhile, the complexity of our method is $O(N^2)$ which is pretty much better.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Internet and social networks become an increasingly popular source of information, but often difficult to mine due to some limitations of current technologies. Text mining is equivalent to text analytics which is referred to as text data mining as well as to the process of deriving high-quality information from text where high-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning [39]. Text mining usually involves the process of -

- Structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database),
- deriving patterns within the structured data, and
- finally, evaluation and interpretation of the output.

'High quality' in text mining usually relates to some combination of relevance, novelty, and interestingness. Compared with the kind of data stored in databases where text is unstructured, amorphous and difficult to deal with algorithmically [1]. Nevertheless, in modern culture, text is the most common vehicle for the formal exchange of information and the field of text mining usually deals with texts whose function is the communication of information or opinions, and the motivation for trying to extract information from such text automatically is compelling—even if success is only partial.

Text analysis includes the application of techniques from areas such as information retrieval, natural language processing, information extraction and data mining [40]. These various stages of a text-mining process can be combined into a single workflow and the detail about of each of these areas and how, together, they form a text-mining pipeline is given below:

➤ **Information retrieval (IR) system –** identify the documents in an accumulation which match a user's query. The most well-known IR systems are search engines such as Google™ which distinguish those documents on the WWW that are relevant to a set of given words. IR systems are often used in libraries where the documents are typically not the books themselves but digital records containing information about the books as well as this is however changing with the advent of digital libraries, where the documents being retrieved are digital versions of books and journals.

IR systems allow a user to narrow down the set of documents that are relevant to a problem. As text mining involves applying very computationally intensive algorithms to large document collections whereas IR can speed up the analysis considerably by reducing the number of documents for analysis. For example, if we are interested in mining information only about similar word interactions, we might restrict our analysis to documents that contain the name of the verb and word 'to interact' or one of its synonyms.

- **Natural language processing (NLP)** – is one of the most former and most difficult problems in the field of artificial intelligence. It is the analysis of human language so that computers can understand natural languages as humans do. Although this goal is still some way off, NLP can perform some types of analysis with a high degree of success. For example:
    - Part-of-speech tagging classifies words into categories such as noun, verb or adjective
    - Word sense disambiguation identifies the meaning of a word, given its usage, from the multiple meanings that the word may have
    - Parsing performs a grammatical analysis of a sentence. Shallow parsers identify only the main grammatical elements in a sentence, such as noun phrases and verb phrases, whereas deep parsers generate a complete representation of the grammatical structure of a sentence

The role of NLP in text mining is to provide the systems in the information extraction phase with linguistic data that they need to perform their task. Often this is done by annotating documents with information such as sentence boundaries, part-of-speech tags and parsing results, filtering stop words, punctuations which can then be read by the information extraction (IE) tools.

- **Information extraction (IE)** – is the process of automatically obtaining structured data from an unstructured natural language document. Often this involves defining the general form of the information that we are interested in as one or more templates which are then used to guide the extraction process. IE systems rely heavily on the data generated by NLP systems. Tasks that IE systems can perform include:

- Term analysis, which identifies the terms in a document, where a term may consist of one or more words. This is especially useful for documents that contain many complex multi-word terms, such as scientific research papers
- Named-entity recognition, which identifies the names in a document, such as the names of people or organizations. Some systems are also able to recognize dates and expressions of time, quantities and associated units, percentages, and so on
- Fact extraction, which identifies and extracts complex facts from documents. Such facts could be relationships between entities or events

A very simplified example of the form of a template and how it might be filled from a sentence is shown in Figure 1. Here, the IE system must be able to identify that 'bind' is a kind of interaction, and that 'myosin' and 'actin' are the names of proteins. This kind of information might be stored in a dictionary or an ontology, which defines the terms in a field and their relationship to each other. But sometimes, using dictionary is become a big trouble for unsupervised data. The data generated during IE are normally stored in a database ready for analysis in the final stage, data mining.

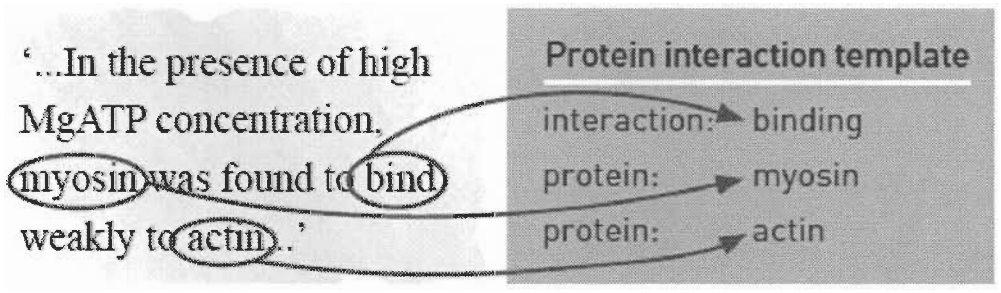

'...In the presence of high MgATP concentration, myosin was found to bind weakly to actin..'
Protein interaction template
interaction: binding
protein: myosin
protein: actin

Figure 1: Template-based Information Extraction

- **Data mining (DM)** (often also known as knowledge discovery) – is the process of identifying patterns in large sets of data. The aim is to uncover previously unknown, useful knowledge. When used in text mining, DM is applied to the facts generated by the information extraction phase. Continuing with our protein interaction example, we may have extracted many protein interactions from a document collection and stored these interactions as facts in a database. By applying DM to this database, we may be able to identify patterns in the facts. This may lead to new discoveries about the types of interactions that can or cannot occur, or the relationship between types of interactions and diseases and so on. The data generated by end-user queries via a suitable graphical interface can also be represented visually, for example, as a network of protein interactions.

Several approaches have been proposed to speed up information retrieval, reading and analysis. Among them, some are symbolic and logical, based on patterns, others are numerical and empirical, based on co-occurrences and some are a kind of pattern recognition based on algebraic formal models. But all those methods have some limitations. For example – in page based methodology, some author like to use search engine like Google, Digg and so on. But sometimes those search engines unable to give proper result. Besieds, some author like to use dictionary as well. That dictionary also has same problem to identify some words.

In this document, we would like to discuss about a novel approach using *n-grams of characters*. Our approach is not dependent on search engine as well as dictionary. Besides, we got the complexity of our algorithm is quadratic complexity $O(N^2)$.

An N-gram is an *N-character* slice of a longer string. Although in the literature the term can include the notion of any co-occurring set of characters in a string (e.g., an *n-gram of characters* made up of the first and third character of a word), in this paper we use the term for contiguous slices only. Typically, one slices the string into a set of overlapping *n-grams of characters*. In our system, we use *tri-grams of characters* for detecting similarities between two large texts. There have two benefits of *n-gram* models or algorithms which are simplicity and scalability. With larger *n*, a model can store more context with a well-understood space-time tradeoff, enabling small experiments to scale up efficiently.

At first, through our method we collect large texts from different sources i.e. pdf, text, docx, pptx, web-sites etc. Then it preprocesses those raw texts for getting good results. For preprocessing it filters all stop-words, punctuations and delimiters. Then this method splits all texts by using *tri-grams of characters*. After then two distance matrices is been created. Using five most popular co-occurrence methodologies, this method will give some values of similarities and dissimilarities. To better understand the similarities and dissimilarities we consider a scale *(0, 1)* where 0 and 1 will be considered as true similar and dissimilar values. These threshold values will provide a precise concept of measurement of similarities and dissimilarities.

Besides *n-grams of characters* models are widely used in various domain i.e. statistical natural language processing, speech recognition, computational biology, probability, data compression and so on. The efficiency as well as the universality usage of n-grams of characters lead us to deduce this innovative technique.

In this paper, we will cover the following topics:

- Chapter – 2 will discuss the related words with examples.
- Chapter – 3 will explain the methodology for similarity measure using several examples. Besides, it will also illustrate the multilingualism of our methodology.
- Chapter – 4 will display the implementation of our method which will illustrate the whole methodology. Beside it will discuss the complexity of our methodology.
- Chapter – 5 will describe the experimentation of our methodology. It will also show some comparison with others methodology.
- Chapter – 6 will provide a brief conclusion of our methodology and it will also indicate the future direction of work.

CHAPTER II

RELATED WORKS

Akermi and Faiz developed a method that extracts semantic similarity between sentences by combining semantic as well as syntactic information [2]. Their method is divided into three phases which are given below:

- **Calculating the semantic similarity:** In the first phase, using all the distinct words in the pairs of sentences and eliminating the functional words, the punctuations as well, they compute similarity scores via their Word Similarity Measure SimFA [3]. Then, by adding the all individual similarity scores as well as by normalizing they measure the semantic similarity between two sentences.
- **Calculating the syntactic similarity:** In the second phase, at first they compute the Jaccard coefficient by creating sets of the sentences including the functional words. Akermi and Faiz noticed that, measuring the word orders similarity scores which is based on the orders between word pairs is very useful to get significant information to distinguish the meaning of sentences. So that, they calculate the score of word order similarity among sentences. At last, they added the jaccard coefficient and the word order similarity in order to obtain the overall syntactic similarity measure.
- **Combine the semantic and the syntactic information:** In the final phase, the sentence similarity measure is evaluated by incorporating the semantic similarity measure as well as syntactic similarity measure.

Example: let us consider two simple text i.e.

text1 – 'Saima reads novels and newspapers.'

text2 – 'Saima reads and enjoys novels.'

- Calculating Semantic Similarity Score: to calculate the semantic similarity between these two sentences, firstly they eliminate the function words i.e. 'and' and punctuation. Then they prepare two sets of these filtered sentences like the following way:

Set_{s1} = {Saima, reads, novels, newspapers};

Set_{s2} = {Saima, reads, enjoys, novels};

Now by selecting word w_i from Set_{s1} and word w_j from Sets_2, they calculate the highest similarity which includes the computation of similarity scores all the pairs (w_i, w_j) using their word similarity measure Sim_{FA} presented in their previous work [3]. To measure Sim_{FA} of each word pair, they use online dictionary and the page count of a search engine named Digg.com. After calculating highest similarity of each pair in the two sets they add the similarity scores and then normalize by following way:

$$SemSim(S_1, S_2) = \frac{\sum StoredScores}{Minimum\ (ls_1, ls_2)}$$

Where, ls_1: the number of the terms of Set_{s1}

ls_2: the number of the terms of Set_{s2}.

Their procedure gives the semantic similarity score of these two sentences is: 0. Besides their dictionary is unable to understand the words like ‘Saima’, ‘reads’, ‘novels’, ‘enjoys’, ‘newspapers’. Note that their dictionary is unable to understands the name and the original words which conjugated with ‘s’.

- Calculating Syntactic Similarity: to calculate the syntactic similarity between the two sentences, firstly they form two sets out of the two sentences including the function words like the following way:

Set_{s1} = {‘Saima’, ‘reads’, ‘novels’, ‘and’, ‘newspapers’}

Set_{s2} = {‘Saima’, ‘reads’, ‘and’, ‘enjoys’, ‘novels’}

Then the calculate the intersection of the words in the two sentences compared to the size of the union of the words in the two sentencs using the Jaccard coefficient:

$$jaccard(S_1, S_2) = \frac{m_c}{ls1 + ls2 - m_c}$$

Where, m_c : the number of common words between the two sets.

ls1 : the number of words in the sets Set_{s1}.

ls2 : the number of words in the sets Set_{s2}.

As par the two given sets, the Jaccard coefficient is: 0.6666667. In addition, by calculating the words order similarity measure between given two sentences, they calculate a word order similarity score in the following way:

$\text{Word}_{\text{order}}(S_1)$ = {(Saima, reads); (Saima, novels); (Saima, and); (Saima, newspapers); (reads, novels); (reads, and); (reads, newspapers); (novels, and); (novels, newspapers); (and, newspapers)}

$\text{Word}_{\text{order}}(S_2)$ = {(Saima, reads); (Saima, and); (Saima, enjoys); (Saima, novels); (reads, and); (reads, enjoys); (reads, novels); (and, enjoys); (and, novels); (enjoys, novels)}

$$Sim_{wo}(S_1, S_2) = \frac{|Word_{order}(S_1) \cap Word_{order}(S_2)|}{|Word_{order}(S_1) \cup Word_{order}(S_2)|}$$

Now the similarity score between these two word order sets is: 0.368421. By adding the jaccard coefficient and the word order similarity they calculate the syntactic similarity measure like the following way:

$$SynSim(S_1, S_2) = Jaccard(S_1, S_2) + Sim_{wo}(S_1, S_2)$$

So, the syntactic similarity between these two given sentences is: 1.035088.

- In the final stage, the overall sentence similarity measure will be calculated by using the following formula:

$$SenSim_{FA}(S_1, S_2) = \propto \times SemSim(S_1, S_2) + (1 - \propto) \times SynSim(S_1, S_2)$$

Where, $\propto \in [0, 1]$. So we get the similarity score between these two sentencs is: 0.

Kumari and K developed a method to identify the numerous semantic relations that exist between two given words, they use a pattern extraction and clustering method [4]. The optimal combination of page counts-based co-occurrence measures and lexical pattern clusters is learned using support vector machine (SVM) used to find semantic similarity between two words. For this, they first downloads few web pages from Google and stores it in the database before their proposed system methods are applied to it. Then, they use web search engine and retrieves page counts for the two words and also for their conjunctive (i.e a word P, another one Q as wel as $P \cap Q$ (or P and Q)). They use four popular similarity scores for calculating page counts-based similarity scores which consider the global co-occurrences of two words on the web. Here, they notice that two words may appear on some pages accidentally for the scale and noise in web data. In order to reduce the contrary effect due to random co-occurrences, the four co-occurrences are set to zero if the page counts for the query P∩Q are less than a threshold c (c is assumed to be 5). Therefore, page counts based similarity measure are liable to noise and are not trustworthy when H(P∩Q) is low. Then, they snippets off the local context of query words which contains a window of text selected from a document that includes the queried words. In this way the database also saves the snippets for the conjunctive query. Thus lexical patterns are extracted which are clustered together and also given into the Support Vector Machine (SVM). Finally the SVM acts up on both results of word co-occurrence measures and also pattern clusters in order to calculate semantic similarity between two given words. Generally, SVM is used to combine both page count-based co-occurrence measures, and snippets-based lexical pattern clusters to construct an accurate semantic similarity measure.

Example: let us consider two words like – P = Food and Q = Fruit. Now, the following figure will illustrate the whole procedure of their methodology

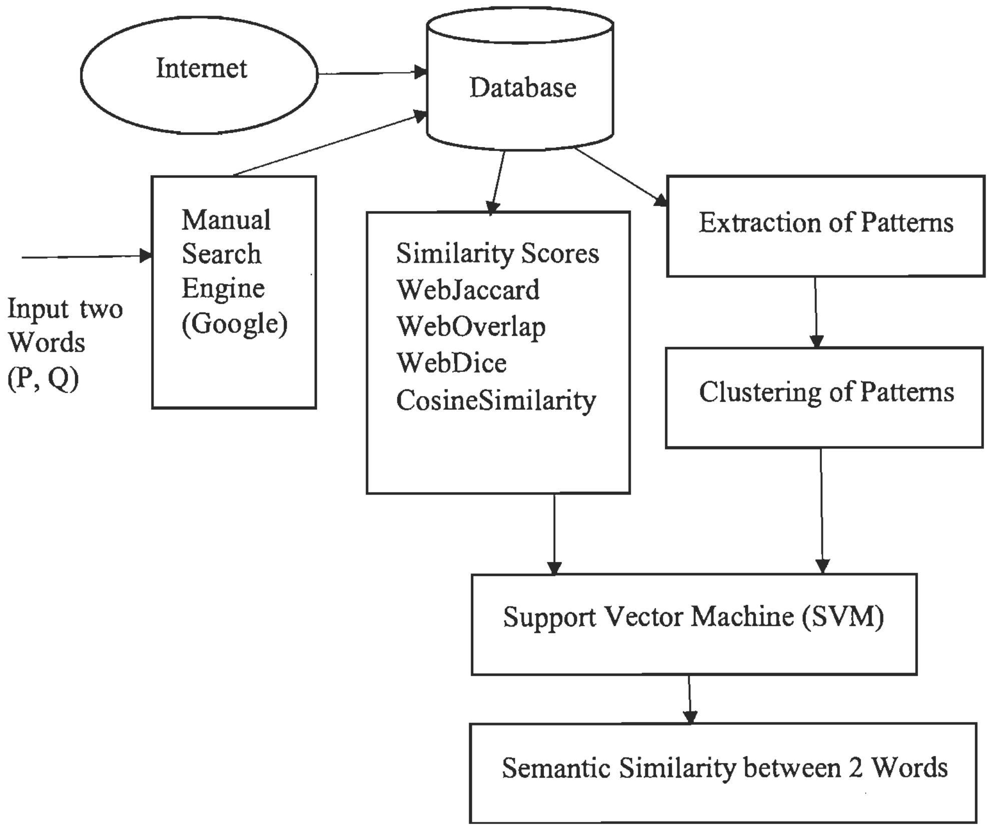


Figure 2: Illustrates the Proposed Method of Kumari & K

According to their proposed method, the similarity between given two words are given below:

Table 1: Semantic similarity between two words

Word 1 (P)	**Word 2 (Q)**	**WebJaccard**	**WebOverlap**	**WebDice**	**CosineSimilarity**	**Semantic Similarity**
Food	Fruit	0.08	0.17	0.14	0.14	0.09

Where, 0.0 is considered as totally dissimilar and 1.0 is considered totally similar value.

Takale and Nandgaonkar presented an approach for measuring semantic similarity between words using the Snippets returned by Wikipedia and the five different similarity measures of association [5]. Firstly, they accumulate snippets using simple vocabulary to explain the word, or giving simple definition or some description about the word. They said that, these snippets are very much suitable to measure semantic similarity between words. After downloaded the snippets from Wikipedia, they has preprocessed that snippets which are given below:

- **Stop Word Removal:** For removing stop words, they use Luhn's [6] method who basically used Zipf's law as a null hypothesis to specify two cut-offs - an upper and a lower.
- **Suffix Stripping and Stemming:** In this stage, they use an algorithm proposed by Porter [7] for suffix stripping. Finally they extracted some keywords from that snippet document (Wikipedia) for each words.
- **Similarity Measure:** Using those extracted keywords, they find out the semantic similarity results by applying five different strategies - Jaccard, Dice, Overlap, Cosine and simple matching.

Example: let us consider two words like –

Word1 = Food

Word2 = Fruit

According to their method, at first they collect some snippet from Wikipedia. The flow chart of their method to measure similarity between two words:

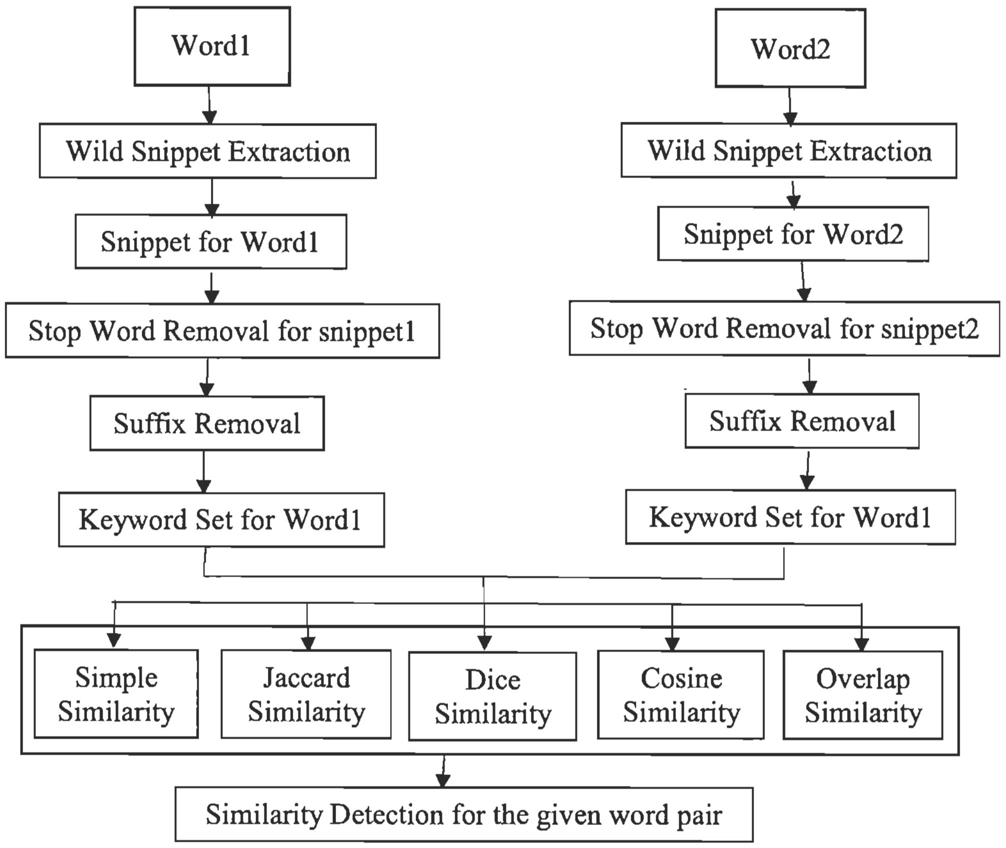


Figure 3: Flow of Similarity Computation between two words

According to the flow, the scores for similarity computation between the given words are:

Table 2: Scores for similarity computation between two given words

Word1	**Word2**	**Simple Similarity**	**Jaccard Similarity**	**Dice Similarity**	**Cosine Similarity**	**Overlap Similarity**
Food	Fruit	3	0.04166667	0.08	0.0857493	0125

Where, 0 is considered as totally dissimilar and 1 is considered totally similar value.

Bollegala, Matsuo and Ishizuka proposed to measure semantic similarity between two words by extracting page counts and snippets using the AND query of the two words from a Web search engine (Google) [8]. They also define numerous similarity scores i.e. Jaccard, Overlap (Simpson), Dice, and PMI (point-wise mutual information) based on page counts and lexico-syntactic patterns and all results are later integrated using support vector machine to form a robust semantic similarity measure. They noticed that page count based similarity measure do not consider the relative distance between words that co-occur in a page while the two words co-occurrence in a page might not be related. Therefore, similarity score defined purely on page counts are prone to noise and are not reliable when the page counts are low. Besides, the snippets capture the local context of query words. To overcome these drawbacks, they propose lexico-syntactic pattern extraction algorithm which automatically extracted from snippets. Now, they run this algorithm with a set of non-synonymous word-pairs and count the frequency of the extracted patterns. Then they use a test of statistical significance to evaluate the probable applicability of a pattern as an indicator of synonymy. Their fundamental idea of this analysis is that, if a pattern appears a statistically significant number of times in snippets for synonymous words than in snippets for non-synonymous words, then it is a reliable indicator of synonymy. Here, to create a set of non-synonymous word-pairs, they select two nouns from WordNet (http://wordnet.princeton.edu/) arbitrarily. If the selected two nouns do not appear in any WordNet synset then they select them as a non-synonymous word-pair. Thus using the pattern frequencies (which are generated by using synonymous word-pairs (positive training instances) and also non-synonymous word-pairs (negative training instances)) with similarity scores, they form a feature vector [9]. Lastly, they train a two class support vector machine (SVM) with the labelled training instances and thus the semantic similarity between two given words is defined as a posterior probability that they belong to positive (synonymous word-pairs) class. Their consideration is, being a large-margin classifier, output of an SVM is the distance from the decision hyper-plane. Here they also use sigmoid functions to convert the un-calibrated distance into a calibrated posterior probability [10].

Example: let us consider, the two words i.e. –

P = food and Q = fruit.

According to their method at first they calculate the page count based similarity scores for the query *P AND Q*. They modify four popular co-occurrence measure – Jaccard, Overlap (Simpson), Dice, PMI (point-wise mutual information) and their modification with the results are given below:

- WebJaccard coefficient between word *P* and *Q* is defined by,

$$WebJaccard(P,Q) = \begin{cases} 0 & if\ H(P \cap Q) \leq c \\ \dfrac{H(P \cap Q)}{H(P) + H(Q) - H(P \cap Q)} & otherwise \end{cases}$$

Where, $P \cap Q$: the conjugation query *P AND Q*.

H(P) : the page count for the query *P* in a search engine (Google)

$H(P \cap Q)$: the page count for the query *P AND Q* in a search engine (Google)

H(Q) : the page count for the query *Q* in a search engine (Google)

c : is a threshold in order to reduce the adverse effects attributable to random co-occurrences. ($c = 5$ in this experiment)

- WebOverlap coefficient is defined as:

$$WebOverlap(P,Q) = \begin{cases} 0 & if\ H(P \cap Q) \leq c \\ \dfrac{H(P \cap Q)}{\min(H(P), H(Q))} & otherwise \end{cases}$$

- WebDice is as a variant of Dice coefficient by:

$$WebDice(P,Q) = \begin{cases} 0 & if\ H(P \cap Q) \leq c \\ \dfrac{2 \times H(P \cap Q)}{H(P) + H(Q)} & otherwise \end{cases}$$

- WebPMI is as a variant from PMI using page counts by:

$$WebPMI(P,Q) = \begin{cases} 0 & if\ H(P \cap Q) \leq c \\ \log_2\left(\dfrac{\frac{H(P \cap Q)}{N}}{\frac{H(P)}{N} \times \frac{H(Q)}{N}}\right) & otherwise \end{cases}$$

Where, *N* : the number of documents indexed by the search engine (they set $N = 10^{10}$ according to the number of indexed pages reported by Google).

After that they extract the lexico-syntactic patterns for the query *'P AND Q'* and for each word pair *P*, *Q* by using a certain algorithm. To leveage the pattern extraction process, they randomly select some certain pairs of synonymous nouns from WordNet synsets. Besides, if the selected noun i.e. *P* and *Q* do not appear in any WordNet synset then the select them as a non-synonymous word-pair. By counting the frequency of the extracted patterns and then using a test of statistical significance to evaluate the probable applicability of a pattern as an indicator of synonymy. Here the fundamental idea of this analysisis that, if a pattern appears a statistically significant number of items in snippets for synonymous words than in snippets for non-synonymous words, then it is a reliable idicator of synonymy. Finally, using SVM (suppot vector machine), they integrate the results of patterns and page counts to find the scores of similarity measure between the given words. So, according to their method, their final results for P and Q are given below:

Table 3: Semantic similarity between P and Q

Word-Pair	**Web Jaccard**	**Web Dice**	**Web Overlap**	**Web PMI**	**Final Score**
Food-fruit	0.753	0.765	1	0.448	0.998

Where, 0 is considered as totally dissimilar and 1 is considered totally similar value.

According to Islam, Milions and Keselj's method, we get to know about the unsupervised approach for measuring text similarity using the tri-gram word similarity [11]. Their main idea is to find for each word in the shorter text, some most similar matching at the word level, in the longer text. Their procedure to get overall text similarity is given below:

1. In the first step they preprocesse the two input texts by removing special characters, puncutations and stop words where two input texts such as, $P = \{p_1, p_2, \ldots, p_m\}$ and $R = \{r_1, r_2, \ldots, r_n\}$ have m and n tokens, respectively and $n \geq m$. Otherwise, they switch P and R.

2. Then they count the number of p_i's (say, δ) for which $p_i = r_j$, for all $p \in P$ and for all $r \in R$ i.e. there are δ tokens in P that exacly match with R, where $\delta \leq m$. Then they remove all δ tokens from both of P and R to get $P = \{p_1, p_2, \ldots, p_{m-\delta}\}$ and $R = \{r_1, r_2, \ldots, r_{n-\delta}\}$ where if all the terms match then $m - \delta = 0$ and then they directly go to the final taks means step 5.

3. Then they construct a $(m - \delta) X (n - \delta)$ semantic similarity matrix which is known as $M = (\alpha_{ij})_{(m-\delta)X(n-\delta)}$. To create matrix they are following some rules, which are given below:

   a. first they calculate semantic relatedness (α_{ij}) of two words getting from two different input text $P = \{p_1, p_2, \ldots, p_{m-\delta}\}$ and $R = \{r_1, r_2, \ldots, r_{n-\delta}\}$ to consider the frequencies of all the tri-grams that start and end with the given pair of words with respect to the uni-gram frequencies of the pair. So the tri-gram word similarity between p_i and r_j, $Sim(p_i, r_j) \in [0, 1]$ defined as following formula:

$$Sim(p_i, r_j) = \begin{cases} \dfrac{\log \frac{\mu(p_i, n_1, r_j, n_2)C^2}{c(p_i)c(r_j)\min(c(p_i), c(r_j))}}{-2 \times \log \frac{\min(c(p_i), c(r_j))}{C}} & if \ \dfrac{\mu(p_i, n_1, r_j, n_2)C^2}{c(p_i)c(r_j)\min(c(p_i), c(r_j))} > 1 \\ \dfrac{\log 1.01}{-2 \times \log \frac{\min(c(p_i), c(r_j))}{C}} & if \ \dfrac{\mu(p_i, n_1, r_j, n_2)C^2}{c(p_i)c(r_j)\min(c(p_i), c(r_j))} \leq 1 \\ 0 & if \ \mu(p_i, n_1, r_j, n_2) = 0 \end{cases}$$

Where, C = the maximum frequency possible among all Google uni-grams,
$c(p) / c(r)$ = the frequency of word p / r in Google uni-grams,
$c(p_i, p_j, p_k)$ = the frequency of the tri-gram p_i, p_j, p_k in Google tri-grams.
$min(x, y)$ = the function that returns the minimum number between x and y.
here, the following function represents the mean frequency of n_1 tri-grams that start with word p_i and end with word r_j and n_2 tri-grams that start with word r_j and end with word p_i.

$$\mu(p_i, n_1, r_j, n_2) = \frac{1}{2}\left(\sum_{k=1}^{n_1} c(p_i\ (p_k\ or\ r_k)\ r_j) + \sum_{k=1}^{n_2} c(r_j\ (r_k\ or\ p_k) p_i)\right)$$

b. Then they put a_{ij} in row i and column j position of the matrix for all $i = 1 \ldots m - \delta$ and $j = 1 \ldots n - \delta$. So the matrix is given below:

$$M = \begin{bmatrix} \alpha_{11} & \cdots\ \alpha_{1j} & \cdots & \alpha_{1(n-\delta)} \\ \vdots & \vdots\ \alpha_{ij} & \ddots & \vdots \\ \alpha_{(m-\delta)1} & \cdots\ \alpha_{(m-\delta)j} & \cdots & \alpha_{(m-\delta)(n-\delta)} \end{bmatrix}$$

4. In this step, they set notation for two known functions mean (μ) and statndard deviation (σ) considering a set of x numbers, $\{a_1, \ldots, a_x\}$ as:

$$\mu(\{a_1, \ldots, a_x\}) = \frac{1}{x}\sum_{i=1}^{x} a_i$$

$$\sigma(\{a_1, \ldots, a_x\}) = \sqrt{\frac{1}{x}\sum_{i=1}^{x} (a_i - \mu(\{a_1, \ldots, a_x\}))^2}$$

Here, for each row in M, they find the set of elements for any row (i) such that each element in the set is larger than the summation of the mean and standard deviation of that row. Thus they are able to find out some most similar matchings which consider only a single matching per word. If there are y_i such elements in the set then they write that set, A_i, in the set-builder notation as:

$$A_i = \{\alpha_{ij} : \alpha_{ij} \in \{\alpha_{i1}, \dots, \alpha_{ij}, \dots, \alpha_{i(n-\delta)}\}, \alpha_{ij} > \mu(\{\alpha_{i1}, \dots, \alpha_{ij}, \dots, \alpha_{i(n-\delta)}\}) + \sigma(\{\alpha_{i1}, \dots, \alpha_{ij}, \dots, \alpha_{i(n-\delta)}\})\}$$

The mean of these y_i elements is $\mu(A_i)$. The summation of the means of all the m-δ rows in M is $\sum_{i=1}^{m-\delta} \mu(A_i)$.

5. They add $\sum_{i=1}^{m-\delta} \mu(A_i)$ and scale this total score by the reciprocal harmonic mean of m and n to obtain a normalized similarity score between 0 and 1, inclusively:

$$S(P, R) = \frac{(\delta + \sum_{i=1}^{m-\delta} \mu(A_i) \times (m+n))}{2mn}$$

Kondrak develop a notion of n-gram similarity and distance where they illustrate that the edit distance and the length of longest common subsequent are special cases of n-gram distance and similarity respectively [12]. Here, they define a class of word similarity measures which include two widely-used measures – the longest common subsequence ratio (LCSR) and the normalized edit distance (NED) and a series of new measures based on n-grams, $n > 1$. At first, they consider two measure-related issues:

- Normalization: it is a method of discounting the length of words that are being compared. As we know that the length of the longest common subsequence (LCS) of two randomly-generated strings grows with the length of the strings, so to avoid the length bias problem, a normalized variant of the LCS is usually preferred here. the longest common subsequence ratio (LCSR) is computed by dividing the length of the longest common subsequence by the length of the longer string and this procedure is quite like normalize edit distance.

- Affixing: it is a way of increasing sensitivity to the symbols at string boundaries. Without it the boundary symbols participate in fewer n-grams than the internal symbols. For example: the word *abc* contains two bigrams: *ab* and *bc* where the initial symbol *a* occurs in only one bigram and the internal symbol *b* occurs in two bigrams. Per their observation this is a highly undesirable effect for measuring word similarity, because the initial symbols play crucial role in human perception of words. So, to avoid negative bias, they sometimes add extra symbols to the beginning and/or ending of words. Here their affixing method is aimed at emphasizing the initial segments which tend to be much more important than final segments in determining word similarity.

Their algorithm of n-gram similarity and distance measure incorporate both normalization and affixing. In the algorithm, a unique special symbol is defined for each letter of the original alphabet and each word is augmented with a prefix composed of *n-1* copies of the special symbol that corresponds to the initial letter of the word. For example: if n=3, '*amikin*' is transformed into '*ââamikin*' if the original words have length *K* and *L* respectively and the number of n-grams is thus increased from *K+L-2(n-1)* to *K+L*. The normalization is achieved by simply dividing the total similarity score by *max(K, L)*, the original length of the longer word. Per Kondrak idea, he assures that the new measure will return 1 if and only if the words are identical and 0 if and only if the words have no letters in common. Finally, their evaluation procedure is as follows:

1. Establish a gold standard set *G* of word pairs that are known to be related.
2. Generate a much larger set *C* of candidate word pairs, $C \supset G$.
3. Compute the similarity of all pairs in *C* using similarity measure.
4. Sort the pairs in *C* per the similarity value, breaking ties randomly.
5. Compute the 11-point interpolated average precision on the sorted list which is an information-retrieval evaluation technique computed for the recall levels of 0%, 10%, 20%, ..., 100% and then average to yield a single number. They set the precision value at 0% recall to 1, and the precision value at 100% recall to 0.

In this paper, they showed that the bigram methods are overall somewhat more effective than the trigram methods. The differences between the positional and the comprehensive n-gram variants, where they exist are insignificant but the binary variant is sometimes much worse.

Example: in their methodology, the use genetic cognates which are words of the same origin that belong to distinct languages. Cognates are usually like their phonetic form which makes string similarity in an important clue for their identification. For example, English 'father', German 'vater', and Norwegian 'far' constitute a set of cognates, since they all derive from a single Proto-Germanic word (reconstructed as *faδēr). At first, they extract all nouns from two machine-readable word list that had been used to produce an Algonquian etymological dictionary which produce two sets named Cree nouns (contain 1628) and Ojibwa nouns (contain 1023) respectively. The set *C* of candidate pairs was created by generating all possible Cree-Ojibwa pairs (a Cartesian product). Their whole task is to identify 409 cognate pairs among huge candidate word pairs (approx. 0.025%). They result is given below:

Table 4: The average interpolated precision for various measures on word-similarity tasks

	DICE	XXDICE	LCS	LCSR	BI-SIM			TRI-SIM		
					Bin	Pos	Com	Bin	Pos	Com
Genetic Cognates	0.394	0.519	0.141	0.564	0.526	0.597	0.595	0.466	0.593	0.589

Table 5: The average interpolated precision for various measures on word-distance tasks

	PREFIX	EDIT	NED_1	NED_2	BI-DIST			TRI-DIST		
					Bin	Pos	Com	Bin	Pos	Com
Genetic Cognates	0.276	0.513	0.592	0.592	0.545	0.602	0.602	0.468	0.589	0.589

CHAPTER III

METHODOLOGY

In many applications, it is necessary to algorithmically quantify the similarity exhibited by two strings composed of symbols from a finite alphabet. Numerous string similarity measures have been proposed. We formulate a concept of n-gram similarity measure and distance measure. Our formulation of text similarity measures based on n-grams as well as report the results of experiments suggest that the new evaluations outperform than others.

3.1 N-grams

An *n-gram* is a sequence of symbols extracted from a long string [13]. The symbol can be a byte, character or word [14]. In simplistic terms this means an *n-gram* is a collection of tokens where "*n*" equals the number of tokens contained within the collection. A token within this context can basically be any portion of data divided into smaller logical pieces by the n-gram creator. One of the simplest examples to consider is a text document. A token within a text document might represent each individual word within the document as delimited by spaces with all punctuation characters removed. However, many alternative tokenization strategies could be devised to produce tokens from a given text document.

A *word n-grams* is a sequence of n successive words. Considering the text document example, a *3-gram* could represent an *n-gram* containing 3 word tokens and a *4-gram* would contain 4 word tokens. Once the n-gram parameter "*n*" and a tokenization method has been decided upon, each *n-gram* can be produced by starting with the first "*n*" tokens and creating the first *n-gram*. From that point on, additional *n-grams* are produced by removing the first token contained within the current *n-gram* and concatenating one additional token to the end of the current *n-gram*. This process is continued until the last "*n*" tokens within the provided document are reached, and the last *n-gram* is then created.

Besides, An *N-Grams of characters* is a sequence of *n* successive characters [15]. Extracting *character n-grams* from a document is like moving a *n* character wide 'window' across the document character by character [14]. Each window position covers *n* characters, defining a single *N-gram*. It is a *bi-grams* for *n= 2*, *tri-grams* for *n=3*, a *quadric-grams* for *n=4*, etc [16]. For example, cutting into tri-grams of the word 'computer' gives 'com', 'omp', 'mpu', 'put', 'ute', 'ter'.

Nowadays, *n-grams of characters* are used for many types of applications including computational linguistics, DNA sequencing, protein sequencing, and data compression just to name a few [17] [18] [19] [20]. *N-grams of characters* have previously been used for subjectivity recognition in text and speech [21], classification of images [16], language-independent categorization of text [22] and so on. On the other hand, Wilson and R. have shown that there is value in using very shallow linguistic representations, such as character n-grams, for recognizing subjective utterances, in particular, gains in the recall of subjective utterances [23]. Besides, in the paper of Kanaris, I., et al., they presented a comparison of words and character n-grams in the framework of content-based anti-spam filtering [24].

The most important property of the *character n-gram* approach is that it avoids the use of tokenizers, lemmatizers, and other language-dependent tools. In addition to *character n-grams* of fixed length, or might be variable-length depends on using which originally used for extracting multi-word terms for information retrieval applications [24]. Results of cost-sensitive evaluation indicate that the variable-length *n-gram* model is more effective in any of the three examined cost scenarios (i.e. low, medium or high cost). Although the majority of the variable-length *n-grams* consists of *3-grams*, there are only a few common members with the fixed-length *3-gram* set. Hence, the information included in the variable length *n-grams* is quite different in comparison to the information represented by case sensitive *3-grams*.

In this paper, we develop a notion of the similarity and distance between *n-gram of characters*. We build our own methodology for measuring distance between *n-grams of characters* which is a special case of *n-gram* distance and similarity respectively. Besides, for measuring similarity, we use five popular co-occurrence text similarity measure. On the other hand, we also have a scale of *(0, 1)* to understand the measure of similarity and dissimilarity more precisely. Here, the scale *(0, 1)* will symbolize the intense similarity or dissimilarity between two files.

3.2 Similarity Measure

In order to compare words for similarity, the first operation is to observe co-occurrence frequencies n_{AB} between any word A and B [25]. These are then interpreted by a co-occurrence significance measure, given the individual word frequencies n_A, n_B and the corpus size n. Out of the many possible measures those were chosen because of their frequently used in various papers along with this paper as well as those are statistically well-founded. String similarity measures operate on string sequences and character composition [26]. A string metric is a metric that measures similarity or dissimilarity (distance) between two text strings for approximate string matching or comparison. In this paper for measuring text similarity as well as dissimilarity, we will demonstrate some very popular methods which we use in our methodology.

- **Jaccard Similarity Coefficient** – also known as Jaccard Index, is a statistic used for comparing the similarity and diversity of sample sets [5]. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A|+|B|-|A \cap B|} \qquad \text{(i)}$$

For example – let say A and B are string sets. To calculate the similarity between: 'computing' and 'recomputing' and now the *tri-grams of characters* sets will be:

A = *{com, omp, mpu, put, uti, tin, ing}* and

B = *{rec, eco, com, omp, mpu, put, uti, tin, ing}*.

Now, set A has seven and set B has nine elements and the intersection of these two sets has seven elements: *com, omp, mpu, put, uti, tin, ing*. Now according to the formula, the calculation would be:

$$J(A,B) = \frac{7}{7+9-7} = 0.778$$

The Jaccard similarity between these two strings is – 0.778 i.e. 77.8% similar to each other.

- **Jaccard Dissimilarity Coefficient** – also known as Jaccard Distance, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union [27] [28] [29]:

$$d_j(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \qquad \textbf{(ii)}$$

For example – to calculate the dissimilarity between: 'computing' and 'recomputing' and now the *tri-grams of characters* sets will be:

A = {com, omp, mpu, put, uti, tin, ing} and

B = {rec, eco, com, omp, mpu, put, uti, tin, ing}

Now, set A and B has seven and nine elements respectively and the intersection of these two sets has seven elements: *com, omp, mpu, put, uti, tin, ing*. Besides, the union of these two sets has nine elements: *com, omp, mpu, put, uti, tin, ing, rec, eco*. Now according to the formula, the jaccarad dissimilarity coefficient is:

$$d_j(A,B) = \frac{9-7}{9} = 0.222$$

The Jaccard dissimilarity coefficient between two strings is – 0.222 i.e. 22.2% dissimilar to each other.

- **Dice Similarity Coefficient** – was independently developed by the botanists Thorvald Sørensen [30] and Lee Raymond Dice [31] who published in different times respectively [5]. It is based on Dice's coefficient, named after Lee Raymond Dice and known as Dice coefficient [5]. Sørensen's original formula was intended to be applied to presence/absence data, and is [30]:

$$QS = \frac{2\,|X \cap Y|}{|X| + |Y|} \qquad \textbf{(iii)}$$

Where, $|X|$ and $|Y|$ are the number of species in the two samples. Based on what is written here, theoretically the coefficient is:

$$\textit{Dice Coefficient} = \frac{2v\,true\ positives\ (TP)}{(2\ true\ positives\ (TP) + false\ positives\ (FP) + false\ negatives\ (FN))},$$

As compared with the Jaccard index, which omits true negatives from both the numerator and the denominator. QS is the quotient of similarity and ranges between 0 and 1. It can be viewed as a similarity measure over sets. Similarly, to the Jaccard index, the set operations can be expressed in terms of vector operations over binary vectors A and B:

$$s_v = \frac{2\,|A\,.\,B|}{|A|^2 + |B|^2}.$$

Which gives the same outcome over binary vectors and gives a more general similarity metric over vectors in general terms. For sets X and Y of keywords used in information retrieval, the coefficient may be defined as twice the shared information (intersection) over the sum of cardinalities when taken as a string similarity measure, the coefficient may be calculated for two strings x and y using bigrams as follows:

$$s = \frac{2\, n_t}{n_x + n_y}$$

Where, n_t = The number of character bigrams found in both strings,

n_x = The number of bigrams in string x and

n_y = The number of bigrams in string y.

For example: to calculate the similarity between: "night" and "nachat", the set of *bi-grams of characters* in each word is: $\{ni, ig, gh, ht\}$ *and* $\{na, ac, ch, ht\}$. Each set has four elements and the intersection of these two sets has only one element: *ht*. Inserting these numbers into the formula, the calculation would be: $s = \frac{(2 \times 1)}{(4+4)} = 0.25$ i.e. the similarity is 25%.

- **Dice Dissimilarity Coefficient** – this coefficient is not very different in form from the Jaccard index [33]. However, since it doesn't satisfy the triangle inequality, it can be considered a semimetric version of the Jaccard index. The function ranges between zero and one, like Jaccard. Unlike Jaccard, the corresponding difference function is not a proper distance metric as it does not possess the property of triangle inequality. The difference function is given below:

$$d = 1 - \frac{2\,|X \cap Y|}{|X| + |Y|} \qquad \textbf{(iv)}$$

But, it is not a proper distance metric as it does not process the property of triangle inequality [32]. The simplest counterexample of this is given by the three sets *{a}*, *{b}*, and *{a, b}*, the distance between the first two being 1, and the difference between the third and each of the others being one-third. To satisfy the triangle inequality, the sum of any two of these three sides must be greater than or equal to the remaining side. However, the distance between *{a}* and *{a, b}* plus the distance between *{b}* and *{a, b}* equals $\frac{2}{3}$ and is therefore less than the distance between *{a}* and *{b}* which is 1.

For example: to calculate the distance between: 'night' and 'nachat', the set of *bi-grams of characters* in each word is: *{ni, ig, gh, ht}* and *{na, ac, ch, ht}*. Each set has four elements and the intersection of these two sets has only one element: *ht*. Inserting these numbers into the formula, the calculation would be:

$$d = 1 - \frac{2 \times 1}{4 + 4} = 0.75$$

So, the dissimilarity coefficient is: 0.75 i.e. these two words are 75% dissimilar to each other.

- **Overlap Coefficient** – it is a similarity measure related to the Jaccard index that measures the overlap between two sets, and is defined as the size of the intersection divided by the smaller of the size of the two sets which is given below [5]:

$$\boldsymbol{overlap(X,Y)} = \frac{|\boldsymbol{X} \cap \boldsymbol{Y}|}{\mathbf{min}(|\boldsymbol{X}|,|\boldsymbol{Y}|)} \qquad \textbf{(v)}$$

If set *X* is a subset of *Y* or the converse, then the overlap coefficient is equal to one. For example: to measure overlap similarity between: 'computing' and 'recomputing', the set of *tri-grams of characters* in each word is –

A = {com, omp, mpu, put, uti, tin, ing} and

B = {rec, eco, com, omp, mpu, put, uti, tin, ing}

Each set has seven and nine elements respectively and the intersection of these two sets has seven elements: *com, omp, mpu, put, uti, tin, ing*. Now inserting these values into the formula and we will get the following result:

$$overlap(A,B) = \frac{7}{\min(7,9)} = 1$$

So, the similarity coefficient is: 1, i.e. these two strings are 100% like each other.

- **Cosine Similarity** – is a measure of similarity between two vectors of *n* dimensions by finding the angle between them [5] [33]. It is often used to compare documents in text mining.

$$\boldsymbol{C(A,B)} = \frac{|\boldsymbol{A} \cap \boldsymbol{B}|}{\boldsymbol{sqrt}(|\boldsymbol{A}|) \times \boldsymbol{sqrt}(|\boldsymbol{B}|)} \qquad \textbf{(vi)}$$

Where, A_i and B_i are components of vector *A* and *B* respectively. The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning the same, with 0 indicating orthogonality (decorrelation), and in-between values indicating intermediate similarity or dissimilarity. For *n-gram* matching, the attribute vectors *A* and *B* are usually the term frequency vectors of the matrices. The cosine similarity can be seen as a method of normalizing matrix length during comparison. The term "cosine similarity" is sometimes used to refer to different definition of similarity provided below.

For example – let us measure similarity between: 'computing' and 'recomputing' and the *tri-grams of characters* sets of these two strings are:

$A = \{com, omp, mpu, put, uti, tin, ing\}$ and

$B = \{rec, eco, com, omp, mpu, put, uti, tin, ing\}$.

So, each set contains seven and nine elements respectively. The intersection of these sets contains seven elements – *com, omp, mpu, put, uti, tin, ing*. Now inserting these values into the formula and we will get the cosine similarity, given below –

$$C(A,B) = \frac{7}{sqrt(7) \times sqrt(9)} = 0.882$$

So, the similarity coefficient is – 0.882 i.e. these two strings are 88.2% like each other.

➢ **Simple Matching Coefficient (SMC)** – The simplest similarity coefficient is called the Simple Matching Coefficient, which is defined by the following formula [34]:

$$SMC = \frac{\textit{number of mathing attributes}}{\textit{number of attributes}} = \frac{f_{11}+f_{00}}{f_{10}+f_{01}+f_{11}+f_{00}} \qquad \textbf{(vii)}$$

Where, f_{11} = is the total number of attributes where A and B both have a value of 1.

f_{01} = is the total number of attributes where the attribute of A is 0 and the attribute of B is 1.

f_{10} = is the total number of attributes where the attribute of A is 1 and the attribute of B is 0.

f_{00} = is the total number of attributes where the attribute of A and B both have a value of 0.

Table 6: Two sets for SMC

		A	
		0	**1**
B	**0**	F_{00}	F_{10}
	1	F_{01}	F_{11}

In many cases, the SMC is inadequate because it treats 0 and 1 as equally important. Often, we are only concerned about when both objects have a 1 for a particular attribute.

For example: let us measure similarity between: 'computing' and recomputing' and the *tri-grams of characters* sets of these two strings are:

A = {com, omp, mpu, put, uti, tin, ing} and

B = {rec, eco, com, omp, mpu, put, uti, tin, ing}.

So, the intersection of these sets contains seven elements – *com, omp, mpu, put, uti, tin, ing*. On the other hand, the total number of attributes (union) of these sets contains nine attributes – *com, omp, mpu, put, uti, tin, ing, rec, eco*. Now, inserting these values into the formula to get the SMC value in the following way –

$$S(A, B) = \frac{7}{9} = 0.778$$

So, the similarity matching between these two strings is: 0.778 i.e. both are 77.8% similar.

3.3 Procedure of Our Methodology

Our method uses the notion of *n-grams* of characters. This notion has been used in the method of [22], [35], [36], [37] etc. articles in a respective manner. As we know, an *n-gram* is a contiguous sequence of *n* items from a given sequence of text or speech and the items might be letters, characters, words per the application. Because of simplicity as well as scalability, we choose *n-grams* of characters in our technique. Additionally, *n-grams* of characters are natural due to the nature of the texts to be analyzed. Besides, while *n-grams* of characters used for language modeling, some independent assumptions need to make where those assumptions are important as those can massively simplify the problem of learning language model from large text. Furthermore, multilingualism can be deal using this model. Our proposed method is based on some steps. To explain our method precisely we will use an example for each step. Afterward, we will illustrate more about our method. The steps of our method are given below:

Step 1: Assembling two large datasets from different web pages, journals, blogs, pdf file, doc file, text file etc. Let, suppose two small texts which are:

Text1: Computing.
Text2: Recomputing.

Step 2: As we know, stop words, punctuations and special characters etc. may or might commonly use in given texts, we would like to remove those so that we can focus on the important words instead. Besides, the raw and large data-files might consume more process time and memory space while creating *n-grams of characters*. So, it is very necessary to preprocess those raw data-files. For Preprocessing those raw data-files we would like to remove special characters, punctuations and stop-words separately. To get better result, converting all upper-case letter into lower case. So, both texts will be:

Text1: computing
Text2: recomputing

Step 3: Producing *n-grams* of characters using those data separately and placing those in a set. We tried all the possible size of *n-grams* (i.e. *n-gram* = 1, 2, 3, 4 ...), but lastly, we get the best result using *"trigram"* (using the size of *n-gram* = 3). Mentioned example will be:

Text1: 'com', 'omp', 'mpu', 'put', 'uti', 'tin', 'ing'.
Text2: 'rec', 'eco', 'com', 'omp', 'mpu', 'put', 'uti', 'tin', 'ing'.

Step 4: Measuring distance between *n-grams* of characters. Basically, the following two distance matrices describe a certain measure of distance between two pair of *n-grams* of characters which lead to measure the similarity between two large texts precisely. From our previous step, we got the *n-grams* of character of two texts. Now, the distance between those *n-gram* of characters are given below:

Table 7: Distance Matrix1

(i, j)	com	omp	mpu	put	uti	tin	ing
com	0	1	2	3	4	5	6
omp	-1	0	1	2	3	4	5
mpu	-2	-1	0	1	2	3	4
put	-3	-2	-1	0	1	2	3
uti	-4	-3	-2	-1	0	1	2
tin	-5	-4	-3	-2	-1	0	1
ing	-6	-5	-4	-3	-2	-1	0

Table 8: Distance Matrix2

(i, j)	rec	eco	com	omp	mpu	put	uti	tin	ing
rec	0	1	2	3	4	5	6	7	8
eco	-1	0	1	2	3	4	5	6	7
com	-2	-1	0	1	2	3	4	5	6
omp	-3	-2	-1	0	1	2	3	4	5
mpu	-4	-3	-2	-1	0	1	2	3	4
put	-5	-4	-3	-2	-1	0	1	2	3
uti	-6	-5	-4	-3	-2	-1	0	1	2
tin	-7	-6	-5	-4	-3	-2	-1	0	1
ing	-8	-7	-6	-5	-4	-3	-2	-1	0

Above mentioned distance matrices are hollow matrixes where we find some explicit matching distance between two *n-grams* of characters denoted by 0. For example - *('com', 'com') = 0, ('omp', 'omp') = 0, ('rec', 'rec') = 0* and so on. The increased positive numeric value of each row from both tables represents a strong positive correlation among two *n-grams* of characters i.e. *('com', 'omp') = 1*, *('eco', 'com') = 1*, *('omp', 'ing') = 5* and so on. Simultaneously, those positive values signify the distance as well. On the other hand, the increased negative value of each row indicates strong negative correlation between two *n-grams* of characters. For example – *('mpu', 'com') = -2*, *('ing', 'rec') = -8* etc. As we know, the distance cannot be negative value and to meet the criteria of a matrix distance all the off-diagonal entries must be positive value where $a_{ij} > 0$ *if* $i \neq j$. So, we will remove those negative values afterward. In this way, our two distance matrices satisfy all properties.

Step 5: Removing all negative values from both distance matrices. So, the distance matrices are:

Table 9: Matrix1 without negative values

(i, j)	com	omp	mpu	put	uti	tin	ing
com	0	1	2	3	4	5	6
omp		0	1	2	3	4	5
mpu			0	1	2	3	4
put				0	1	2	3
uti					0	1	2
tin						0	1
ing							0

Table 10: Matrix2 without negative values

(i, j)	rec	eco	com	omp	mpu	put	uti	tin	ing
rec	0	1	2	3	4	5	6	7	8
eco		0	1	2	3	4	5	6	7
com			0	1	2	3	4	5	6
omp				0	1	2	3	4	5
mpu					0	1	2	3	4
put						0	1	2	3
uti							0	1	2
tin								0	1
ing									0

Step 6: Measuring the similarity and dissimilarity from these two matrixes. From above two matrices, we get similarities between them if and only if $a_{1ij} = a_{2ij}$ = distance value. For example – the matching true matching values between two matrices are ('com', 'com', 0), ('put', 'uti', 1), and so on. Beside rest of the word pairs with distance values are dissimilar. That means the word 'computing' and 'recomputing' is 62.22% similar. Whether the dissimilarities between them is only 37.78%.

Step 7: Calculating the similarity and dissimilarity scores more preciously by using five popular co-occurrence measures: Jaccard, Dice, Overlap, Cosine and Simple.

Sometimes according to the values of five co-occurrences it would be very hard to take right decision. To solve this problem, let us consider a threshold value for similarities is – $\alpha \geq 0.3$ to take the right decision. That means if we find any of the values of five co-occurrences for measuring similarity is equal or greater than the threshold value (α), then those sentences are similar to each other.

Here, the matrices use association ratios between n-grams that are computed using their co-occurrence frequency in datasets. The basic assumption of this approach is that high co-occurrence frequencies indicate high association ratios and high association ratios indicate a semantic relation between n-grams. The five similarity measures proposed here are based on the five commonly used measures of association in information retrieval. The combination of row and column from each matrix used here which are produced from the set of *n-grams* of characters and the counting measure |.| gives the size of the set. Let suppose, the set for Matrix1 is A and Matrix2 is B which are given below respectively:

A = {[‘com’, ‘com’, 0], [‘com’, ‘omp’, 1], [‘com’, ‘mpu’, 2], [‘com’, ‘put’, 3], [‘com’, ‘uti’, 4], [‘com’, ‘tin’, 5], [‘com’, ‘ing’, 6], [‘omp’, ‘omp’, 0], [‘omp’, ‘mpu’, 1], [‘omp’, ‘put’, 2], [‘omp’, ‘uti’, 3], [‘omp’, tin’, 4], [‘omp’, ‘ing’, 5], [‘mpu’, ‘mpu’, 0], [‘mpu’, ‘put’, 1], [‘mpu’, ‘uti’, 2], [‘mpu’, ‘tin’, 3], [‘mpu’, ing’, 4], [‘put’, ‘put’, 0], [‘put’, ‘uti’, 1], [‘put’, ‘tin’, 2], [‘put’, ‘ing’, 3], [‘uti’, ‘uti’, 0], [‘uti’, ‘tin’, 1], [‘uti’, ‘ing’, 2], [‘tin’, ‘tin’, 0], [‘tin’, ing’, 1], [‘ing’, ‘ing’, 0]}.

B = {[‘rec’, ‘rec’, 0], [‘rec’, ‘eco’, 1], [‘rec’, ‘com’, 2], [‘rec’, ‘omp’, 3], [‘rec’, ‘mpu’, 4], [‘rec’, ‘put’, 5], [‘rec’, ‘uti’, 6], [‘rec’, ‘tin’, 7], [‘rec’, ‘ing’, 8], [‘eco’, ‘eco’, 0], [‘eco’, ‘com’, 1], [‘eco’, ‘omp’, 2], [‘eco’, mpu’, 3], [‘eco’, ‘put’, 4], [‘eco’, ‘uti’, 5], [‘eco’, ‘tin’, 6], [‘eco’, ing’, 7], [‘com’, ‘com’, 0], [‘com’, ‘omp’, 1], [‘com’, ‘mpu’, 2], [‘com’, ‘put’, 3], [‘com’, ‘uti’, 4], [‘com’, ‘tin’, 5], [‘com’, ‘ing’, 6], [‘omp’, ‘omp’, 0], [‘omp’, ‘mpu’, 1], [‘omp’, ‘put’, 2], [‘omp’, ‘uti’, 3], [‘omp’, tin’, 4], [‘omp’, ‘ing’, 5], [‘mpu’, ‘mpu’, 0], [‘mpu’, ‘put’, 1], [‘mpu’, ‘uti’, 2], [‘mpu’, ‘tin’, 3], [‘mpu’, ing’, 4], [‘put’, ‘put’, 0], [‘put’, ‘uti’, 1], [‘put’, ‘tin’, 2], [‘put’, ‘ing’, 3], [‘uti’, ‘uti’, 0], [‘uti’, ‘tin’, 1], [‘uti’, ‘ing’, 2], [‘tin’, ‘tin’, 0], [‘tin’, ing’, 1], [‘ing’, ‘ing’, 0]}.

Now, we can compute five co-occurrences measures using both sets which are drawn above in the 'Similarity Measure' section. From that section, we will find several equations to measure the five co-occurrences in the following way:

1) Jaccard Similarity Coefficient – using these two sets – *A* and *B*, we can measure the demanded values, given below:

   Intersection Value of these sets $(A \cap B) = 28$

   Total number of elements in set $A = 28$

   Total number of elements in set $B = 45$

   Now, from the equation (i), the Jaccard similarity coefficient is –

$$J(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{28}{28 + 45 - 28} = 0.6222222$$

2) Jaccard Dissimilarity Coefficient – using these two sets – A and B, we can measure the demanded values, given below:

   Intersection Value of these sets $(A \cap B) = 28$

   Union Value of these sets $(A \cup B) = 45$

   Now, inserting these values into the equation (ii), we will find the Jaccard dissimilarity coefficient –

$$d_j(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} = \frac{45 - 28}{45} = 0.3777778$$

3) Dice Similarity Coefficient – the intersecting value and the total number of both set *A* and *B* is following:

   Intersection Value $(A \cap B) = 28$

   Number of elements in set $A = 28$

   Number of elements in set $B = 45$

   From the equation (iii), the dice similarity coefficient is –

$$Dice(A,B) = \frac{2 \times |A \cap B|}{|A| + |B|} = \frac{2 \times 28}{28 + 45} = 0.7671233$$

4) Dice Dissimilarity Coefficient – we know the intersecting value of the given sets and the total number of both set *A* and *B* separately is following:

Intersection Value *(A ∩ B)* = 28

Number of elements in set *A* = 28

Number of elements in set *B* = 45

From the equation (iv), the dice dissimilarity coefficient is –

$$d = 1 - \frac{2 \times |A \cap B|}{|A| + |B|} = 1 - \frac{2 \times 28}{28 + 45} = 0.2328767$$

5) Overlap Coefficient – we have the intersect value of these sets and the total number of elements in both sets separately as well. from equation (v) the overlap coefficient is –

Intersection Value *(A ∩ B)* = 28

Number of elements in set *A* = 28

Number of elements in set *B* = 45

$$overlap(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)} = \frac{28}{\min(28, 45)} = 1$$

6) Cosine Similarity – from the set *A* and *B*, the intersection value is – 28 and the numbers of elements in set *A* and *B* are – 28 and 45 respectively. So, from the equation (vi), the cosine similarity value is –

$$C(A, B) = \frac{|A \cap B|}{sqrt(|A|) \times sqrt(|B|)} = \frac{28}{sqrt(28) \times sqrt(45)} = 0.7888106$$

7) Simple Matching Coefficient (SMC) – for the equation (vii), we need the number of matching attributes that means the intersection value of the given sets and also the total number of attributes that means the union value of both set which are given below –

Intersection Value *(A ∩ B)* = 28

Union Value *(A ∪B)* = 45

Now, from equation (7), the simple matching coefficient is –

$$SMC = \frac{number\ of\ marching\ attributes}{number\ of\ attributes} = \frac{|A \cap B|}{|A \cup B|} = \frac{28}{45} = 0.6222222$$

Now, the collection of all coefficients into a table is given below –

Table 11: Similarity and Dissimilarity Coefficients

Co-occurrence Measures	Values
Jaccard Coefficient for Similarity	0.6222222
Jaccard Coefficient for Dissimilarity	0.3777778
Dice Similar Coefficient	0.7671233
Dice Dissimilarity Coefficient	0.2328767
Overlap Similar Coefficient	1
Cosine Similarity Measure	0.7888106
Simple Matching Coefficient	0.6222222

As we mentioned earlier in the last paragraph of section *'N-grams'* that, we have a scale (0, 1) to understand the similarity and dissimilarity between two texts more precisely where 0 and 1 will signify the intense similarity or dissimilarity. Here, Jaccard and SMC give the same value – 0.6222, where Dice and Cosine give almost close values i.e. 0.767 and 0.789 respectively. It is noticeable that, the similarity values of Jaccard, Dice, Cosine and Simple Matching are greater than the threshold value (α) i.e. $\alpha \geq 0.3$. Besides, the overlap coefficient provides 1 that means these texts are pretty similar. So, form the above table, we can conclude that we got some satisfactory similar coefficient values.

3.4 Several Examples

For another evaluation, let us consider *tri-grams of characters*. At first, we will collect our text from different kind of web-pages, blogs, journals etc. So, we have applied our algorithm on more than 30 sentences which are collected from a webpage like "Wikipedia" and we got a satisfactory result. We took 15 sentences from Wikipedia of both keywords like "**Apple Inc.**" as company name and "**Apple**" as fruit name separately. There have many other sources for large text files, because of the versatile collection of sentences, we choose Wikipedia. Besides, our local system has some limitations to process large files and that's why we took only 30 sentences for further procedure. Using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following result:

Table 12: Similarity Measure between 30 Sentences

Co-occurrence Measures	Values
Jaccard Coefficient for Similarity	0.0007952482
Jaccard Coefficient for Dissimilarity	0.9992048
Dice Similar Coefficient	0.0015866
Dice Dissimilarity Coefficient	0.9984134
Overlap Similar Coefficient	0.002578805
Cosine Similarity Measure	NAN
Simple Matching Coefficient	0.0007932998

Per our opinion, these two keywords are totally different, so their texts must be different as well. From the above table, the dissimilar coefficient is – $0.99 \approx 1$. That means these two large sentences are quite dissimilar to each other. On the other hand, the similarity values are less than the threshold value (α). So, we can conclude that our methodology provides a satisfactory result.

Now, we will provide some examples to verify that our methodology is free from any grammatical rules of English and French language respectively. As we know, speakers of any language use many forms of grammatical rules to give clear instructions as well descriptions in all sources of information. Because of complying the instructions of grammar step by step, the written language tends to be more standardized nowadays. So, we can say that every source of information follows the rules of grammar and there have a lot of vocabularies. There have many categorizations of grammatical rules. At first, we will demonstrate about the English grammatical rules with examples and afterwards we will discuss about French with examples in a following way:

1. **Noun** in a simple sentence / phrase / clause – consists of name of person, place or thing etc.

   For example: let consider two noun phrase i.e. –

   text1 – 'Karen rode on a yellow skate board.' and
   text2 – 'Karen lives in the yellow house.'

   Result: Using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following result:

Table 13: Similarity Measure of Noun Phrases

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.05166667
Jaccard Coefficient (Dis-similarity)	0.9483333
Dice Similar Coefficient	0.09825674
Dice Dissimilar Coefficient	0.9017433
Overlap Similar Coefficient	0.1225296
Cosine Similarity Measure	0.1002433
Simple Matching Coefficient	0.05166667

As per our opinion, these two sentences are totally different. Besides, from our above table, our dissimilarity section contains highest values – the dissimilarity coefficient of Jaccard and Dice are 0.95 and 0.9 respectively which is ≈ 1 that means the given sentences are dissimilar. Besides, the similar values like Jaccard, Dice, Overlap, Cosine and Simple matching are less than the threshold value ($\alpha \geq 0.3$). So, it has been proved that these two sentences are dissimilar.

2. **Declarative Sentence** – simply makes a statement or expresses an opinion. In other words, it makes a declaration. This kind of sentence ends with a period. For example – let us consider two text like:

text1 – 'I want to be a good writer.' and
text2 – 'My friend is a really good writer.'

Now using the steps as well as (i), (ii), (iii), (iv), (v), (vi) and (vii) equations, we get following results:

Table 14: Similarity Measure of Declarative Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.1687117
Jaccard Coefficient (Dis-similarity)	0.8312883
Dice Similar Coefficient	0.2887139
Dice Dissimilar Coefficient	0.7112861
Overlap Similar Coefficient	0.5238096
Cosine Similarity Measure	0.3230825
Simple Matching Coefficient	0.1687117

As our opinion, these two sentences are quite different but the quality i.e. 'good writer' of the two-different subject is same. Among these two sentences, the first sentence expresses about the future aim of someone where the second sentence express about the present situation of someone.

Simultaneously, our method gives the exact result which is exactly in our opinion. From our table, the dissimilarity values are higher than similarity value. From our table the got the similarity values – 0.17, 0.29, 0.52, 0.32, 0.17 given by Jaccard, Dice, Overlap Cosine and Simple Matching respectively where the Jaccard and Dice give the dissimilar value i.e. 0.83 and 0.71 respectively. Because of the adjective 'good writer' of these two sentences, the Overlap and Cosine provide the similarity value i.e. 0.52 and 0.32 which are greater than our threshold value ($\alpha \geq 0.3$). But from the table, the similar values of Jaccard, Dice and Simple Matching are very less than our threshold value ($\alpha \geq 0.3$) which satisfy our expectation.

3. **Imperative Sentences** – gives a command or makes a request. It usually ends with a period but can, under certain circumstances, end with an exclamation point. For example – let us consider two sentences:

text1 – 'Please sit down.' and
text2 – 'I need you to sit down now!'

Using the seven noted equations as well as the steps, we get following results:

Table 15: Similarity Measure of Imperative Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.04761905
Jaccard Coefficient (Dis-similarity)	0.952381
Dice Similar Coefficient	0.09090909
Dice Dissimilar Coefficient	0.9090909
Overlap Similar Coefficient	1
Cosine Similarity Measure	0.2182179
Simple Matching Coefficient	0.04761905

According to our results, we get less similar value than dissimilar value except for Overlap coefficient. Here, overlap coefficient gives the similar value – 1, that means this coefficient shows the intense similarity between the given sentences which satisfy our expectation.

4. **Interrogative Sentence** – asks a question. This type of sentence often begins with who, what, where, when, why, how, or do, and it ends with a question mark. For example – let us consider two texts:

   text1 – 'When are you going to turn in your writing assignment?' and
   text2 – 'Do you know what the weather will be tomorrow?'.

   Now, using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following result:

Table 16: Similarity Measure of Interrogative Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

Basically, as our perception, these two sentences are totally dissimilar. The table above also shows the dissimilarity between the given sentences.

5. **Exclamatory Sentence** – is a sentence that expresses great emotion such as excitement, surprise, happiness and anger, and ends with an exclamation point. For example – let us consider two texts like:

   text1 – 'It is too dangerous to climb that mountain!' and
   text2 – 'I got an A on my book report!'.

   Now the described steps and the seven noted equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are mentioned above, we get following results:

Table 17: Similarity Measure of Exclamatory Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

The table shows that these two sentences are totally dissimilar to each other which satisfy our expectation.

6. **Simple sentence** – has one independent clause. For example – let us consider,

text1 – 'Tom reads novels.' and
text2 – 'Tom reads and enjoys novels.'

Using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following result:

Table 18: Similarity Measure of Simple Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.2043011
Jaccard Coefficient (Dis-similarity)	0.7956989
Dice Similar Coefficient	0.3392857
Dice Dissimilar Coefficient	0.6607143
Overlap Similar Coefficient	0.5428572
Cosine Similarity Measure	0.3659942
Simple Matching Coefficient	0.2043011

As per our opinion these two sentences are similar. According to our result we get similar values like 0.204, 0.339, 0.543, 0.366, 0.204 are given by Jaccard, Dice, Overlap, Cosine, Simple similarity coefficients respectively. Here, the similar values of Dice, Overlap and

Cosine are greater than our assumed threshold ($\alpha \geq 0.3$) which means these similar values shows us the similarity between these two sentences. So, we can say that these two sentences are similar.

7. **Compound Sentence** – has two independent clauses joined by
   a. A coordinating conjunction (e.g. for, and, nor, but, or, yet, so),
   b. A conjunctive adverb (e.g. however, therefore) or
   c. A semicolon alone.

For example – let us consider two sentences like

text1 – 'Tom reads novels, but Jack reads comics.' and
text2 – 'Tom reads novels; however, Jack reads comics.'

Using the steps and (i), (ii), (iii), (iv), (v), (vi) and (vii) equations which are described above, we get the following results:

Table 19: Similarity Measure of Compound Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.9447514
Jaccard Coefficient (Dis-similarity)	0
Dice Similar Coefficient	0.9715909
Dice Dissimilar Coefficient	0.02840909
Overlap Similar Coefficient	0.9715909
Cosine Similarity Measure	0.9715909
Simple Matching Coefficient (SMC)	1

As per our perception these two sentences are approximately alike each other. From the table, we get the highest similarity in the scale of (0, 1) – 0.945, 0.97, 0.97, 0.97 and 1 are given by Jaccard, Dice, Overlap, Cosine and Simple matching similarity coefficient respectively and these values are greater than our threshold value ($\alpha \geq 0.3$). Here, the SMC gives the highest similarity which is 1. On the other hand, we get dissimilarity value is 0 from Jaccard dissimilar coefficient but we get a little – 0.02841 (28.41%) dissimilar value from Dice dissimilar coefficient. So, we found that these two sentences are similar.

8. **Complex Sentence** – has one dependent clause (headed by a subordinating conjunction or a relative pronoun) joined to an independent clause.

For example1 – let us consider two sentences,

text1 – 'Although Tom reads novels, Jack reads comics.' and
text2 – 'Jack reads comics although Tom reads novels.'

Using the steps and (i), (ii), (iii), (iv), (v), (vi) and (vii) equations, we get following results:

Table 20: Similarity Measure of Complex Sentences (ex1)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.248227
Jaccard Coefficient (Dis-similarity)	0.7426471
Dice Similar Coefficient	0.3977273
Dice Dissimilar Coefficient	0.6022727
Overlap Similar Coefficient	0.3977273
Cosine Similarity Measure	0.3977273
Simple Matching Coefficient	0.2573529

According to the above table the similarity values are – 0.25, 0.398, 0.398, 0.398, 0.26 which are given by Jaccard, Dice, Overlap, Cosine, Simple Matching coefficient respectively. We can observe that the Dice, Overlap and Cosine give same similar values that is – 0.3977273 and this value is greater than our threshold value ($\alpha \geq 0.3$). So, we can say that these two sentences are similar.

For example2 – let us consider:

text1 – 'Jack Smith, who reads comics, rarely reads novels.' and
text2 – 'People like Jack Smith, who reads comics rarely reads novels.'

So, using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following result:

Table 21: Similarity Measure of Complex Sentences (ex2)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.5850623
Jaccard Coefficient (Dis-similarity)	0.4025424
Dice Similar Coefficient	0.7382199
Dice Dissimilar Coefficient	0.2617801
Overlap Similar Coefficient	0.9825784
Cosine Similarity Measure	0.762165
Simple Matching Coefficient	0.5974576

According to our perception, these two sentences are similar because these two sentences are mainly talking about the same verb and objects of some similar personality. From the above table the similarity values are – 0.5851, 0.7382, 0.9826, 0.7622 and 0.5975 from Jaccard, Dice, Overlap, Cosine and Simple matching similarity coefficient respectively which are greater than our threshold value ($\alpha \geq 0.3$). As we can see that the Overlap gives the highest similar value – 0.9826 (98.26%). The dissimilar values are – 0.4025 (40.25%) and 0.2618 (26.18%) from Jaccard and Dice dissimilar coefficient. So, our method also satisfies our expectation that means these two sentences are similar.

9. **Compound-Complex Sentence –** has two independent clauses joined to one or more dependent clauses.

For example1 – let the two texts are:

tex1 – 'While Tom reads novels, Jack reads comics, but Sam reads only magazines.'
text2 – 'Tom reads novels, but Jack reads comics because books are too difficult.'

Now, implying the steps and (i), (ii), (iii), (iv), (v), (vi) and (vii) equations, we get following results:

Table 22: Similarity Measure of Compound-Complex Sentences (ex1)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.4307305
Jaccard Coefficient (Dis-similarity)	0.5581396
Dice Similar Coefficient	0.6021127
Dice Dissimilar Coefficient	0.3978873
Overlap Similar Coefficient	0.9715909
Cosine Similarity Measure	0.6510236
Simple Matching Coefficient	0.4418605

As our opinion, these two sentences are similar because these two sentences are talking about the same subject as well as same object. From our table, the similar values are – 0.431, 0.602, 0.9716, 0.651 and 0.442 given by Jaccard, Dice, Overlap, Cosine and Simple Matching coefficient respectively and these values are greater than our assumed threshold ($\alpha \geq 0.3$). Among those values, the Overlap gives the best result which is 0.9716 that means these two sentences are 97.16% similar which satisfy our expectation.

For example2 – Another example for this type of sentence is,

text1 – 'Jack, who reads comics, rarely reads novels; however, Tom enjoys novels.'
text2 – 'People like Jack, who reads comics rarely reads novels; they often find books difficult.'

So, using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following results:

Table 23: Similarity Measure of Compound-Complex Sentences (ex2)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.2328326
Jaccard Coefficient (Dis-similarity)	0.7628415
Dice Similar Coefficient	0.3777198
Dice Dissimilar Coefficient	0.6222802
Overlap Similar Coefficient	0.4549266
Cosine Similarity Measure	0.3832798
Simple Matching Coefficient	0.2371585

According to our perspective, these two sentences are similar because first sentence is talking about a specific person 'Jack', but the second sentence is also talking about of those persons who have similar personality like 'Jack' as well as explain the reason for finding difficulty to read novels unlike 'Tom'. Here, the similarity values of Dice, Overlap and Cosine are – 0.378, 0.455 and 0.3833 respectively and these values are greater than the threshold value ($\alpha \geq 0.3$). So, our method shows that these two sentences are similar which satisfy our expectation.

10. **Flat adverbs –** adjectives that do not change form (add -ly) to become adverbs are called flat adverbs. Typical flat adverbs are early, late, hard, fast, long, high, low, deep, near. For example – let us consider the word 'Early' using as an adjective and as an adverb in two different sentences are given below:

text1 – 'The early train arrives at 8.45 a.m.' ('Early' used as an adjective)
text2 – 'The 8.45 a.m. train arrived early.' ('Early' used as an adverb)

Now using our described steps and also the seven equations, we get the following result:

Table 24: Similarity Measure of Flat Adverb Sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.1583012
Jaccard Coefficient (Dis-similarity)	0.8416988
Dice Similar Coefficient	0.2733333
Dice Dissimilar Coefficient	0.7266667
Overlap Similar Coefficient	0.2733333
Cosine Similarity Measure	0.2733333
Simple Matching Coefficient	0.1583012

As our perception, these two sentences are dissimilar because first sentence provides the information that 'the early train' may or might arrives late which is '8.45 a.m.' whereas the exact meaning of second sentence is 'the train' of '8.45 a.m.' arrived before the time. For this example, the dissimilarity values are 84.2% and 73%, given by Jaccard and Dice dissimilar coefficient severally. Here, Jaccard and Simple Matching coefficient provide the same similar value – 0.158 (15.8%). Besides, the Dice, Overlap and Cosine provide the same similar value i.e. 0.273 (27.3%). So, these similarity values are less than our threshold value ($\alpha \geq 0.3$) which proves that these two sentences are dissimilar.

11. **Comparative and Superlative Sentence** – the adverb become comparative and superlative when the adjectives of more than one syllable. For example – let suppose,

text1 – 'Moe played the tune more lyrically than Sam.'
text2 – 'Jack played the tune most lyrically.'

Now using our steps and the seven equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) which are mentioned above, we get following results:

Table 25: Similarity Measure of comparative and superlative sentences

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.448718
Jaccard Coefficient (Dis-similarity)	0.551282
Dice Similar Coefficient	0.619469
Dice Dissimilar Coefficient	0.380531
Overlap Similar Coefficient	0.7
Cosine Similarity Measure	0.6236095
Simple Matching Coefficient	0.448718

According to our opinion these two sentences are similar because these three persons – 'Moe', 'Sam' and 'Jack' have same knowledge which is playing the tune but among them Jack is very good tune player. From the above table, the similar values are – 0.449, 0.619, 0.7, 0.624 and 0.449 given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficients respectively. Here, Jaccard and Simple matching provide same similar value. Among them the Overlap provide highest similar value. These similar values are greater than the threshold value ($\alpha \geq 0.3$) which indicate that the similarity among these two sentences.

Now we are going to discuss the comparison of the grammatical rules of French sentences using some grammatical rules which are given below –

12. Auxiliary Verbs – there are only two auxiliary verbs in French language – avoir and être. They are used to change the tense of the main verb. These two verbs are irregular verbs whose conjunctions must be memorized. They are important verbs because they serve both as auxiliary verbs and main verbs.

For Example1 – let us consider two French sentences using 'avoir' verb –

text1 – 'J'ai ce livre.' ('avoir' as main verb)
text2 – 'J'ai acheté ce livre.' ('avoir' as auxiliary verb)

So, using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following results:

Table 26: Similarity Measure between Two Sentences Contain 'Avoir' Verb

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.1142857
Jaccard Coefficient (Dis-similarity)	0.8857143
Dice Similar Coefficient	0.2051282
Dice Dissimilar Coefficient	0.7948718
Overlap Similar Coefficient	0.4444444
Cosine Similarity Measure	0.2434322
Simple Matching Coefficient	0.1142857

We have shown the use of avoir both as auxiliary verb and main verb and these two sentences are dissimilar per our opinion. From the above table, we can find the similarity values are – 0.114, 0.205, 0.243 and 0.114 given by Jacccard, Dice, Cosine and Simple matching coefficient which are less than our threshold value ($\alpha \geq 0.3$). On the other hand, because of the string 'j'ai' and 'livre', only the Overlap similar coefficient gives 0.444 similarity value which is greater than our threshold value ($\alpha \geq 0.3$). According to the scores of Jaccard, Dice, Cosine and Simple matching coefficients, we can conclude that these two sentences are dissimilar.

For Example2 – let us consider two French sentences using 'être' verb –

text1 – 'Je suis à la maison.' ('être' as main verb)
text2 – 'Je suis allé au restaurant.' ('être' as auxiliary verb)

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 27: Similarity Measure between Two Sentences Contain 'Être' Verb

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

From the above table, we find these two sentences are dissimilar which satisfy our expectation.

13. **Passé Composé –** the passé compose is formed with the auxiliary verb 'avoir' or 'être' conjugated in the present tense of avoir or être + past participle of main verb.

For example – let us consider the following sentences –

text1 – 'Le garçon a mangé la pomme.'
text2 – 'La fille a mangé la pomme.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 28: Similarity Measure between Sentences of Passé Composé

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.2736318
Jaccard Coefficient (Dis-similarity)	0.7263682
Dice Similar Coefficient	0.4296875
Dice Dissimilar Coefficient	0.5703125
Overlap Similar Coefficient	0.4583333
Cosine Similarity Measure	0.4305292
Simple Matching Coefficient	0.2736318

Though the subject of these two sentences are different but both of them are doing the same thing 'a mangé la pomme'. So, per our view these two sentences are similar. From this table, the similarity values are – 0.274, 0.4297, 0.458, 0.4305 and 0.274 given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient respectively. Among them, the similarity values of Dice, Overlap, Cosine are greater than the threshold value ($\alpha \geq 0.3$). according to these values we can say that these two sentences are similar.

14. **L'imparfait de L'indicatif** –It is used to express a repeated action in the past or an interrupted action in the past. It also expresses the difference between 'before' and 'now'. To form l'imparfait de l'indicatif, taking the plural form of first person and the present form of verb by adding for terminating – '-ons' '-ais', '-ait', '-ions', '-iez', '-aient' at the verbal base. Only exception for – 'être'.

For example – let us consider two l'imparfait de l'indicatif sentences which are given below:

text1 – 'Elle chantait quand je suis arrivé.'
text2 – 'Quand j'étais enfant je venais souvent à cet endroit.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 29: Similarity Measure between Sentences of L'imparfait de L'indicatif

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.002781641
Jaccard Coefficient (Dis-similarity)	0.9971949
Dice Similar Coefficient	0.00554785
Dice Dissimilar Coefficient	0.9944522
Overlap Similar Coefficient	0.02197802
Cosine Similarity Measure	0.008352933
Simple Matching Coefficient	0.002805049

According to our belief these two sentences are totally dissimilar. From the table, we can see that the similarity values are – 0.0028, 0.00555, 0.02198, 0.008353 and 0.0028 are given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient and these values are lower than the threshold value ($\alpha \geq 0.3$). So, its proved that these sentences are totally dissimilar.

15. **Le Plus-que-parfait de L'indicatif** – le plus-que-parfait used with another tense of past (passé-composé or passé simple) allows to express the priority of an action in relation to a past action. For example – let us consider two sentences of le plus-que-parfait:

text1 – 'Elle m'a dit qu'il ne l'avait jamais vu.'
text2 – 'Elle me raconta qu'il ne l'avait jamais vu.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 30: Similarity Measure between Sentences of le Plus-que-parfait de L'indicatif

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.3710938
Jaccard Coefficient (Dis-similarity)	0.62818
Dice Similar Coefficient	0.5413105
Dice Dissimilar Coefficient	0.4586895
Overlap Similar Coefficient	0.5413105
Cosine Similarity Measure	0.5413105
Simple Matching Coefficient	0.37182

According to our opinion these two sentences are similar. From our table, the similar values are – 0.371, 0.541, 0.541, 0.541 and 0.372 given by Jaccard, Dice, Overlap, Cosine, Simple matching coefficient respectively. Here, its noticeable that the similar value of Dice Overlap and Cosine is same – 0.541 and the value of Jaccard and Simple matching is same 0.371 ≈ 0.372. Besides, all values are greater than the threshold value (α ≥ 0.3). So, we can definitely demand that these two sentences are similar.

16. **Le Futur Simple –** the simple future expresses a fact or an action that will take place later and it has not yet taken place at the moment when we express ourselves.

For example – let us consider two sentences of le future simple:

text1 – 'La semaine prochaine nous partirons en Grèce pour les vacances de Pâques.'
text2 – 'Nous allons partir en Grèce pour les vacances de Pâques.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 31: Similarity Measure between Sentences of le Futur Simple

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.1747419
Jaccard Coefficient (Dis-similarity)	0.8238591
Dice Similar Coefficient	0.2974983
Dice Dissimilar Coefficient	0.7025017
Overlap Similar Coefficient	0.6267806
Cosine Similarity Measure	0.3496347
Simple Matching Coefficient	0.1761409

Though the first sentence is finite i.e. it is defined that 'la semain prochaine' some people must go to Greece for ester holiday whereas the second sentences is infinite, but the main verb as well as object are similar. So, we will consider these two sentences are similar.

Now, from the table, the lowest similar values are given by Jaccard and Simple matching coefficient and these two are almost similar values – 0.1747 ≈ 0.1761. Besides, the similarity value of Dice similar coefficient is almost same to the threshold value (α ≥ 0.3) which is 0.2974983 ≈ 0.3. On the other hand, the similar values of Overlap and Cosine similar coefficient are – 0.626781 and 0.349635, greater than the threshold value (α ≥ 0.3). So, it is clear that these two sentences are similar.

17. **Futur Antérieur** – The futur antérieur corresponds to the future perfect in English. It indicates a supposition that an action will have been completed by the time of speaking, or by a specified point in the future.

For example – let us consider two sentences of future antérieur:

text1 – 'Je commencerai à travailler lorsque j'aurai terminé mon déjeuner.'
text2 – 'Je me serai trompé dans mon calcul.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 32: Similarity Measure between Sentences of Futur Antérieur

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

As per our perception these two sentences are dissimilar and this table - 31 satisfy our expectation.

18. **Les Prépositions de Temps** – there have so many different prépositions de temps in French with different uses – à, en, dans, depuis, pendant, durant, and pour (although pour is almost never used to express time). Among them we will give example of 'depuis' which referes to the duration of something that is still going on in the present, or was still going on when something else happened.

for example – let us consider two sentences of 'depuis' préposition de temp:

text1 – 'Je suis à Montréal depuis une semaine.'
text2 – 'Je suis à Montréal depuis le 3 juillet.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 33: Similarity Measure between Sentences of 'Depuis' Préposition de Temp

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.1314554
Jaccard Coefficient (Dis-similarity)	0.8685446
Dice Similar Coefficient	0.2323651
Dice Dissimilar Coefficient	0.7676349
Overlap Similar Coefficient	0.2666667
Cosine Similarity Measure	0.2343117
Simple Matching Coefficient	0.1314554

As our perception, the meaning of these two sentences are different. From our table, the similar values are – 0.131, 0.232, 0.267, 0.234 and 0.131 given by Jaccard, Dice, Overlap, Cosine and Simple Matching coefficients and all of these values are very lower than the threshold value ($\alpha \geq 0.3$). So, according to the result, we can definitely conclude that these two sentences are dissimilar which satisfy our expectation.

19. **Le Futur Proche** – to express an action that will take place in a future very close to the present, we are using near future. The futur proche is usually translated into English as going + infinitive (e.g., going to eat, going to drink, going to talk). The futur proche is characteristic of spoken French but may be used in informal writing. It is formed with the verb aller (to go) conjugated in the present tense followed by an infinitive.

For example – let us consider two sentences of le future proche, given below:

text1 – 'Je vais prendre une douche.'
text2 – 'Il va prendre une douche.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 34: Similarity Measure between Sentences of le Futur Proche

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	1
Jaccard Coefficient (Dis-similarity)	0
Dice Similar Coefficient	1
Dice Dissimilar Coefficient	0
Overlap Similar Coefficient	1
Cosine Similarity Measure	1
Simple Matching Coefficient	1

These two sentences are similar as per our perception. We also can notice that, we get all similar values are – 1 which means that these two sentences are similar.

20. **L'indicatif Présent** – l'indicatif présent is a time of simple verb which is part of the indicative mode and which situates the fact at the moment of the utterance. It expresses a fact or an action that takes place at the moment when we express ourselves.

for example1 – let us consider two sentences of l'indicatif présent of 1er groupe (-er):

text1 – 'Je mange à la cafétéria chaque jour.'
text2 – 'Il mange à la cafétéria chaque jour.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 35: Similarity Measure between Sentences of L'indicatif Present (ex1)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	1
Jaccard Coefficient (Dis-similarity)	0
Dice Similar Coefficient	1
Dice Dissimilar Coefficient	0
Overlap Similar Coefficient	1
Cosine Similarity Measure	1
Simple Matching Coefficient	1

These two sentences are similar as per our opinion. We also get 100% similarity between them from the table mentioned above.

For example2 – let us consider two sentences of l'indicatif présent of 2e groupe (-ir):

text1 – 'Le professeur répète toujours la même chose.'
text2 – 'Elle s'appelle Marie.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 36: Similarity Measure between Sentences of L'indicatif Present (ex2)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

These two sentences are definitely dissimilar according to our view and we also found the same result from the given table.

For example3 – let us consider two sentences of l'indicatif présent of 3e groupe (-ir):

text1 – 'Je sais cela depuis lundi.'
text2 – 'Je sais cela depuis une semaine.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 37: Similarity Measure between Sentences of L'indicatif Present (ex3)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.07058824
Jaccard Coefficient (Dis-similarity)	0.9294118
Dice Similar Coefficient	0.1318681
Dice Dissimilar Coefficient	0.8681319
Overlap Similar Coefficient	0.1666667
Cosine Similarity Measure	0.13484
Simple Matching Coefficient	0.07058824

As per our opinion the meaning of these two sentences are dissimilar. From our table, we get the similar values are – 0.0706, 0.1319, 0.1667, 0.1348 and 0.0706 given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient respectively and these values are lower than the threshold value ($\alpha \geq 0.3$). So, we definitely say that these two sentences are dissimilar.

21. **L'impératif présent** – l'impératif présent is used to formulate an order, a request, a council. It is intended to set fort an order or a prohibition.

For example – let us consider two sentences of l'impératif présent which are given below:

text1 – 'Finissez vos devoirs!'
text2 – 'Ne discute pas!'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 38: Similarity Measure between Sentences of L'impératif Présent

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

These two sentences are dissimilar according to our opinion. From this table, we also find the dissimilarity between them.

22. **Les Pronoms compléments** – les pronoms compléments are used to replace a name. They are always placed before the verb except to the imperative of affirmative form. Les pronoms personnels compléments d'objets directs (COD) replace names of things or defined person. They answer question: 'what?' or 'who?'. Besides, les pronoms personnels complément d'objet indirect (COI) replace names of persons preceded by the preposition 'to'. They answer the question: 'to whom?'.

For example1 – let us consider two sentences of COD:

Text1 – 'Tu chantes la Marseillaise. Tu la chantes.'
Text2 – 'Vous avez rencontré Marie. Vous l'avez rencontrée.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 39: Similarity Measure between Sentences of COD

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.003554502
Jaccard Coefficient (Dis-similarity)	0.9962547
Dice Similar Coefficient	0.007083825
Dice Dissimilar Coefficient	0.9929162
Overlap Similar Coefficient	0.008547009
Cosine Similarity Measure	0.007189966
Simple Matching Coefficient	0.003745318

As per our opinion these two sentences are dissimilar. From the above table, the similar values are – 0.0035, 0.0071, 0.0085, 0.0072 and 0.0037 which are given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient and these values are lower than the threshold value ($\alpha \geq 0.3$). So, we can say that these two sentences are dissimilar.

For example2 – let us consider two sentences of COI:

text1 – 'Il téléphone à mon amie. Il lui téléphone.'
text2 – 'Il téléphone à mon amie. Il lui téléphone.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 40: Similarity Measure between Sentences of COI

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.8007118
Jaccard Coefficient (Dis-similarity)	0
Dice Similar Coefficient	0.8893281
Dice Dissimilar Coefficient	0.1106719
Overlap Similar Coefficient	0.8893281
Cosine Similarity Measure	0.8893281
Simple Matching Coefficient	1

These two sentences are similar according to our view. From this table, the Dice, Overlap and Cosine give the same similar value i.e. 0.8893281 which is greater than our threshold value ($\alpha \geq 0.3$). Besides, the Jaccard gives the similar value is – 0.8007118 and the Simple matching coefficient gives the similar value is – 1. So, it is confirmed that these two sentences are similar.

23. Les Déterminants Possesifs – les déterminants possesifs agree in gender and number with the name they determine. As the name indicates, they serve to designate a possession, a relationship of belonging, paternity, origin etc.

For example – let us consider two sentences of le déterminant possessifs :

text1 – 'Ton chien est petit, mais le mien est gros.'
text2 – 'Ton chien est petit, mais le sien est gros.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 41: Similarity Measure between Sentences of les Déterminants Possesifs

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	1
Jaccard Coefficient (Dis-similarity)	0
Dice Similar Coefficient	1
Dice Dissimilar Coefficient	0
Overlap Similar Coefficient	1
Cosine Similarity Measure	1
Simple Matching Coefficient	1

According to our view, these two sentences are similar. From above table, we also find that all similarity values are – 1 which means these two sentences are similar.

24. La Négation – the place of la négation differs depending on whether it concerns verb or a verb infinitive. La négation always contains two elements: ne ... pas / plus, ne ... jamais, ne … rien, personne ne … etc.

For example – let us consider two sentences of le negation which are given below:

text1 – 'Je ne crois pas qu'il faille se fâcher.'
text2 – 'Je n'ai pas voulu en arriver là.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 42: Similarity Measure between Sentences of la Négation

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

These two sentences are dissimilar as per our opinion. We also find that, all similar values are – 0 which define that these two sentences are dissimilar.

25. **L'interrogation** – l'interrogation is asking questions in different ways. In French for asking questions we use – 'Où?', 'Quand?', 'Comment?', 'Combien?', 'Pourquoi?' etc.

For example – let us consider two sentences of l'interrogation:

text1 – 'Que fais-tu ce matin?'
text2 – 'Qu'est-ce que tu fais ce matin?'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 43: Similarity Measure between Sentences of L'ingerrogation

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	1
Jaccard Coefficient (Dis-similarity)	0
Dice Similar Coefficient	1
Dice Dissimilar Coefficient	0
Overlap Similar Coefficient	1
Cosine Similarity Measure	1
Simple Matching Coefficient	1

These two sentences are similar according to our opinion. From the table, all similar values are – 1 which means that these two sentences are similar.

26. Les Adjectifs Démonstratifs – les adjectifs démonstratifs is used to designate something or someone or to take back a name already quoted.

For example – let us consider two sentences of les adjectifs démonstratifs:

text1 – 'Cet appartement est moderne.'
text2 – 'Cet hôtel est splendide.'

Now, using the seven steps as well as the equations - (i), (ii), (iii), (iv), (v), (vi) and (vii) we get the following table –

Table 44: Similarity Measure between Sentences of les Adjectifs Démonstratifs

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice Similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Similar Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

As per our opinon, these two sentences are dissimilar because first sentence is talking about an apartment whether the second sentence is talking about a hotel. From the above table, we find all similarity values are – 0 which define that these two sentences are dissimilar.

Now using some translations as well grammatical rules of sentences from English ⇋ French, to find out the similarities between two sentences.

27. Present Tense – Let us consider two examples of multilingualism which are given below respectively:

For example1: let us consider a sentence in English as text1 and the French translation of that is in text2, given below:

text1 – 'I am going to university.'
text2 – 'Je vais à l'université.'

Now comparing these two sentences, using our described steps and the seven equations, we get the following results:

Table 45: Multilingualism of Present Tense Sentences (ex1)

Jaccard Coefficient (Similarity)	0.2121212
Jaccard Coefficient (Dis-similarity)	0.7878788
Dice Similar Coefficient	0.35
Dice Dissimilar Coefficient	0.65
Overlap Similar Coefficient	0.5090909
Cosine Similarity Measure	0.3684529
Simple Matching Coefficient	0.2121212

As per our opinion these two sentences are similar. From our result, we can see that the five-similarity coefficient give a very good result i.e. 0.21, 0.35, 0.51, 0.37, 0.18 and those are given by Jaccard, Dice, Overlap, Cosine, Simple similar coefficient respectively. besides, the similar values of Dice, Overlap and Cosine are greater than the threshold value ($\alpha \geq 0.3$). So, we can definitely say that these two sentences are similar.

For example2 – now considering another example where:

text1 – 'I am reading philosophy.'
text2 – 'Je lis de la philosophie.'

Now, implying the described steps and also the seven equations, we get the following results:

Table 46: Multilingualism of Present Tense Sentences (ex2)

Jaccard Coefficient (Similarity)	0.1884817
Jaccard Coefficient (Dis-similarity)	0.8105263
Dice Similar Coefficient	0.3171806
Dice Dissimilar Coefficient	0.6828194
Overlap Similar Coefficient	0.3956044
Cosine Similarity Measure	0.3236029
Simple Matching Coefficient	0.1894737

From the above table, the similar values are – 0.1885, 0.3172, 0.3956, 0.3236 and 0.1894 given by Jaccard, Dice, Overlap, Cosine and Simple matching similarity respectively. Among them the similar values of Dice, Overlap and Cosine are greater than the threshold value ($\alpha \geq 0.3$) which proved that these two sentences are similar.

28. Interrogative Sentence - Let us consider another two interrogative sentences as example of English ⇆ French translation:

text1 – 'Who will send him this information?'
text2 – 'Qui lui enverra cette information?'

Implying the steps and the equations which are described above, we get the following result:

Table 47: Multilingualism of Interrogative Sentences

Jaccard Coefficient (Similarity)	0.270936
Jaccard Coefficient (Dis-similarity)	0.729064
Dice Similar Coefficient	0.4263566
Dice Dissimilar Coefficient	0.5736434
Overlap Similar Coefficient	0.5238096
Cosine Similarity Measure	0.4339327
Simple Matching Coefficient	0.270936

As we know these two sentences are similar because basically it is a translation from English to French. Now from the above table, the similar values are – 0.271, 0.426, 0.524, 0.434 and 0.271 given by Jaccard, Dice, Overlap, Cosine and simple matching coefficient respectively. The similar values of Jaccard and Simple matching coefficient is same i.e. – $0.271 \approx 0.3$ that means very much close to the threshold value ($\alpha \geq 0.3$). Besides, the similar values of Dice, Overlap and Cosine are greater than the threshold value ($\alpha \geq 0.3$). Among them the Overlap gives the highest similar value i.e. $0.524 > 0.3$. So, it is clear that these two sentences are similar.

29. Imperative Sentence – let us consider the translation of imperative sentences between English one i.e. text1 to French one i.e. text2 is:

text1 – 'Perfect ressemblance of a portrait.'
text2 – 'Parfaite ressemblance d'un portrait.'

Applying the steps and the seven equation, we get the following results:

Table 48: Multilingualism of Imperative Sentences

Jaccard Coefficient (Similarity)	0.3658537
Jaccard Coefficient (Dis-similarity)	0.6335078
Dice Similar Coefficient	0.5555556
Dice Dissimilar Coefficient	0.4642857
Overlap Similar Coefficient	0.5555556
Cosine Similarity Measure	0.5360563
Simple Matching Coefficient	0.3664922

From the table, the similar values are – 0.366, 0.556, 0.556, 0.536 and 0.366 given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient respectively and each value is greater than the threshold value (α ≥ 0.3). Among them the Dice and Overlap give the same similar value – 0.556 and this value along with Cosine coefficient (0.536) show the highest similarity. So, it is proved that these two sentences are similar.

30. Past Tense – let us consider other translation of past tense like:

text1 – 'The Information was bad.'
text2 – 'L'information était mauvaise.',

Applying the procedure of steps and the seven equations, the results are given below:

Table 49: Multilingualism of Past Tense

Jaccard Coefficient (Similarity)	0.2235772
Jaccard Coefficient (Dis-similarity)	0.7764227
Dice Similar Coefficient	0.3654485
Dice Dissimilar Coefficient	0.6345515
Overlap Similar Coefficient	0.6043956
Cosine Similarity Measure	0.3978619
Simple Matching Coefficient	0.2235772

From the table, the similar values are – 0.2235, 0.3654, 0.6044, 3979 and 0.2236 given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient respectively. Among them Overlap coefficient gives the highest similar value. Besides, the similar values of

Dice, Overlap and Cosine are greater than the threshold value (α ≥ 0.3) which indicate that these two sentences are similar.

31. **Future tense** – we can give an example like future tense of French and English translation which is given below:

text1 – 'He will go to a restaurant for soup and salad.'
text2 – 'Il ira à un restaurant soupe et salade.'

Applying the steps and the seven equations, we get following result:

Table 50: Multilingualism of Future Tense

Jaccard Coefficient (Similarity)	0.2439613
Jaccard Coefficient (Dis-similarity)	0.7560387
Dice Similar Coefficient	0.392233
Dice Dissimilar Coefficient	0.607767
Overlap Similar Coefficient	0.531579
Cosine Similarity Measure	0.406446
Simple Matching Coefficient	0.2439613

From the above table, the Jaccard and Simple Matching provide same similar value i.e. 0.2439613. Besides, the other similar values are – 0.392, 0.5316, 0.40645 given by Dice, Overlap and Cosine respectively and these values are greater than the threshold value (α ≥ 0.3) which proved the similarity between these two sentences.

32. **Conditional Sentence –** in this section we will show an example which are given below:

For example: now we will take a conditional sentence of French and its translation of English is given below:

text1 – 'Could you give some vinaigrette in the salad?' and
text2 – 'Pourriez-vous mettre de la vinaigrette dans la salade?'

From our steps and the seven equations, we get the following result:

Table 51: Multilingualism of Conditional Sentences

Jaccard Coefficient (Similarity)	0.2272727
Jaccard Coefficient (Dis-similarity)	0.772296
Dice Similar Coefficient	0.3703704
Dice Dissimilar Coefficient	0.6296296
Overlap Similar Coefficient	1
Cosine Similarity Measure	0.4767313

Simple Matching Coefficient	0.227704

From above table, we get same similar values i.e. 0.2272727, 0.227704 from Jaccard and Simple Matching coefficient respectively. Besides, the similar values of Dice and Cosine coefficients are – 0.3704 and 0.4767 respectively which are greater than the threshold value ($\alpha \geq 0.3$). Among them, the highest similar value is given by Overlap coefficient which is – 1 and it shows the 100% similarity between these two given sentences.

33. Negative Sentence – now we will take a sentence of negative and the translation of this sentence is like:

text1 – 'Dan was not amazed by the transformation of Linda.'
text 2 – 'Dan n'était pas émerveillé par la transformation de Linda.'

Now, implying the described steps and the equations over this given example, we get the following results:

Table 52: Multilingualism of Negative Sentences

Jaccard Coefficient (Similarity)	0.2403487
Jaccard Coefficient (Dis-similarity)	0.7596513
Dice Similar Coefficient	0.3875502
Dice Dissimilar Coefficient	0.6124498
Overlap Similar Coefficient	0.4436782
Cosine Similarity Measure	0.390689
Simple Matching Coefficient	0.2403487

The similar value of Jaccard and Simple matching coefficient is same i.e. 0.1403487. The highest similar value is given by Overlap similar coefficient i.e. 0.44368 which is greater than the threshold value ($\alpha \geq 0.3$). Besides, the similar values of Dice and Cosine are – 0.38755 and 0.39069 respectively which are also greater than the threshold value. These values proved that these two sentences are similar.

3.5 Limitations

As we know, no one's work is beyond limitations as like ours. Our knowledge base is built on uncovering each piece of the puzzle, one at a time and limitations shows us where new efforts need to be made. We get limitations only for some grammatical rules which are described below:

- **Passive Sentence** – the subject in a passive sentence does not perform the action in the sentence. In fact, the action is performed on it.

  For example1 – let us consider first example for passive sentence:

  text1 = 'I am eating rice.'
  text2 = 'Rice is being eaten by me.'

  Now, removing all the stop-words, punctuations and all. Simultaneously, converting the whole string into lowercase like the following way:

  filteredText1 = 'eating rice'
  filteredText2 = 'rice eaten'

  Generating list of tri-grams using both filtered text files as follows:

  list1 = {'eat', 'ati', 'tin', 'ing', 'ng ', 'g r', ' ri', 'ric', 'ice'}
  list2 = {'ric', 'ice', 'ce ', 'e e', ' ea', 'eat', 'ate', 'ten'}

  Now, creating the matrices using mentioned lists as well as eliminating negative values which are given below:

Table 53: Matrix1 which is generated from list1

	'eat'	'ati'	'tin'	'ing'	'ng '	'g r'	' ri'	ric	ice
'eat'	0	1	2	3	4	5	6	7	8
'ati'		0	1	2	3	4	5	6	7
'tin'			0	1	2	3	4	5	6
'ing'				0	1	2	3	4	5
'ng '					0	1	2	3	4
'g r'						0	1	2	3
' ri'							0	1	2
ric								0	1
ice									0

Table 54: Matrix2 which is generated from list2

	'ric'	'ice'	'ce '	'e e'	' ea'	'eat'	'ate'	'ten'
'ric'	0	1	2	3	4	5	6	7
'ice'		0	1	2	3	4	5	6
'ce '			0	1	2	3	4	5
'e e'				0	1	2	3	4
' ea'					0	1	2	3
'eat'						0	1	2
'ate'							0	1
'ten'								0

Using the seven noted equations of similarity measure coefficients and seven the equations, we get following similarities as well as dissimilarities values:

Table 55: Similarity and Dissimilarity Measures of Passive Sentences (ex1)

Coefficient	Values
Jaccard Similarity Coefficient	0.05194805
Jaccard Dissimilarity Coefficient	0.9480519
Dice Similar Coefficient	0.09876543
Dice Dissimilarity Coefficient	0.9012346
Overlap Similar Coefficient	0.1111111
Cosine Similarity Measure	0.09938081
Simple Matching Coefficient	0.05194805

As per our opinion, these two sentences are similar. But from the table, the similar values are – 0.052, 0.099, 0.111, 0.0994 and 0.052 given by Jaccard, Dice, Overlap, Cosine and Simple matching coefficient which are very low regarding our threshold value ($\alpha \geq 0.3$) and these values indicate the limitation of our work. But without using any help of online dictionaries as well as web-pages, these similar values would be very convincing as our opinion.

For example2 – let us consider the two texts:

text1 – 'Carl sounded the alarm due to the panic.' and
text2 – 'The alarm was sounded by Carl due to the panic.'

Result: Using the steps and the equations – (i), (ii), (iii), (iv), (v), (vi) and (vii) which are described above, we get following result:

Table 56: Similarity Measure of Passive Sentences (ex2)

Co-occurrence Measures	Values
Jaccard Coefficient (Similarity)	0.119469
Jaccard Coefficient (Dis-similarity)	0.880531
Dice Similar Coefficient	0.2134387
Dice Dissimilar Coefficient	0.7865613
Overlap Similar Coefficient	0.2134387
Cosine Similarity Measure	0.2134387
Simple Matching Coefficient	0.119469

As per our opinion these two sentences are similar. Here, the Jaccard and the Simple Matching give the same similar value i.e. 0.119469 and the Dice and the Cosine give the same similar value i.e. 0.2134387. But these similar coefficients are lower than our threshold value ($\alpha \geq 0.3$). These types of examples would be a limitation of our method. Without using any helping things like online dictionary, search engine and so on, these similar values would be very convincing according to our perception.

- **Infinitive Sentence** – now we will see an example of infinitive sentence e.g.

text1 – 'She loves to sing.'
text2 – 'Elle aime chanter.'

From our described steps and equations, the results are given below:

Table 57: Multilingualism of Infinitive Sentence

Jaccard Coefficient (Similarity)	0
Jaccard Coefficient (Dis-similarity)	1
Dice similar Coefficient	0
Dice Dissimilar Coefficient	1
Overlap Coefficient	0
Cosine Similarity Measure	0
Simple Matching Coefficient	0

Though these two sentences are like each other as per our opinion, but here we get dissimilarity between them which would be a limitation of our method for this grammatical rule.

Overall, we have seen so many examples using several types of grammatical rules for English and French respectively which give clear view that our method is not dependent on any grammatical rules. Simultaneously, we have seen many examples using the translations of English to French sentences which gives an idea of multilingualism of our method. Though we found limitations only for few examples, but most other examples give very satisfactory results without using any helping items like online dictionaries, web-pages etc. Besides, it is noticeable that our method is learning by itself. So, it is demanded that the main idea our method is best which would be very beneficial for further use in the field of artificial intelligence.

In the following chapter, we will discuss about our implementation and algorithm as well as we will give some user interfaces of our method. Afterwards, we demonstrate comparisons of our methodology with others

CHAPTER IV

IMPLEMENTATION

The whole theory presented here was implemented by C#. The following algorithm contributes a clear concept about our design:

4.1 Algorithm

```
1. inputFile = Two large data-sets from different sources of large text file i.e. pdf, doc,
   docx, pptx, ppt or web-pages.
2. nGramSize = must be non-negative integer value which is $0 < nGramSize \leq 10$.
//calculating n-grams of characters of both text separately using nGramsize.
3. while (text.line != null) do
     a. filteredData = remove stop-words, punctuation, delimiters from each line of the
        text and convert them to lowercase.
     b. for i := 0 to (filteredData.Length - nGramSize + 1) do
          i. list.Add(filteredData.Substring(i, nGramSize))
     c. end for
4. end while
5. CreateDistanceMatrix(X)
6. K ← length(X)
7. For i ← 1 to K do
8.   For j ← 1 to K do
9.       R ← X[14]
10.      C ← X[i]
11.      V ← i – j
```

```
12.         Matrix.Add(Row = R, Column = C, Value = V
13. JaccardSimilarity(Matrix1, Matrix2) = $\frac{|Matrix_1 \cap Matrix_2|}{|Matrix_1| + |Matrix_2| - |Matrix_1 \cap Matrix_2|}$
14. JaccardDissimilarity(Matrix1, Matrix2) = $\frac{|\mathrm{Matrix}_1 \cup \mathrm{Matrix}_2| - |Matrix_1 \cap Matrix_2|}{|\mathrm{Matrix}_1 \cup \mathrm{Matrix}_2|}$
15. DiceSimilarity(Matrix1, Matrix2) = $\frac{2 \times |Matrix_1 \cap Matrix_2|}{|Matrix_1| + |Matrix_2|}$
16. DiceDissimilarity(Matrix1, Matrix2) = $1 - \frac{2 \times |Matrix_1 \cap Matrix_2|}{|Matrix_1| + |Matrix_2|}$
17. OverlapSimilarity(Matrix1, Matrix2) = $\frac{|Matrix_1 \cap Matrix_2|}{\min(|Matrix_1|,\ |Matrix_2|)}$
18. CosineSimilarity(Matrix1, Matrix2) = $\frac{|Matrix_1 \cap Matrix_2|}{sqrt(|Matrix_1|) \times sqrt(|Matrix_2|)}$
19. SimpleMatching(Matrix1, Matrix2) = $\frac{|Matrix_1 \cap Matrix_2|}{|\mathrm{Matrix}_1 \cup \mathrm{Matrix}_2|}$
20. output: Five different similarity coefficients (Jaccard, Dice, Overlap, Cosine and
    Simple Matching) as well as two dissimilarity coefficients (Jaccard and Dice) in the
    scale of (0, 1).
```

According to the algorithm, the *inputFile* will collect two large text files from different kind of sources like pdf, doc, docx, pptx, ppt or different web-pages etc. Here, the *nGramSize* is an integer non-negative value which defines the size of *n-grams of characters* and it must be $0 < nGramSize \leq 10$. For example – the size 1 of *character n-gram (n-gram of character)* is referred to as a *'unigram of character'*, size 2 is a '*bigram of character*', size 3 is a '*trigrams of character*' etc. Using *trigrams of characters*, we got a satisfying result. In the *while* loop, it filters stop words, punctuation and delimiters from those raw texts and splits each line of those texts separately based on the given *n-gram* size until it gets null value. In this way, it produces *tri-grams of characters* of both given texts separately. As we mentioned earlier that, stop-words, punctuations and delimiters are used commonly over the text and by removing those we can focus on the important text which will save more process time and space of memory. Now, using *CreateDistanceMatrix()* function, it generates two distance matrices of two list of *n-grams of characters* separately named as – *Matrix1, Matrix2*. Then, using those two distance matrices, it calculates the coefficients of similarity measures – Jaccard, Dice, Overlap, Cosine, Simple Matching and dissimilarity measures – Jaccard and Dice. Thus, we get the similarity and dissimilarity coefficients to measure the similarity as well as dissimilarity more preciously. To better understand the measure of similarity and dissimilarity, we also have a scale of *(0, 1)*. Here, the scale *(0, 1)* will symbolize the intense similarity or dissimilarity between given two files.

4.2 Time Complexity

The time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input [35], [38]. The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, where an elementary operation takes a fixed amount of time to perform. Thus, the amount of time taken and the number of elementary operations performed by the algorithm differ by at most a constant factor. Deliberating the complexity of the above algorithm, we deduce that the overall algorithm complexity is quadratic time denoted by $\boldsymbol{O(N^2)}$ which represents that the performance of an algorithm is directly proportional to the square of the size of the input data set. That means, our algorithm performs more efficient for detecting similarities between large texts.

4.3 Interfaces of Our Methodolgoy

We have some user interfaces of our method which is developed by C#. For now, our application is windows based and we will prepare it more versatile afterwards. The following snaps will provide a clear idea about our user interface as well as our methodology:

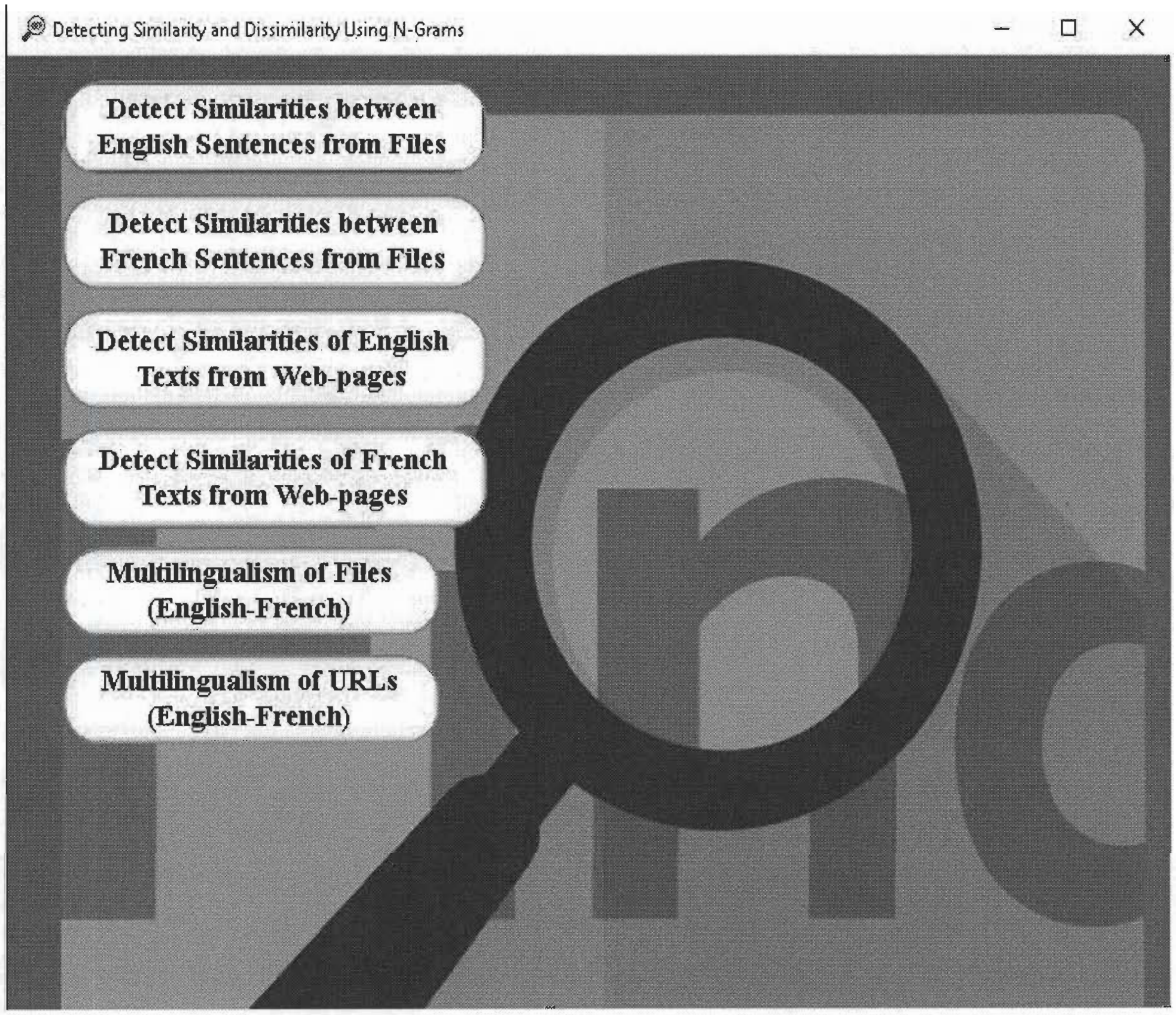
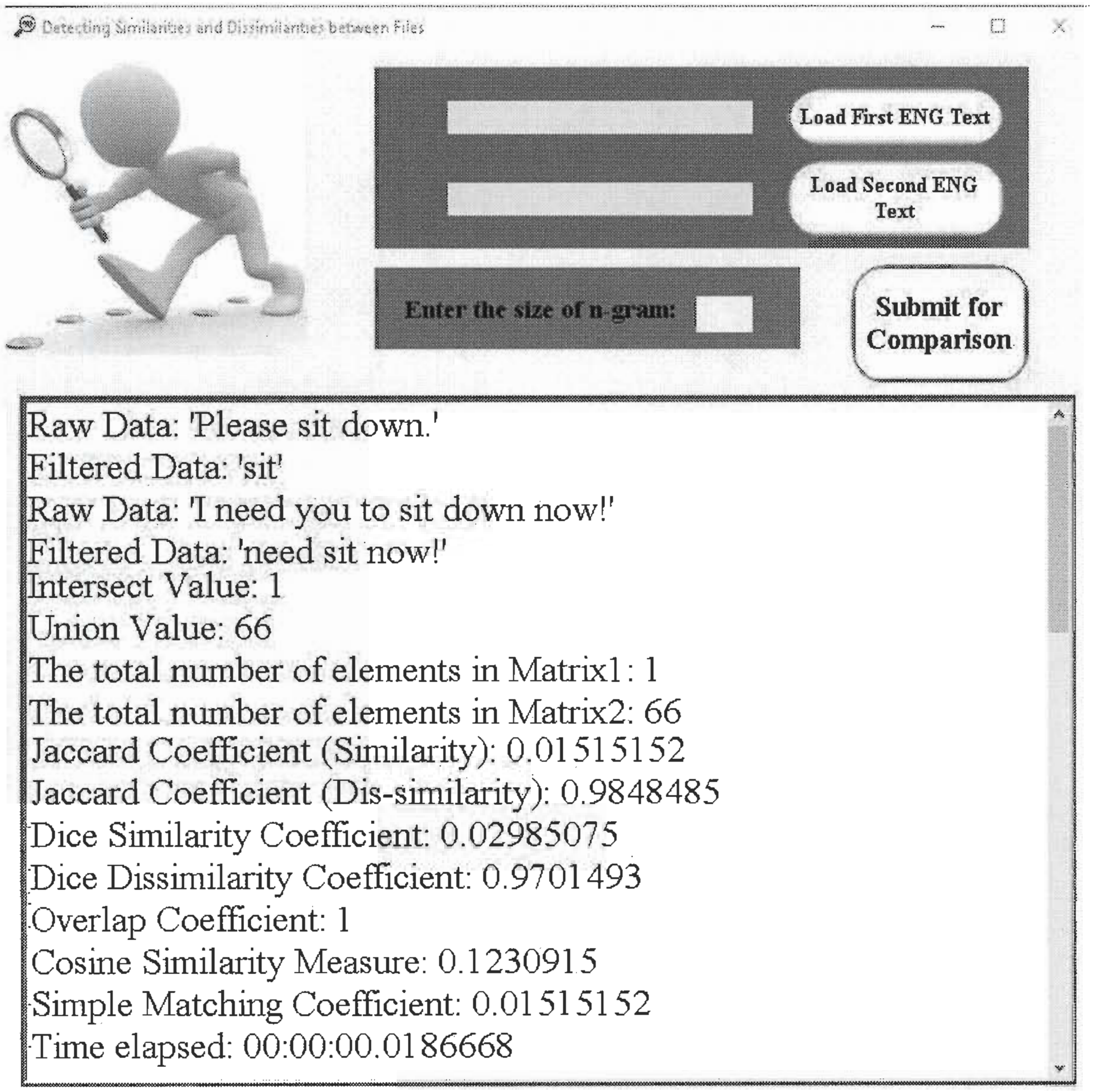
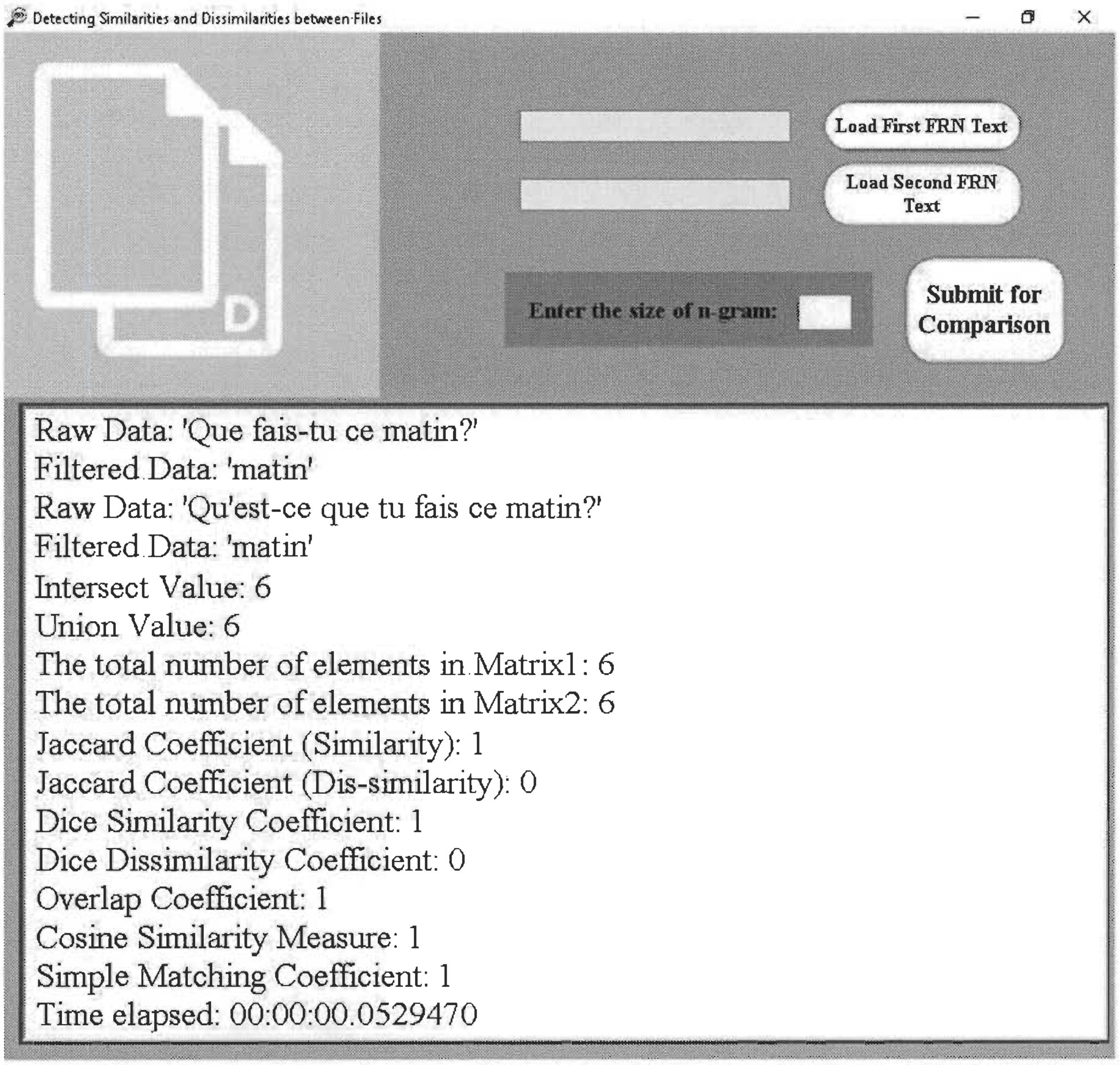

Detecting Similarity and Dissimilarity Using N-Grams
Detect Similarities between English Sentences from Files
Detect Similarities between French Sentences from Files
Detect Similarities of English Texts from Web-pages
Detect Similarities of French Texts from Web-pages
Multilingualism of Files (English-French)
Multilingualism of URLs (English-French)

Figure 4: Interface – Home Page


Detecting Similarities and Dissimilarities between Files
Load First ENG Text
Load Second ENG Text
Enter the size of n-gram:
Submit for Comparison

```
Raw Data: 'Please sit down.'
Filtered Data: 'sit'
Raw Data: 'I need you to sit down now!'
Filtered Data: 'need sit now!'
Intersect Value: 1
Union Value: 66
The total number of elements in Matrix1: 1
The total number of elements in Matrix2: 66
Jaccard Coefficient (Similarity): 0.01515152
Jaccard Coefficient (Dis-similarity): 0.9848485
Dice Similarity Coefficient: 0.02985075
Dice Dissimilarity Coefficient: 0.9701493
Overlap Coefficient: 1
Cosine Similarity Measure: 0.1230915
Simple Matching Coefficient: 0.01515152
Time elapsed: 00:00:00.0186668
```

Figure 5: Interface – Detecting Similarities between Two English Contained Files


Detecting Similarities and Dissimilarities between Files
Load First FRN Text
Load Second FRN Text
Enter the size of n-gram:
Submit for Comparison

```
Raw Data: 'Que fais-tu ce matin?'
Filtered Data: 'matin'
Raw Data: 'Qu'est-ce que tu fais ce matin?'
Filtered Data: 'matin'
Intersect Value: 6
Union Value: 6
The total number of elements in Matrix1: 6
The total number of elements in Matrix2: 6
Jaccard Coefficient (Similarity): 1
Jaccard Coefficient (Dis-similarity): 0
Dice Similarity Coefficient: 1
Dice Dissimilarity Coefficient: 0
Overlap Coefficient: 1
Cosine Similarity Measure: 1
Simple Matching Coefficient: 1
Time elapsed: 00:00:00.0529470
```

Figure 6: Interface – Detecting Similarities between Two French Contained Files

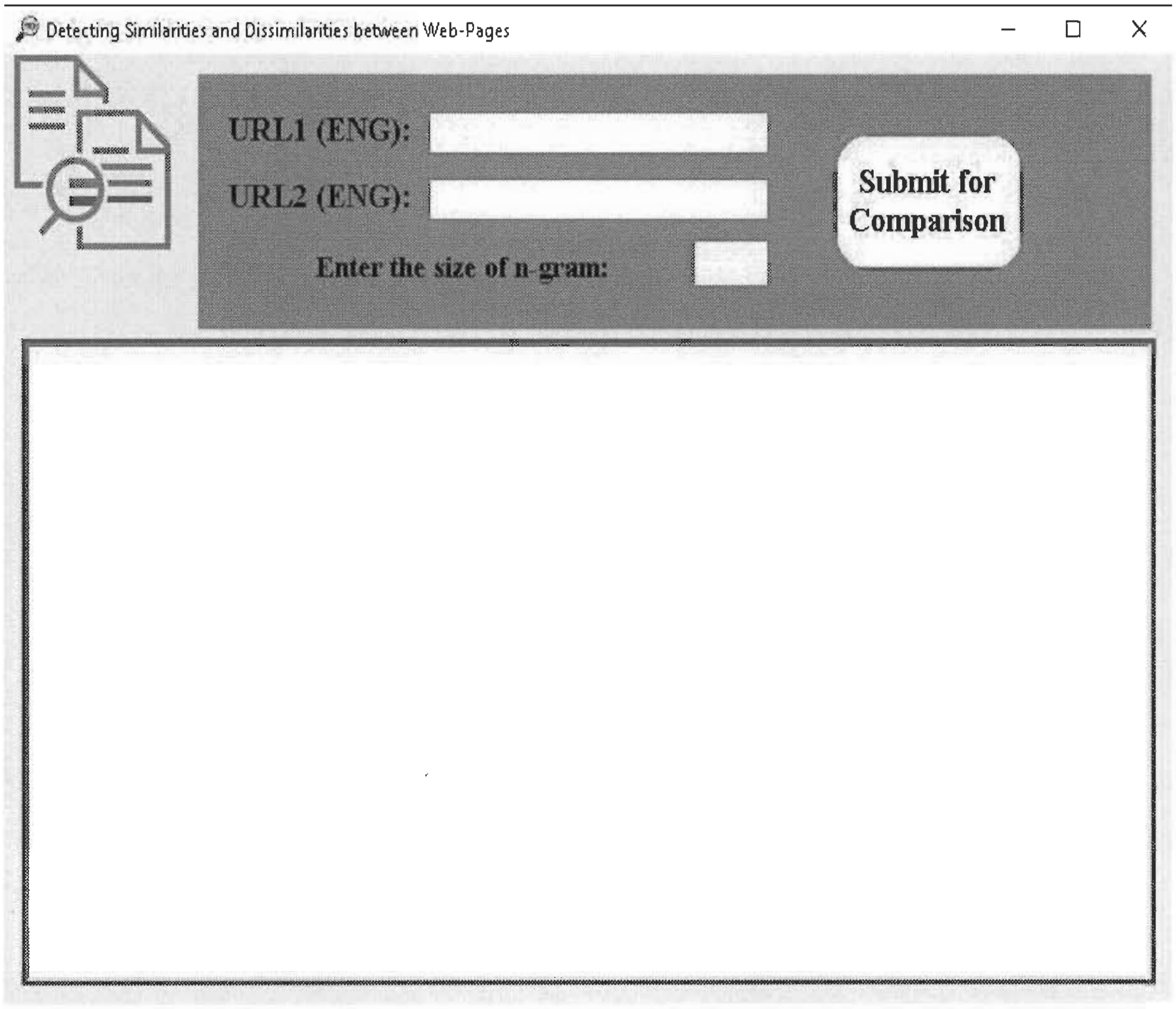

Detecting Similarities and Dissimilarities between Web-Pages
URL1 (ENG):
URL2 (ENG):
Enter the size of n-gram:
Submit for Comparison

Figure 7: Interface – Detecting Similarities between Two English Web-Pages

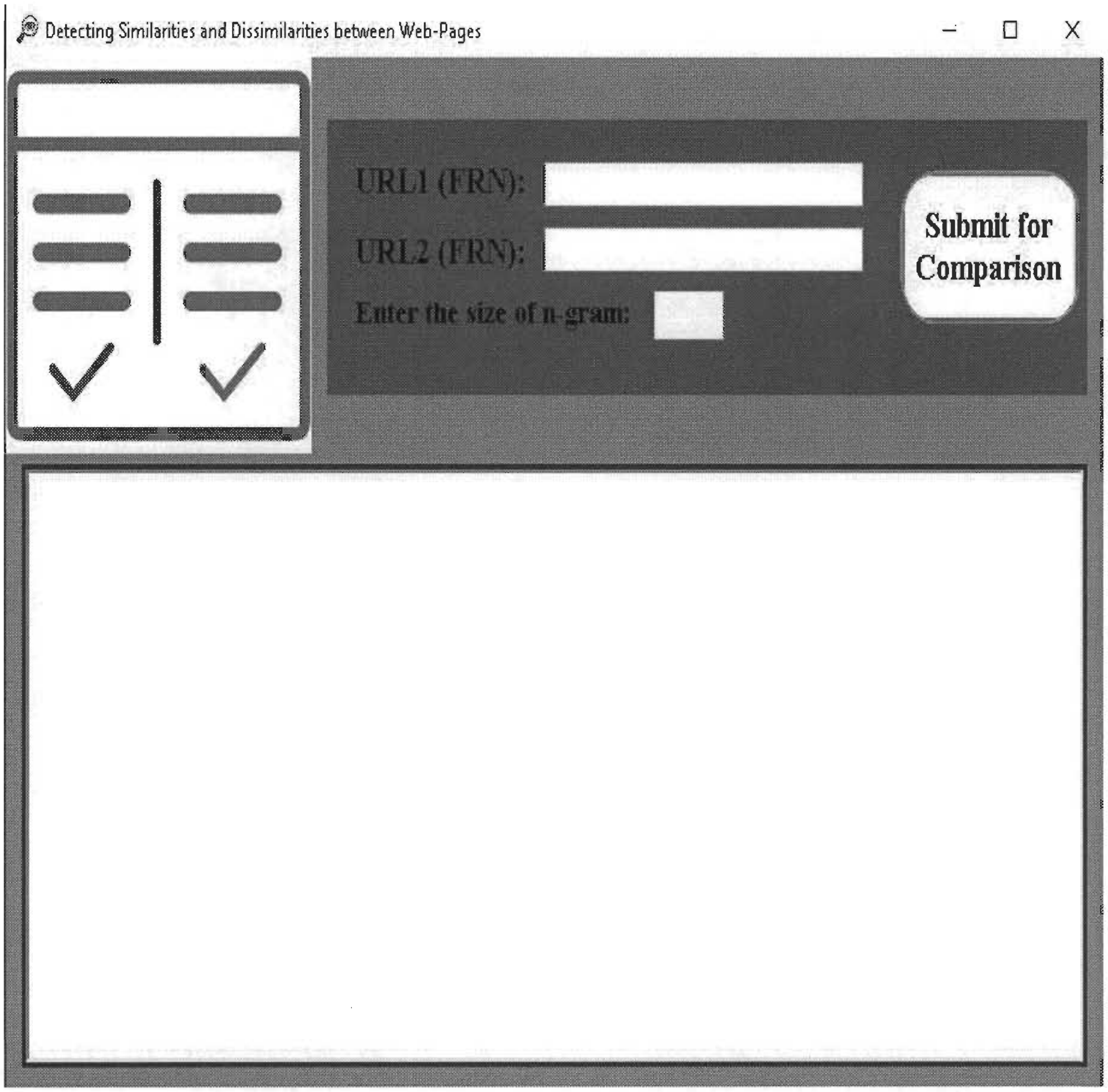
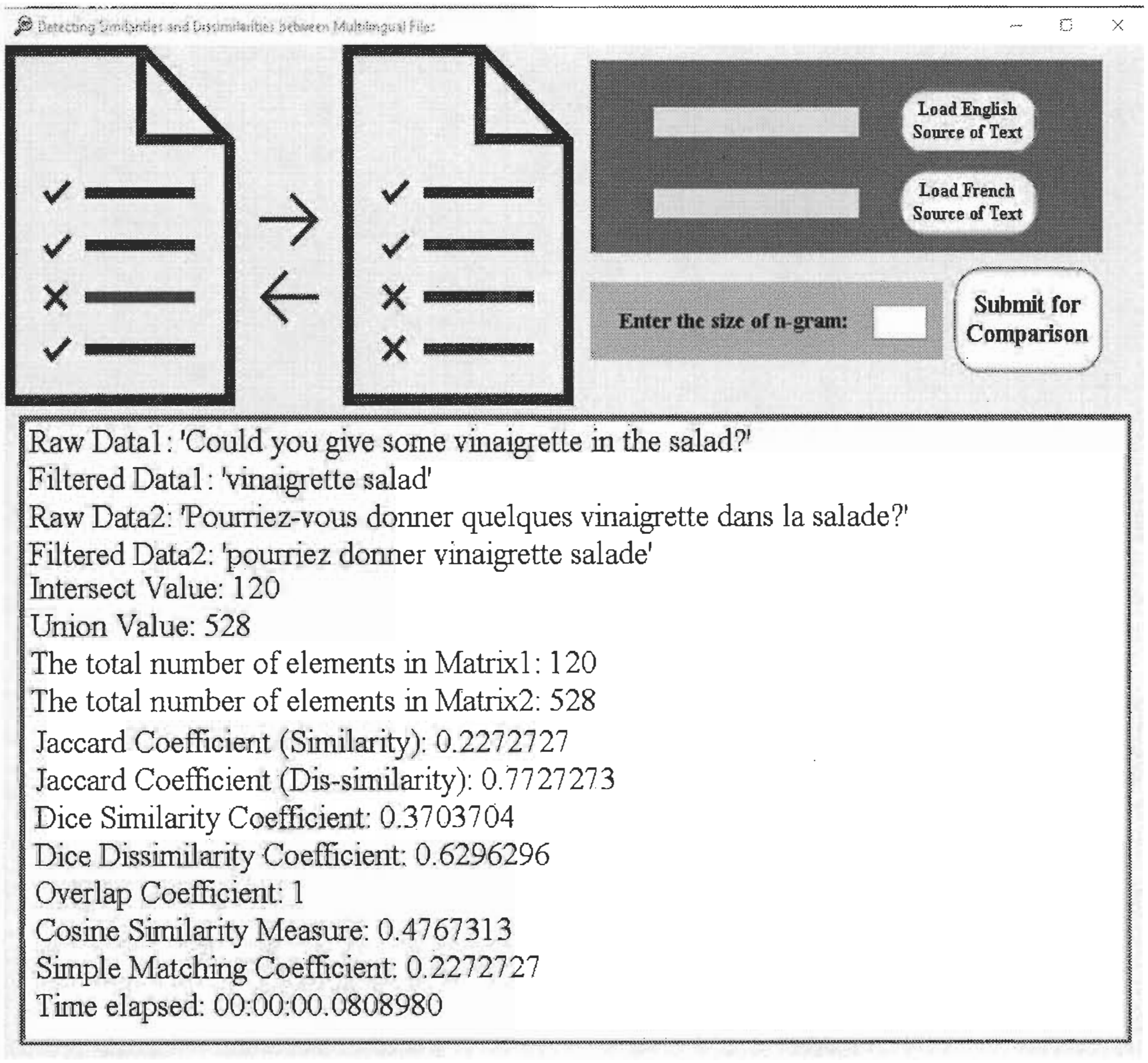

Detecting Similarities and Dissimilarities between Web-Pages
URL1 (FRN):
URL2 (FRN):
Enter the size of n-gram:
Submit for Comparison

Figure 8: Interface – Detecting Similarities between Two French Web-Pages


Detecting Similarities and Dissimilarities between Multilingual Files
Load English Source of Text
Load French Source of Text
Enter the size of n-gram:
Submit for Comparison

```
Raw Data1: 'Could you give some vinaigrette in the salad?'
Filtered Data1: 'vinaigrette salad'
Raw Data2: 'Pourriez-vous donner quelques vinaigrette dans la salade?'
Filtered Data2: 'pourriez donner vinaigrette salade'
Intersect Value: 120
Union Value: 528
The total number of elements in Matrix1: 120
The total number of elements in Matrix2: 528
Jaccard Coefficient (Similarity): 0.2272727
Jaccard Coefficient (Dis-similarity): 0.7727273
Dice Similarity Coefficient: 0.3703704
Dice Dissimilarity Coefficient: 0.6296296
Overlap Coefficient: 1
Cosine Similarity Measure: 0.4767313
Simple Matching Coefficient: 0.2272727
Time elapsed: 00:00:00.0808980
```

Figure 9: Interface – Detecting Similarities between Two Multilingual Files

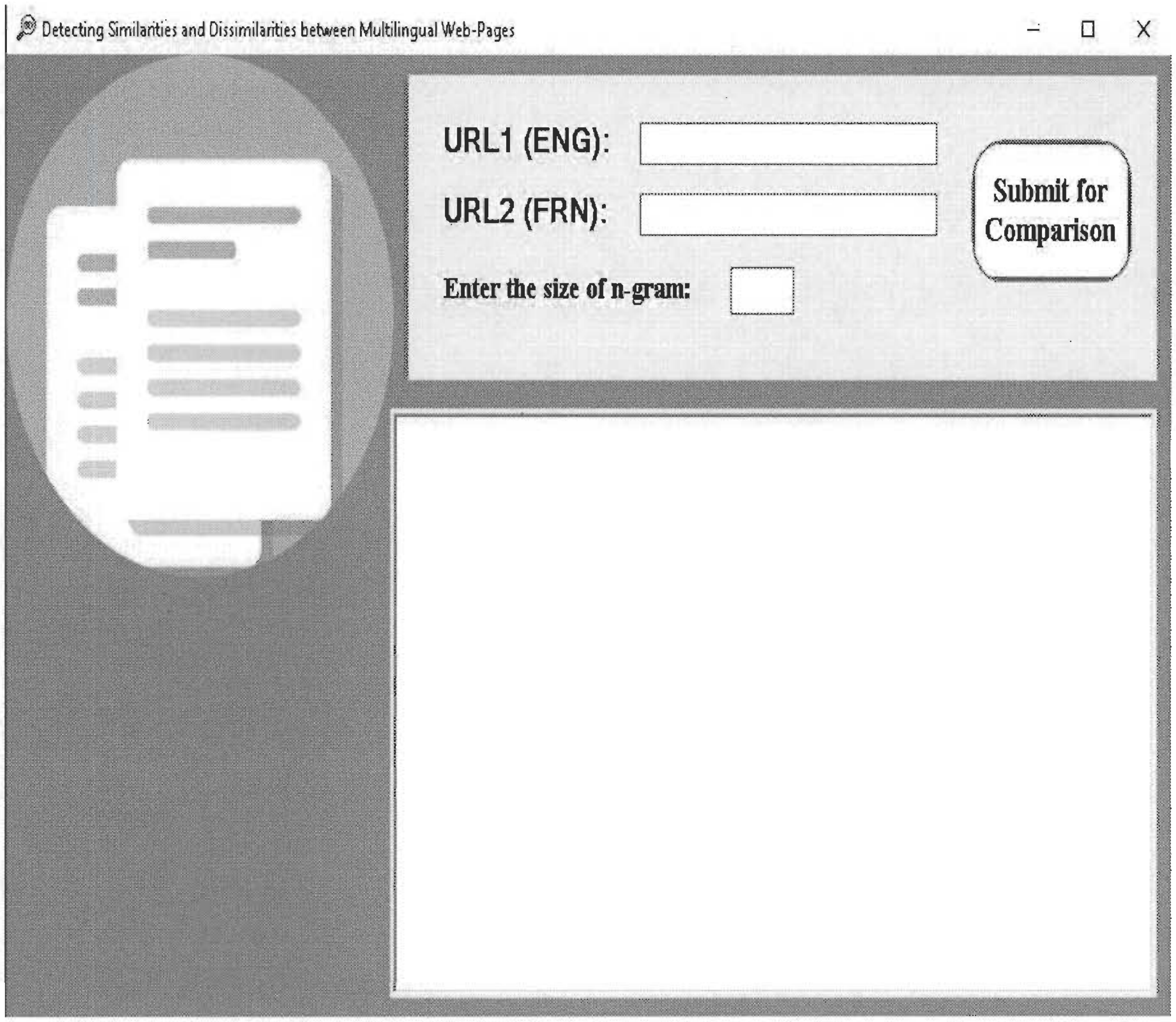

Detecting Similarities and Dissimilarities between Multilingual Web-Pages
URL1 (ENG):
URL2 (FRN):
Enter the size of n-gram:
Submit for Comparison

Figure 10: Interface – Detecting Similarities between Two Multilingual Web-Pages

CHAPTER V

EXPERIMENTATION

In this section, we categories the text probabilities getting from different web-pages. Using four various categorized sentences like 'positive similarities when the similarity of the texts should be positive', 'positive similarities when the similarity of texts should be negative', 'negative similarities when the texts should be positive', and 'negative similarities when the similarity of texts should be negative', we will illustrate the comparison between our methodology and the methodology of most recent journal of Akermi and Faiz [2]. The comparison of our method with some classified examples are given below:

Table 58: Comparison between Our Methodology to Others

Positive similarities when the similarity of the texts should be positive				
	Example	**Akermi and Faiz's Algorithm**	**Our Method**	
			Coefficients	**Values**
Text 1	I am Saima Sultana. I live in Canada.	unable to identify name of person & also place by both dictionary and web search engine	Similarities	
			Jaccard Coeff.	0.2488688
			Dice Coeff.	0.3985507
			Overlap Coeff.	0.3985507
			Cosine Sim.	0.3985507
			Simple M.	0.2505695
Text 2	I live in Canada. I am Saima Sultana.		Dissimilarities	
			Jaccard	0.7494305
			Dice	0.6014493
Positive similarities when the similarity of the texts should be negative				
	Example	**Akermi and Faiz's Algorithm**	**Our Method**	
			Coefficients	**Values**
Text 1	She is a good girl.	Though these sentences have negative meaning but this algo. shows 1 score	Similarities	
			Jaccard Coeff.	0.2564103
			Dice Coeff.	0.4081633

		[scale (0, 1)]	Overlap Coeff.	0.4761905
			Cosine Sim.	0.4123931
			Simple M.	0.2564103
Text 2	She is a bad girl.		Dissimilarities	
			Jaccard	0.7435898
			Dice	0.5918368

Negative similarities when the similarities of the texts should be positive

	Example	**Akermi and Faiz's Algorithm**	**Our Method**	
			Coefficients	**Values**
Text 1	I didn't say anything.	This algo. gives 0.4173559 similarities score. In addition, their used dictionary is unable to detect the word "didn't".	Similarities	
			Jaccard Coeff.	0
			Dice Coeff.	0
			Overlap Coeff.	0
			Cosine Sim.	0
			Simple M.	0
Text 2	I said nothing.		Dissimilarities	
			Jaccard	1
				1

Negative similarities when the similarities of the texts should be negative

	Example	**Akermi and Faiz's Algorithm**	**Our Method**	
			Coefficients	**Values**
Text 1	I am a master's student in UQTR.	Their algo. gives 0.2 similarity score. Besides Their used dictionary and also search engine (Digg.com) is unable to detect the university name "UQTR" and also the word "master's".	Similarities	
			Jaccard Coeff.	0.006479482
			Dice Coeff.	0.01287554
			Overlap Coeff.	0.01578947
			Cosine Sim.	0.01310056
			Simple M.	0.006479482
Text 2	UQTR is a beautiful university.		Dissimilarities	
			Jaccard	0.9935205
			Dice	0.9871244

We will discuss about each section of the earlier table. For the first section entitled 'Positive similarities when the similarities of the texts should be positive', we took two similar texts and the meaning of these texts is positive. Here, each text contains two sentences, but in different order and the meaning of these two texts is positive as well. Using our method, the Dice, Overlap and Cosine provide the same similar value i.e. 0.3985507 which is greater than the threshold value ($\alpha \geq 0.3$). So, according to our method we can say that these two sentences are similar. On the other hand, Akermi and Faiz's algorithm [2] is unable to identify name of person ('Saima Sultana') and also place ('Canada') by both dictionary and web search engine. So, we can say that their method was unable to detect similarity between these two sentences.

For the second section entitled 'Positive similarities when the similarities of the texts should be negative', we took two similar texts but the meaning of these texts is negative. Using our method, the similar values of Dice, Overlap and Cosine are – 0.4081633, 0.4761905 and 0.4123931 respectively and these values are greater than the threshold value ($\alpha \geq 0.3$). Besides, we got same similar value from Jaccard and Simple matching which is – 0.2564103 and this value is lower than the threshold value ($\alpha \geq 0.3$). So, only because of 'girl' we get some similar values from Dice, Overlap and Cosine and we get an idea how much these two sentences are similar in the scale of (0, 1) while the meaning of these two sentences is different. On the other hand, the lower similar value from Jaccard and Simple matching shows how much these two sentences have negative meaning. On the other hand, Akermi and Faiz's algorithm [2] shows similar value is 1 that means it shows 100% similarity between these two sentences while the meaning of these two sentences is totally different.

For the third section entitled 'Negative similarities when the similarities of the texts should be positive', we took two different texts but the meaning of these texts is positive. Using our method, we get the similarity value is – 0 that means it shows these two sentences are totally dissimilar and we consider it would be a limitation of our methodology. On the other hand, the similar value from Akermi and Faiz's algorithm [2] is – 0.4173559 of these two sentences.

For the fourth section entitled 'Negative similarities when the similarities of the texts should be negative', we took two different texts with the negative meaning. From our method, we found the Jaccard and Simple matching give the same similar value i.e. 0.006479482. Besides, the other similar values are – 0.0129, 0.0158 and 0.0131 given by Dice, Overlap and Cosine respectively. These values are lower than the threshold value ($\alpha \geq 0.3$) which signify the dissimilarity between these two sentences. On the other hand, the similarity score from Akermi and Faiz's algorithm [2] is – 0.2 which is greater than the similar values of our method as well as their algorithm is unable to detect the university name "UQTR" and also the word "master's".

We have used around 150 sentences in total for experimenting the methodology and each time selected the *n-gram* size is 3 as the parameter for *nGramSize*. For experimenting, we use plain texts as input file but our real application can also compile pdf, doc, docx, ppt, pptx, plain text, web-pages and so on.

Though the opinion may vary person to person, but now according to our opinion, we will try to find out the accuracy of our results based on the actual relationship between our experimented texts separately. For this, we have four parameters – True Similar (TS), False Similar (FS), True Dissimilar (TD), False Dissimilar (FD) to find out the actual relationship between two texts. According to our concept, we will find out the relationship between given texts and then based on the result of our method we will find out the relationship between those texts. Here, we will find accuracy of the grammatical part of both English and French as well as the multilingualism of English to French respectively. Here, we will take only the similar scores which satisfy our expectations while comparing those values with the threshold value ($\alpha \geq 0.3$) and then we will able to find out the relationship between the given texts according to our method. Now we will shortly demonstrate that how we will get TS, TD, FS and FD from the results of our methodology –

i. **True Similarity (TS)**– let us consider the original two texts are true similar (TS). If one or more than one results of our five different coefficients are greater than or equal to the threshold value (0.3), then we will consider those sentences as True Similar (TS) according to the results of our method.

ii. **True Dissimilar (TD)** – let us consider, the original two texts are true dissimilar (TD). If more than one results of our five different coefficients are less than the threshold value (0.3), then we will consider those sentences as True Dissimilar (TD) according to the results of our method.

iii. **False Dissimilar (FD)** – let us consider, the original two texts are true similar (TS). If more than one results of our five different coefficients are less than the threshold value (0.3), then we will consider those sentences as False Dissimilar (FD) according to the results of our method.

iv. **False Similar (FS) –** let us consider, the original two texts are true dissimilar (TD). If more than one results of our five different coefficients are greater than or equal to the threshold value (0.3), then we will consider those sentences as False Similar (FS) according to the results of our method.

Lastly, we will count them and able to find out the accuracy of our method in the following way –

Table 59: Accuracy Measuring of Our Methodology using English Sentences

	English Sentences			**Our Method**				
		True Similar	**True Dissimilar**	**Results (Similar values)**	**True Similar**	**False Similar**	**True Dissimilar**	**False Dissimilar**
Txt.1	Computing	TS		J.– 0.62 D.– 0.77 O. – 1 C.– 0.77 S. – 0.62 > 0.3	TS			
Txt. 2	Recomputing							
Txt. 1	Karen rode on a yellow skate board.		TD	J.–0.052 D.–0.09 O. –0.12 C. – 0.1 S. – 0.05 < 0.3			TD	
Txt.2	Karen lives in the yellow house.							
Txt. 1	I want to be a good writer.		TD	J. & S. – 0.1687 D.– 0.2887 < 0.3			TD	
Txt. 2	My friend is a really good writer.							
Txt.1	Please sit down.	TS		O. – 1 > 0.3	TS			
Txt.2	I need you to sit down now!							
Txt.1	When are you going to turn in your writing assignment?		TD	J., D., O., C. & S. – 0			TD	

Txt.2	Do you know what the weather will be tomorrow?			< 0.3				
Txt.1	It is too dangerous to climb that mountain!		TD	J., D., O., C. & S. – 0 < 0.3			TD	
Txt.2	I got an A on my book report!							
Txt.1	Tom reads novels.	TS		D.– 0.34 O.– 0.54 C.– 0.36 > 0.3	TS			
Txt.2	Tom reads and enjoys novels.							
Txt.1	Tom reads novels, but Jack reads comics.	TS		J. – 0.94 D., O. & C. – 0.9716 S. – 1 > 0.3	TS			
Txt.2	Tom reads novels; however, Jack reads comics.							
Txt.1	Although Tom reads novels, Jack reads comics.	TS		D., O. & C. – 0.39773 > 0.3	TS			
Txt.2	Jack reads comics although Tom reads novels.							
Txt.1	Jack Smith, who reads comics, rarely reads novels.	TS		J. – 0.58 D. –0.74 O. –0.98 C. –0.76 S. – 0.597 > 0.3	TS			
Txt.2	People like Jack Smith, who reads comics rarely reads novels.							
Txt.1	While Tom	TS		J. – 0.43	TS			

	reads novels, Jack reads comics, but Sam reads only magazines.			D. – 0.6 O. –0.97 C. –0.65 S. – 0.44 > 0.3				
Txt.2	Tom reads novels, but Jack reads comics because books are too difficult.							
Txt.1	Jack, who reads comics, rarely reads novels; however, Tom enjoys novels.	TS		D. –0.38 O. –0.45 C. – 0.3833 > 0.3	TS			
Txt.2	People like Jack, who reads comics rarely reads novels; they often find books difficult.							
Txt.1	Moe played the tune more lyrically than Sam.	TS		J. & S. – 0.44872 D. –0.62 O. – 0.7 C. –0.62 > 0.3	TS			
Txt.2	Jack played the tune most lyrically.							
Txt.1	The early train arrives at 8.45 a.m.		TD	J. & S. – 0.1583 D., O. & C. – 0.27333 < 0.3			TD	
Txt.2	The 8.45 a.m. train arrived early.							
Txt.1	15 Sentences of 'Apple Inc.'		TD	J. & S. – 0.00079 D. – 0.0015 O. –			TD	
Txt.2	15 Sentences of 'Apple'							

				0.0026 C. – null < 0.3				
Txt.1	I am eating rice.	TS		J. & S. – 0.05195 D. –0.09 O. –0.11 C. –0.09 < 0.3				FD
Txt.2	Rice is being eaten by me.							
Txt.1	Carl sounded the alarm due to the panic	TS		J. & S. – 0.1195 D., O. & C. – 0.21344 < 0.3				FD
Txt.2	The alarm was sounded by Carl due to the panic							
Txt.1	I am Saima Sultana. I live in Canada.	TS		D., O. & C. – 0.3985 > 0.3	TS			
Txt.2	I live in Canada. I am Saima Sultana.							
Txt.1	She is a good girl.		TD	D. –0.41 O. –0.48 C. – 0.4124 > 0.3		FS		
Txt.2	She is a bad girl.							
Txt.1	I didn't say anything.	TS		J., D., O., C. & S. – 0 < 0.3				FD
Txt.2	I said nothing.							
Txt.1	I am a master's student in UQTR.		TD	J. & S. – 0.006 D. –0.01 O. – 0.016 C. –0.01 < 0.3			TD	
Txt.2	UQTR is a beautiful university.							

Now, we will calculate the accuracy of French sentences in the equivalent way which is given below:

Table 60: Accuracy Measuring of Our Methodology using French Sentences

	French Sentences			Our Method				
		True Similar	**True Dissimilar**	**Results (Similar values)**	**True Similar**	**False Similar**	**True Dissimilar**	**False Dissimilar**
Txt.1	J'ai ce livre.		TD	J.& S.– 0.11428 D.– 0.2 C.– 0.24 < 0.3			TD	
Txt. 2	J'ai acheté ce livre.							
Txt. 1	Je suis à la maison.		TD	J., D, O., C. & S. – 0< 0.3			TD	
Txt.2	Je suis allé au restaurant.							
Txt. 1	Le garçon a mangé la pomme.	TS		D.– 0.429 O. –0.46 C. –0.43 > 0.3	TS			
Txt. 2	La fille a mangé la pomme.							
Txt.1	Elle chantait quand je suis arrivé.		TD	J. & S. – 0.0028 D. – 0.005 O. – 0.022 C. – 0.00835 < 0.3			TD	
Txt.2	Quand j'étais enfant je venais souvent à cet endroit.							
Txt.1	Elle m'a dit qu'il ne l'avait jamais vu.	TS		J. & S. – 0.371 D., O. & C. – 0.54131 > 0.3	TS			
Txt.2	Elle me raconta qu'il ne l'avait jamais vu.							
Txt.1	La semaine prochaine nous partirons en Grèce pour les	TS		D. – 0.297 ≈ 0.3	TS			

	vacances de Pâques.			O. –0.63 C. –0.35 > 0.3				
Txt.2	Nous allons partir en Grèce pour les vacances de Pâques.							
Txt.1	Je commencerai à travailler lorsque j'aurai terminé mon déjeuner.		TD	J., D., O., C. & S. – 0 < 0.3			TD	
Txt.2	Je me serai trompé dans mon calcul.							
Txt.1	Je suis à Montréal depuis une semaine.		TD	J., & S. – 0.1314 D.–0.23 O. –0.27 C. – 0.2343 < 0.3			TD	
Txt.2	Je suis à Montréal depuis le 3 juillet.							
Txt.1	Je vais prendre une douche.	TS		J., D., O., C. & S. – 1 > 0.3	TS			
Txt.2	Il va prendre une douche.							
Txt.1	Je mange à la cafétéria chaque jour.	TS		J., D., O., C. & S. – 1 > 0.3	TS			
Txt.2	Il mange à la cafétéria chaque jour.							
Txt.1	Le professeur répète toujours la même chose.		TD	J., D., O., C. & S. – 0 < 0.3			TD	
Txt.2	Elle s'appelle Marie.							
Txt.1	Je sais cela depuis lundi.		TD	J. & S. – 0.07059 D. –0.13 O. –0.17			TD	
Txt.2	Je sais cela depuis une semaine.							

				C. –0.13 < 0.3				
Txt.1	Finissez vos devoirs!		TD	J., D., O, C., & S. – 0 < 0.3			TD	
Txt.2	Ne discute pas!							
Txt.1	Tu chantes la Marseillaise. Tu la chantes.		TD	J. & S. – 0.003 D. – 0.00708 O. – 0.00854 C. – 0.00718 < 0.3			TD	
Txt.2	Vous avez rencontré Marie. Vous l'avez rencontrée.							
Txt.1	Il téléphone à mon amie. Il lui téléphone.	TS		J. – 0.8 D., O. & C. – 0.88933 S. – 1 > 0.3	TS			
Txt.2	Il téléphone à mon amie. Il lui téléphone.							
Txt.1	Ton chien est petit, mais le mien est gros.	TS		J., D., O., C. & S. – 1 > 0.3	TS			
Txt.2	Ton chien est petit, mais le sien est gros.							
Txt.1	Je ne crois pas qu'il faille se fâcher.		TD	J., D., O., C. & S. – 0 < 0.3			TD	
Txt.2	Je n'ai pas voulu en arriver là.							
Txt.1	Que fais-tu ce matin?	TS		J., D., O., C. & S. – 1 > 0.3	TS			
Txt.1	Qu'est-ce que tu fais ce matin?							
Txt.1	Cet appartement est moderne.		TD	J., D., O., C. & S. – 0 < 0.3			TD	
Txt.2	Cet hôtel est splendide.							

Now, we will calculate the accuracy of some multilingual sentences i.e. English ⇋ French in the equivalent way which is given below:

Table 61: Accuracy Measuring of Our Methodology using Multilingual Sentences

	English ⇋ French Sentences			**Our Method**				
		True Similar	**True Dissimilar**	**Results (Similar Values)**	**True Similar**	**False Similar**	**True Dissimilar**	**False Dissimilar**
Txt.1	I am going to university.	TS		D. –0.35 O. –0.51 C. –0.37 > 0.3	TS			
Txt.2	Je vais à l'université.							
Txt.1	I am reading philosophy.	TS		D. –0.32 O. –0.39 C. –0.32 > 0.3	TS			
Txt.2	Je lis de la philosophie.							
Txt.1	Who will send him this information?	TS		D. –0.43 O. –0.52 C. –0.43 > 0.3	TS			
Txt.2	Qui lui enverra cette information?							
Txt.1	Perfect ressemblance of a portrait.	TS		J. – 0.36 D. & O. –0.5556 C. –0.54 S. – 0.3665 > 0.3	TS			
Txt.2	Parfaite ressemblance d'un portrait.							
Txt.1	The Information was bad.	TS		D. –0.36 O. – 0.6 C. – 0.398 > 0.3	TS			
Txt.2	L'information était mauvaise.							
Txt.1	He will go to a restaurant for soup and salad.	TS		D. –0.39 O. –0.53 C. –0.41 > 0.3	TS			
Txt.2	Il ira à un restaurant soupe et salade.							

Txt.1	Could you give some vinaigrette in the salad?	TS		D. –0.37 O. – 1 C. –0.48 > 0.3	TS			
Txt.2	Pourriez-vous mettre de la vinaigrette dans la salade?							
Txt.1	Dan was not amazed by the transformation of Linda.	TS		D. –0.39 O. –0.44 C. –0.39 > 0.3	TS			
Txt.2	Dan n'était pas émerveillé par la transformation de Linda.							
Txt.1	She loves to sing.	TS		J., D., O., C. & S. – 0 < 0.3				FD
Txt.2	Elle aime chanter.							

From the above three tables – 58, 59 & 60, the values of True Similar (TS), True Dissimilar (TD), False Dissimilar (FD) and False Similar (FS) are 26, 18, 4 and 1 respectively. Here, we find out the values of TS, TD, FD & FS from all the examples which is described in the section of 'Several Examples' in Methodology chapter i.e. examples of different grammatical rules of both English & French as well as examples of multilingual sentences i.e. English ⇆ French translations respectively.

Now using some methods of machine learning we can find out the rate of true similar and true dissimilar of our methodology. Then we can find out the accuracy of our methodology. Besides, using these values we can also find out the similar predictive value and dissimilar predictive value of our methodology like the following way [41] –

$$True\ Similar\ Rate = \frac{TS}{TS + FD} = \frac{26}{26 + 4} = 0.8667 = 86.67\%$$

$$True\ Dissimilar\ Rate = \frac{TD}{TD + FS} = \frac{18}{18 + 1} = 0.9474 = 94.74\%$$

$$Accuracy = \frac{TS + TD}{TS + TD + FS + FD} = \frac{26 + 18}{26 + 18 + 1 + 4} = 0.89796 = 89.796\%$$

$$Similar\ Predictive\ Value = \frac{TS}{TS + FS} = \frac{26}{26 + 1} = 0.962963 = 96.2963\%$$

$$Dissimilar\ Predictive\ Value = \frac{TD}{TD + FD} = \frac{18}{18 + 4} = 0.8182 = 81.82\%$$

So, from the above calculations, we can say that the accuracy of our method is 89.796% which is quite satisfying without taking any help of the online dictionary and any search engine etc. The true similar and dissimilar rate of our method are 86.67% & 94.74% respectively.

Now, with the help of Table – 58, 59 & 60, we will compare these values of our method with Akermi and Faiz's method [2] as well as Takale & Nandgaonkar's methodlogy [5]. We will follow the same rules which are described before to find out the TS, TD, FS and FD respectively. As Akermi and Faize's method have only one result, so we are going to be compare only their presented result with the threshold value (0.3). As Takale and Nandgaonkar's method [5] have five similarity measuring coefficients, so, we are going to follow the equivalent way that we followed before. The whole procedure is given below –

Table 62: Accuracy comparison

Actual Type of Texts	Sentences	Our Method	Akermi and Faiz's Method (Similar Value)	Takale & Nandgaonkar's Method (Similar Value)
TS	Computing	Yes	Result: 0 < 0.3 = FD (No)	Result: Error
	Recomputing			
TD	Karen rode on a yellow skate board.	Yes	Result: 0.358 > 0.3 = FS (No)	Result: 0 < 0.3 = TD (Yes)
	Karen lives in the yellow house.			
TD	I want to be a good writer.	Yes	Result: 1 > 0.3 = FS (No)	Result: 1 > 0.3 = FS (No)
	My friend is a really good writer.			
TS	Please sit down.	Yes	Result: 1 > 0.3 = TS (Yes)	Result: 1 > 0.3 = TS (Yes)
	I need you to sit down now!			
TD	When are you going to turn in your writing assignment?	Yes	Result: 0.19 < 0.3 = TD (Yes)	Result: J. – 0.4; D. – 0.57; O. – 0.67; C. – 0.58; S. – 0.5 > 0.3 = FS (No)
	Do you know what the weather will be tomorrow?			
TD	It is too dangerous to climb that mountain!	Yes	Result: 0 < 0.3	Result: 0 < 0.3 =

	I got an A on my book report!		= TD (Yes)	TD (Yes)
TS	While Tom reads novels, Jack reads comics, but Sam reads only magazines.	Yes	Result: 0 < 0.3 = FD (No)	Result: 0 < 0.3 = FD (No)
	Tom reads novels, but Jack reads comics because books are too difficult.			
TS	Jack, who reads comics, rarely reads novels; however, Tom enjoys novels.	Yes	Result: 0.46 > 0.3 = TS (Yes)	Result: 0 < 0.3 = FD (No)
	People like Jack, who reads comics rarely reads novels; they often find books difficult.			
TS	Moe played the tune more lyrically than Sam.	Yes	Result: 0.535 > 0.3 = TS (Yes)	Result: 0.33 > 0.3 = TS (Yes)
	Jack played the tune most lyrically.			
TD	The early train arrives at 8.45 a.m.	Yes	Result: 0.844 > 0.3 = FS (No)	Result: 0 < 0.3 = TD (Yes)
	The 8.45 a.m. train arrived early.			
TS	I am eating Rice.	No	Result: 0.125 < 0.3 = FD (No)	Result: 0 < 0.3 = FD (No)
	Rice is being eaten by me.			
TS	I am Saima Sultana. I live in Canada.	Yes	Result: Unable to detect name.	Result: 1 > 0.3 = TS (Yes)
	I live in Canada. I am Saima Sultana.			
TD	She is a good girl.	No	Result: 1 > 0.3 = FS (No)	Result: 1 > 0.3 = FS (No)
	She is a bad girl.			
TS	I didn't say anything.	No	Result: 0.42 > 0.3 = TS (Yes)	Result: 0 < 0.3 = FD (No)
	I said nothing.			
TD	I am a master's student in UQTR.	Yes	Result: 0.2 < 0.3 = TD (Yes)	Result: 0.0286 < 0.3 = TD (Yes)
	UQTR is a beautiful university.			

From the above table, we find, the values of TS, TD, FS and FD from Akermi and Faiz's methodology are – 4, 3, 4 and 3 respectively and the values of TS, TD, FS and FD from Takale & Nandgaonkar's methodology are – 3, 4, 3 and 4 respectively. Now we can find out the accuracy of them respectively which are given below –

- Methodology of Akermi and Faiz –

$$True\ Similar\ Rate = \frac{TS}{TS + FD} = \frac{4}{4+3} = 0.57 = 57\%$$

$$True\ Dissimilar\ Rate = \frac{TD}{TD + FS} = \frac{3}{3+4} = 0.4286 = 42.86\%$$

$$Accuracy = \frac{TS + TD}{TS + TD + FS + FD} = \frac{4+3}{4+3+4+3} = 0.5 = 50\%$$

$$Similar\ Predictive\ Value = \frac{TS}{TS + FS} = \frac{4}{4+4} = 0.5 = 50\%$$

$$Dissimilar\ Predictive\ Value = \frac{TD}{TD + FD} = \frac{3}{3+3} = 0.5 = 50\%$$

- Methodology of Takale & Nandgaonkar –

$$True\ Similar\ Rate = \frac{TS}{TS + FD} = \frac{3}{3+4} = 0.4286 = 42.86\%$$

$$True\ Dissimilar\ Rate = \frac{TD}{TD + FS} = \frac{4}{4+\ 3} = 0.57 = 57\%$$

$$Accuracy = \frac{TS + TD}{TS + TD + FS + FD} = \frac{3+4}{3+4+3+4} = 0.5 = 50\%$$

$$Similar\ Predictive\ Value = \frac{TS}{TS + FS} = \frac{3}{3+3} = 0.5 = 50\%$$

$$Dissimilar\ Predictive\ Value = \frac{TD}{TD + FD} = \frac{4}{4+4} = 0.5 = 50\%$$

We got the accuracy level of our method is – 89.796%and their (Akermy & Faiz and Takale & Nandgaonkar) accuracy level is – 50% and 50% respectively. Besides, the rates of true similar and dissimilar are – 86.67% and 94.74% respectively. From the method of Akermy and Faiz, the rates of true similar and dissimilar are – 57% and 42.86% severally. Besides, from the method of Takale and Nandgaonkar, the rates of true similar and dissimilar are – 42.86% and 57% respectively. So, we can definitely say that, we get the highest true similar and dissimilar rate as well as the accuracy level is greater than their results. On the other hand, we also noticed that, their method is unable to detect any French texts or multilingual sentences. Besides, our method is free from these types of limitations.

From above experimentation, we can conclude that our n-gram is most beneficial and well-fixed algorithm than others because our method need not to follow any grammatical rules, any kind of search engine as well as dictionaries. Comparing our result with others, we get a satisfactory result than them. We can find out the main advantages of our method which is given below:

- Process random sentences i.e. any syntax of grammar,
- Does not follow any dictionary so the processing is quite simple and takes less time than others.
- Does not follow the grammar rules, it can learn by itself which could be beneficial for further implementation.

- Does not follow any certain web-pages like Wikipedia, so it could be more versatile and the results would be more authentic.
- Does not depend on some certain search engines like Digg.com as well as Google.com which prevents getting error for over- accessing as well as falsify or exaggerated information.
- Using dictionary and search engine sometimes may have problem while exceeding the usage per day i.e. for if the usage of dictionary crossed 500 times per day then it shows error. But, regarding to our method, we do not have such kind of problems.
- Beside the most important part of our algorithm is learning by itself.
- Some dictionaries might be case sensitive. So sometimes it might be a great problem. In our case, we do not have such dilemma.
- The great advantage of our algorithm is multilingualism i.e. our method would be beneficial to find out the similarity between other languages as well as the translated sentences like English ↔ French.
- Our method can be used to process any kind of languages like English, French, Chinese, Arabic, Bangla etc. Though this time we tried our method to process English and French sentences, but in near future we will make it more versatile for any kind of languages.

Regardless of few limitations we got the accuracy level of our method is 89.796% and the true similar and dissimilar rate of our method are 86.67% & 94.74% respectively. These results are very much satisfying comparing to other method. So, we will take some other necessary steps to make the similarity detection even more precise to avoid such kind of limitations further.

CHAPTER VI

CONCLUSION

In this research, our aim is to find out the similarity and dissimilarity between two large texts in an authentic way, that would give a precise result and our method would be used universally. We have formulated a new concept for measuring similarities as well as dissimilarities using *n-grams of characters* without taking any help from online dictionary, search engine (Digg.com or Google) or certain webpages (Wikipedia). In this way, our method becomes more authentic and unique than others. Besides, it is free from any grammatical rules which would make it more versatile. We presented state of the art where we discuss about many others method like Akermy and Faiz, Takale & Nandgaonkar and so on.

Though Akermy and Faiz use online dictionary and search engine to get a good result, but we think it would be a great limitation while using their method in real life. Because sometimes if their used online dictionary exits 500 queries per day then it is unable to execute. Sometimes, if their used search engine does not load properly then their method gives incorrect result. Besides, the main lacking is that their algorithm is very heavy and complex which consumes more process time and space and it would be a big problem to execute in any local system. But, our method is free from these kinds of lacking. We proved that our method is better than their method.

To establish our method, firstly, we collect two large texts from various kind of data sources like pdf, doc, docx, plain text file, ppt, pptx or web-pages separately. Then we preprocess both raw datasets by removing stop-words, punctuations, delimiters etc. so that we can focus on fundamental data-sets. Then we split our both filtered datasets into *trigrams of characters* separately. Then we create two different distance matrices of those *trigrams of characters* separately. Afterward, we can able to find out the intersection, union, or the size of each matrix values. Finally, using these values as well as the equations of popular co-occurrence measures – Jaccard, Dice, Overlap, Cosine and Simple, we can find out the similarity and dissimilarity values more precisely in the scale of *(0, 1)*.

We have evaluated our method which gives a quite good result than other method as well. Besides, it is noticeable that, in some experimentations, our method gives unpredictable result while comparing two sentences like 'I am eating rice' and 'Rice is being eaten by me' and so on. But overall our method gives a predictable result. But we can analyze these problems and improve these limitations as well. Our method is wide-ranging enough to incorporate artificial intelligence (AI), if needed. In the future, we would like to test our text similarity method for long documents and would like to integrate our method into other applications or other languages.

BIBLIOGRAPHY

1. Witten, I.H., Text mining. Practical handbook of Internet computing, 2005: p. 14-1.
2. Akermi, I. and R. Faiz, An Approach to Semantic Text Similarity Computing, in Modern Trends and Techniques in Computer Science: 3rd Computer Science On-line Conference 2014 (CSOC 2014), R. Silhavy, et al., Editors. 2014, Springer International Publishing: Cham. p. 383-393.
3. Akermi, I. and R. Faiz, Hybrid Method for Computing Word-Pair Similarity based on Web Content, in Proceedings of the 2nd International Conference on Web Intelligence, Mining, and Semantics. 2012, ACM: Craiova, Romania. p. 1-4.
4. Kumari, P. and K. Ravishankar, Measuring Semantic Similarity between Words using Page-Count and Pattern Clustering Methods. 2013.
5. Takale, S.A. and S.S. Nandgaonkar, Measuring semantic similarity between words using web documents. International Journal of Advanced Computer Science and Applications (IJACSA), 2010. 1(4).
6. Rijsbergen, C.J.V., Information Retrieval. 1979: Butterworth-Heinemann. 208.
7. Porter, M.F., An algorithm for suffix stripping. Program, 1980. 14(3): p. 130-137.
8. Bollegala, D., Y. Matsuo, and M. Ishizuka. Websim: A web-based semantic similarity measure. in Proc. of 21st Annual Conference of the Japanese Society of Artitificial Intelligence. 2007.
9. Manning, C., Foundations of Statistical Natural Language Processing. Natural language engineering, 2002. 8(1): p. 91-92.
10. Platt, J., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers, 1999. 10(3): p. 61-74.
11. Islam, A., E. Milios, and V. Kešelj. Text similarity using google tri-grams. in Canadian Conference on Artificial Intelligence. 2012. Springer.
12. Kondrak, G. N-gram similarity and distance. in International Symposium on String Processing and Information Retrieval. 2005. Springer.

13. William B. Cavnar. Using an n-gram-based document representation with a vector processing retrieval model. In TREC-3, pages 269–278, 1994.
14. Miao, Y., V. Kešelj, and E. Milios. Document clustering using character N-grams: a comparative evaluation with term-based and word-based clustering. in Proceedings of the 14th ACM international conference on Information and knowledge management. 2005. ACM.
15. Biskri, I. and L. Rompré, Using association rules for query reformulation, in Next Generation Search Engines: Advanced Models for Information Retrieval. 2012, IGI Global. p. 291-303.
16. Laouamer, Lamri, Ismaïl Biskri, and Benamar Houmadi. "Towards an automatic classification of images: Approach by the n-grams." Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Orlando, FL. 2005.
17. Sidorov, G., Syntactic dependency based n-grams in rule based automatic English as second language grammar correction. International Journal of Computational Linguistics and Applications, 2013. 4(2): p. 169-188.
18. White, O., et al., A quality control algorithm for DNA sequencing projects. Nucleic Acids Research, 1993. 21(16): p. 3829-3838.
19. Sidorov, G., et al., Syntactic N-grams as machine learning features for natural language processing. Expert Systems with Applications, 2014. 41(3): p. 853-860.
20. Manning, C.D. and H. Schütze, Foundations of statistical natural language processing. Vol. 999. 1999: MIT Press, ISBN 0-262-13360-1.
21. Raaijmakers, S. and W. Kraaij. A shallow approach to subjectivity classification. in ICWSM. 2008.
22. Damashek, M. Gauging similarity with n-grams: Language-independent categorization of text. in Science, 267 (1995), pp. 843 - 848.
23. Wilson, T. and S. Raaijmakers. Comparing word, character, and phoneme n-grams for subjective utterance recognition. in INTERSPEECH. 2008.
24. Kanaris, I., et al., Words versus character n-grams for anti-spam filtering. International Journal on Artificial Intelligence Tools, 2007. 16(06): p. 1047-1067.

25. Bordag, S. A comparison of co-occurrence and similarity measures as simulations of context. in International Conference on Intelligent Text Processing and Computational Linguistics. 2008. Springer.

26. Gomaa, W.H. and A.A. Fahmy, A survey of text similarity approaches. International Journal of Computer Applications, 2013. 68(13).

27. Kosub, S., A note on the triangle inequality for the Jaccard distance. arXiv preprint arXiv:1612.02696, 2016.

28. Lipkus, A.H., A proof of the triangle inequality for the Tanimoto distance. Journal of Mathematical Chemistry, 1999. 26(1): p. 263-265.

29. Levandowsky, M. and D. Winter, Distance between Sets. Nature, 1971. 234(5323): p. 34-35.

30. Sørensen, T., A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. Biol. Skr., 1948. 5 (4): p. 1-34.

31. Dice, L.R., Measures of the amount of ecologic association between species. Ecology, 1945. 26(3): p. 297-302.

32. Gallagher, E., COMPAH documentation. University of Massachusetts, Boston, 1999.

33. Giller, G.L., The statistical properties of random bitstreams and the sampling distribution of cosine similarity. 2012.

34. Segaran, T., Programming collective intelligence: building smart web 2.0 applications. 2007: " O'Reilly Media, Inc.".

35. Grefenstette, G. Comparing two language identification schem. in JADT, 1995.

36. Huffman, S. and M. Damashek. Acquaintance: A novel vector-space n-gram technique for document categorization. in NIST Special Publication, National Institute of Standards and Technology, 1995, pp. 305 - 310.

37. Biskri, I. and S. Delisle. Les n-grams de caractères pour l'aide à l'extraction de connaissances dans des bases de données textuelles multilingues. in Proceedings of TALN-2001, 2001, pp. 93-102.

38. Sipser, M., Introduction to the Theory of Computation. Vol. 2. 2006: Thomson Course Technology Boston.

WEBOGRAPHY

39. contributors, W. Text mining. 2016 02:35 UTC 19 August 2016 07:30 UTC [cited 2016 20 August]; Available from: https://en.wikipedia.org/wiki/Text_mining.

40. Redfearn, N.T.M.C.a.J. Text Mining. 15 September 2008 [cited 2008 15 September]; Version 2: [Available from:

http://www.webarchive.org.uk/wayback/archive/20140614041852/http://www.jisc.ac.uk/publications/briefingpapers/2008/bptextminingv2.aspx.

41. http://www.uta.fi/sis/tie/tl/index/Rates.pdf