DE-NOISING OF SMS TEXT USING TERNARY TREE FOR FAQ RETRIEVAL

A DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF DEGREE
OF
MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted by:

VIPRA BHATT
2K18/CSE/20

Under the supervision of
Dr. Manoj Kumar
(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

JUNE, 2020

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi - 110042

CANDIDATE'S DECLARATION

I, Vipra Bhatt, Roll No. 2K18/CSE/20 student of M. Tech (Computer Science and Engineering), hereby declare that the project Dissertation titled **"**De-noising of SMS Text using Ternary Tree for FAQ Retrieval**"** which is submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of and Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi — Vipra Bhatt

Date: — 2k18/CSE/20

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi - 110042

CERTIFICATE

I hereby certify that the Project Dissertation titled **"De-noising of SMS Text using Ternary Tree for FAQ Retrieval"** which is submitted by Vipra Bhatt, 2K18/CSE/20 Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date:

Dr. Manoj Kumar

Associate Professor

Head (Computer Center)

Department of CSE

ACKNOWLEDGMENT

The success of this project requires the assistance and input of numerous people and the organization. I am grateful to everyone who helped in shaping the result of the project.

I express my sincere thanks to **Dr. Manoj Kumar**, my project guide, for providing me with the opportunity to undertake this project under his guidance. His constant support and encouragement have made me realize that it is the process of learning which weighs more than the end result. I am highly indebted to the panel faculties during all the progress evaluations for their guidance, constant supervision and for motivating me to complete my work. They helped me throughout with new ideas, provided information necessary and pushed me to complete the work.

I also thank all my fellow students and my family for their continued support.

VIPRA BHATT

2K18/CSE/20

ABSTRACT

Everyone in today's world uses mobile and technology and can get any information sitting at home within no time. A lot of information is available over the web, which is easily accessible to us. But the low-cost mobile devices don't have access to the internet. Every device ranging from inexpensive feature phone to the smartphones have the facility of Short messaging Service (SMS). So, instead of using the internet facility, most of the people prefer to use the facility of SMS to access the information. Due to the limited number of characters in SMS, we need to remove a few characters, and thus, these texts become noisy. There are many other intentional and non-intentional errors that add noise to the SMS. Intentional errors include few intentional dropping of characters, phonetic substitutions, abbreviations, etc, and non-intentional errors include typing error and errors due to the small screen, damaged display, multi-tap keyboard, QWERTY keyboard, etc. Efficient de-noising of such texts is necessary to remove noise for the correct information gain. A model is developed for de-noising the SMS text by calculating prefix, suffix, and similarity score to find the best matching word corresponding to the noisy term. Prefix and suffix scores are calculated using Ternary Tree, and the similarity score uses the Longest Common Subsequence to get the correct English word for the noisy SMS word. The overall score is calculated using the above three scores. The ternary tree helps in reducing the memory space as compared to the previous models using trie for de-noising the text. Few noisy words are generated to test the model, and the results are compared with some previous models. The proposed novel method outperforms all the previous approaches.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. NLP: Natural Language Processing
2. IR: Information Retrieval
3. SMS: Short Messaging Services
4. FAQ: Frequently Asked Questions
5. Q-A: Question Answering
6. WED: Weighted Edit Distance
7. TST: Ternary Search Tree
8. LCS: Longest Common Subsequence
9. ED: Edit Distance
10. MRR: Mean Reciprocal Rank

CHAPTER 1

INTRODUCTION

1.1 NATURAL LANGUAGE PROCESSING

Humans can interact with each other with their natural language. They can speak, transfer their ideas, and understand each other's opinions, but the machine cannot do so. The machine systematically needs instructions to work. We need to make the computer understand the natural language so that humans and machines can interact with each other. Natural language processing helps in achieving such tasks.

Natural Language Processing (NLP) is the branch of Computer Science or Artificial Intelligence that makes humans and machines interact with each other. It involves an interaction between natural language and machine language. It trains the computer or device to interact with the human language such as speech or text. It makes the machine read, understand, and generate meaning for human language. Natural language processing involves speech recognition, understanding, and generation of natural language.

1.2 INFORMATION RETRIEVAL

Information retrieval (IR) is the process of storing and accessing the information from the database. The goal is to find the document from the database according to the user's needs. The user asks the query; the most relevant answer is seen from the database and is returned to the user. Today, with the advancement in technology and the easy accessibility to the internet, we can quickly retrieve the information. Sometimes, the retrieved information might not be according to the user's requirement. The aim is to get

the correct information as per the need. Relevant feedback is provided so that the user can improve his/her query and get the best answer.

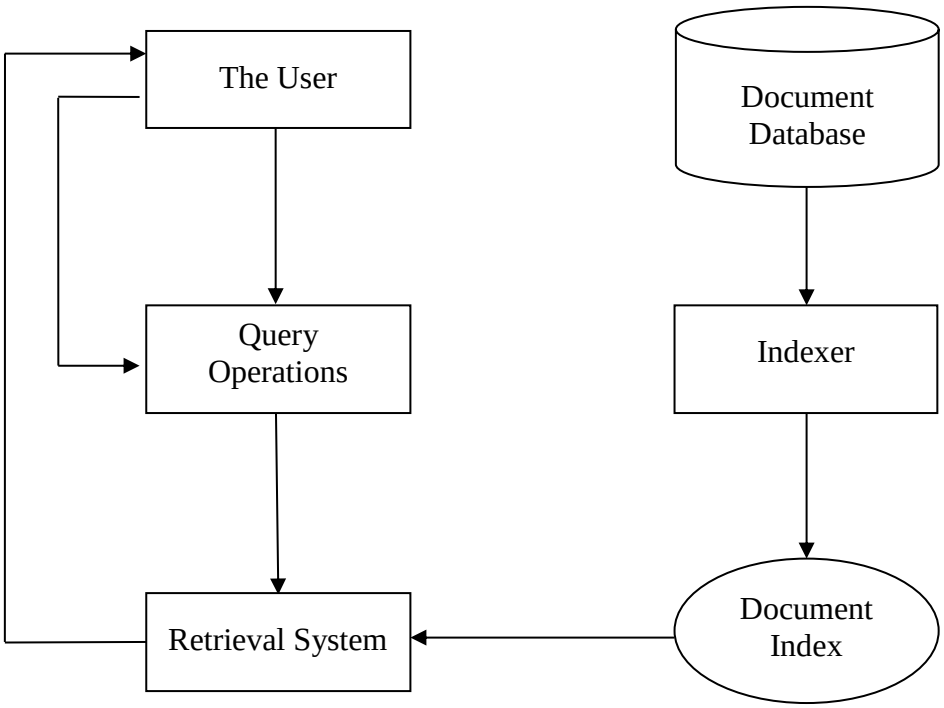

Figure 1.1. The basic block of Information Retrieval System

1.3 SMS BASED FAQ RETRIEVAL

With the increase in mobile technology, everyone is using mobile phones nowadays. Mobile phones have become the most widely used device due to their low cost, portability, and easy accessibility. A lot of information is available over the internet and is freely available to us. But in our country, not every individual uses the internet facility as most of the population doesn't know how to use the smartphones, and they use the lost cost feature phone. The facility of SMS is available on every mobile phone. So, everyone can use SMS for accessing information from the frequently asked questions (FAQs) in various fields like agriculture, health, travel, railway reservation, insurance, banking, etc. Using SMS for information retrieval saves time and money. It prevents us from the hassle of going to a particular place and also to avoid waiting in a queue. Some businesses also provide the facility of queries in natural language.

But, the number of characters is limited in SMS, i.e., 160 per SMS, so there is a need to shorten the word while writing the SMS query, which makes the SMS noisy. Also, there are many other factors such as the small screen, damaged display, multi-tap keyboard in low-cost mobile phones where the same key is pressed repeatedly to type a character, QWERTY keyboard, which makes SMS query noisy. Such noisy texts are the combination of unintentional errors and a few intentional dropping of characters, phonetic substitutions, and abbreviations.

In SMS, we have to deal with many non-English words and ungrammatical sentences. Noisy SMS refers to the words misspelled, characters removed, abbreviated, transliterated, or truncated and are not present in the actual dictionary. Information access via SMS is based on :

- Human Intervention:
  In the human intervention based method, human experts are required to interpret the SMS query the user has asked and return the answer to the user for the question.

- Automatic information retrieval system:
  An automatic system is developed where the system automatically removes the noise from the SMS and get the correct answer from the FAQ database. FAQ based retrieval system consists of the given set of questions and answers, and the aim is to get the question that matches the most, i.e., the question with the highest score and returns its corresponding answer.

1.4 TYPES OF SMS BASED FAQ RETRIEVAL

SMS based FAQ retrieval is divided into three subtasks:

- Monolingual FAQ retrieval:
  In Monolingual subtask, SMS and the FAQ set are written in the one language, say L1. In this, find the question that matches the most from the FAQ corpus of the same language.

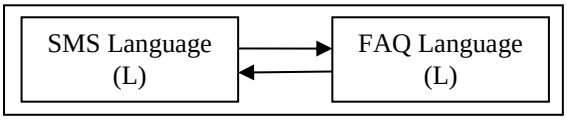

Figure 1.2. Retrieval of FAQ based on monolingual [14]

- Cross-lingual FAQ retrieval:
  In the cross-lingual subtask [15], the language of the FAQ database and SMS queries are different. In this, find the most similar questions from FAQ corpus of one language (L2) while the incoming query is written in another language (L1).

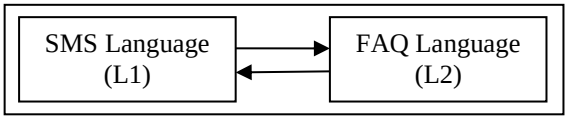

Figure 1.3. Retrieval of FAQ based on cross-lingual [14]

- Multilingual FAQ retrieval:
  In a multilingual subtask, the SMS queries are written in different languages and are matched with the FAQ database from a different language. However, the question and its corresponding answer to FAQ should be in the same language.

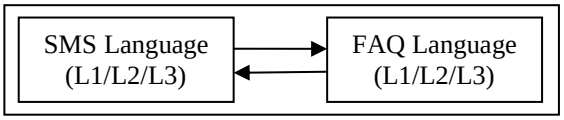

Figure 1.4. Retrieval of FAQ based on multilingual [14]

1.5 TYPES OF NOISE IN SMS

Different types of noise [1, 2] in SMS text includes:

- Deletion of Characters:
  Deletion of vowels and repeated characters to make the word shorter. Such as "rmv" for "remove" or "schl" for "school".

- Truncation:
  Truncation of words is done to reduce the number of characters in the word. Such as "sat" for "saturday", "prep" for "preparation" and "approx" for "approximate".

- Abbreviations:
  The use of abbreviations is widespread amongst youngsters. Such as "IDK" for "I Don't Know", "BTW" for "By The Way", and "FYI" for "For Your Information".

- Phonetic substitution:
  Their shorter substitute replaces the same sounding words. For example, "r" for "are" and "ur" for "your" or "you are".

- Dialects and informal use:
  Multiple words are combined to a single word such as "gotta" for "have got to" or "gimme" for "give me".

- Deleting articles and pronouns:
  Deleting articles and pronouns from the sentences to reduce its length. Such as "gng schl" for "I am going to school".

Other types of noise include grammatical mistakes, spelling errors, mixing of words of two languages such as the use of interjections ("ooch", "ooh," "huh") and emoticons (☺, ☹), and many others are an example of noise in texts. The use of Hinglish while speaking and writing documents is another type of noise. For example, "Hungry

Kya?" - the tagline of Domino's pizza is the combination of an English and a Hindi word. These words may look normal for humans to understand but are difficult for a computer or an automated system to follow. So, there is a need to de-noise them efficiently. A single English word "tomorrow" [13] can be represented in many versions - tomorro, tomoro, tomorrrow, tmrrw and tomrw, 2morrow, 2mrow, 2morow, 2moro, 2morro, 2mrrw, 2mrw. Various measures have been taken to remove noise for the SMS effectively for the correct information access.

<table>
<tr><td rowspan="8">SMS Noise</td><td rowspan="6">User Improvisation</td><td colspan="2">Deletion of words</td></tr>
<tr><td colspan="2">Deletion of characters</td></tr>
<tr><td colspan="2">Truncation</td></tr>
<tr><td colspan="2">Repeated character removal</td></tr>
<tr><td colspan="2">Abbreviations</td></tr>
<tr><td colspan="2">Phonetic Substitution</td></tr>
<tr><td rowspan="4">Typographical Errors</td><td rowspan="3">Input Device</td><td>Multitap Keyboard</td></tr>
<tr><td>Qwerty keyboard</td></tr>
<tr><td>Touch Screen</td></tr>
<tr><td colspan="2">Typing Errors</td></tr>
</table>

Table 1.1. SMS noise classification

CHAPTER 2

PRIOR WORK

A significant amount of work [3-10] has been done in the field of de-noising SMS queries. The text written in SMS language is generally misspelled, transliterated, have grammatical mistakes, abbreviations, etc. Many models have been proposed to automate the SMS based FAQ retrieval system. The proposed model mainly focuses on the algorithm used in the paper by Agarwal et al. [3] to de-noise the SMS text and uses the ternary tree to calculate matching prefix and suffix length. Noise is efficiently removed from the SMS words by finding the top seven matching candidates for the noisy word. The overall score to calculate the matching depends on matching prefix length and matching suffix length and similarity score between the SMS words and the English dictionary words. Based on the total score, a list of the top seven best matching candidates are generated for each of the noisy tokens.

SMS based interface for FAQ retrieval

Kothari et al. [4] gave an FAQ based Q-A system for the SMS interface and were the first to develop a system to handle this problem. They used the combinatorial search approach and handled semantic variation along with the similarity using the synonym dictionary. A search set comprises of the combination of all token variation in the dictionary in the noisy queries. Semantic variation refers to the words that are lexically different (different in words) but semantically similar (meaning is the same). They are the synonyms of the given words. For example, words such as “quick”, “rapid”, “speedy” are all semantic variations of “fast”. Other examples include “nation” for “country”, “originated” for “started”.

They use two algorithms, namely, naïve and prune algorithms, to generate a list of the candidate terms for the noisy tokens. Both the algorithm works the same but differ

in complexities. The proposed model involves the use of a dictionary, a list of SMS token variants, and noise removal. No training data on SMS normalization is required for this model. They formulated a dictionary from all the words appearing in the FAQ set and then define similarity for all words of the SMS query to the dictionary terms. The similarity is determined by the longest common subsequence ratio to the edit distance between noisy SMS token (s) and the dictionary word (t) and is given as:

$$\alpha(t,s) = \begin{cases} \frac{LCSRatio(t,s)}{EditDistance_{SMS}(t,s)} & ,if\ t[0] == s[0] \\ 0 & ,otherwise \end{cases} \quad (2.1)$$

Where LCS ratio is the ratio of the length of LCS to the length of the dictionary term and edit distance is the number of operations (substitutions, deletions, and insertions) required to change one string to another. The word with the highest score is taken as the best candidate for the noisy word using the similarity score.

Improving accuracy of SMS based FAQ retrieval system

The paper given by D. Shaikh et al. [6] uses the same method in [7] to calculate the similarity measure. They make the additional change in the scoring function from [7], adding length score and proximity score along with the similarity score. The FAQ database is generated by combining many datasets from different domains such as Health, Bank, Agriculture, Railway reservation, Insurance, Tourism, etc. The SMS and FAQ are matched based on the value of the three scores: the similarity score, the proximity score, and the length score. They give their model for monolingual English and monolingual Hindi tasks and achieved excellent results. The similarity measure remains the same as in [7]. Methods of list creation and candidate set generation remain the same as in [7].

They experimented in four ways (considering the similarity score in all of them):

1). The only Similarity score for matching.
2). Similarity and Proximity score are considered.
3). Similarity score and Length score are taken.
4). All three (similarity, length, and proximity) scores are taken. Best accuracy is achieved while considering all three of them.

SMS based FAQ retrieval

The method used by Shivhre et al. [8] is for finding the ranked list of words in the same as defined in [7]. They use the weight function to find the similarity between the dictionary terms and the noisy SMS token. The weight function between the SMS token and the dictionary term uses the LCS ratio between them, the similarity ratio, the inverse document frequency of the word in the dictionary, and the Levenshtein distance between the two. The similarity ratio is twice the number of common characters between SMS and dictionary terms to the number of characters in both. These methods, combined, provide the measure to find the weight for all dictionary terms for the noisy SMS token. Weight of the dictionary term t w.r.t. SMS token s is given by:

$$Weight(t,s) = \frac{LCSR(t,s) * SMRatio(t,s) * IDF(t)}{LevDistance(t,s)} \quad (2.2)$$

Weighted edit distance based FAQ retrieval using noisy queries

S. Mhaisale et al. [9] use the weighted edit distance (WED) to find the similarity between the SMS and the dictionary word instead of constant edit distance. Instead of giving equal weights to insertion, deletion, and substitution as in constant edit distance, they use normalization of the query, then the scoring model (corpus-based), to get the relevant FAQ question list. Instead of giving equal importance to each of the string editing operations (substitutions, deletions, and insertions), they provide relative importance to them. The emphasis on each one of them is given in decreasing order as:

Substitutions >Insertions > Deletions

Substitution of words such as “you” for SMS token “u,” “office” for “ofc,” etc. are more common than the insertion of “are,” “am,” “is.” The use of words such as “huh”, “ahem” is sporadic. So, the deletion of words is the least frequent operation to be performed. Character insertion is more than character deletion. Also, consonant insertion is less than vowel insertion as vowels are generally removed from SMS token. At the end, the insertion of character is more frequent as truncation is done for the end of the word. So, lower cost is assigned to more frequent edit operations. Better exactness is obtained in

this model than a model in [4]. The weighted edit distance between SMS token (s) and the dictionary term (t) is given as:

$$WED(t,s) = 0.3 * x_1 + 0.5 * x_2 + 0.3 * x_3 + 0.5 * x_4 + 2.0 * x_5 \quad (2.3)$$

Where x_1, x_2, x_3, x_4, x_5 denotes the number of vowel insertions, consonant insertions, insertions at the end of token, vowel substitutions, and deletions, respectively. The similarity in their approach was based on inverse document frequency, LCS ratio, and the weighted edit distance of the dictionary term t and noisy term s and it is given as:

$$Sim(t,s) = \frac{IDF(t) * LCSRatio(t,s)}{WeightedEditDistance(t,s)} \quad (2.4)$$

Efficiently denoising SMS text for FAQ retrieval

In the approach [4], there is a need to look at the entire dictionary to get the list of variants for SMS tokens. To improve the efficiency of the dictionary lookup, R. Batra et al. [10] limit the search space, i.e., match only those words in the dictionary whose length is higher than or is equal to the length of SMS token as a user don't type extra characters than the original word. Example, users don't type "loooong" for "long". But sometimes users may commit typos and type additional characters unknowingly, so the relaxation of 2 is provided to the user, i.e., words are taken from the dictionary if it satisfies the following condition:

$$LCS\ (s,t) \geq length\ (s) - 2 \quad (2.5)$$

This condition helps in reducing the computation time since there is no need to check the entire document. If the LCS of the dictionary term and the SMS token is equal to the length of the SMS word, then the similarity score is increased by one, i.e., if $\text{LCS}\ (\text{s},\text{t}) == \text{length}\ (\text{s}), \text{then SM+= 1}$.

Removing such words helps in improving the algorithm and helps to reduce the calculation time. Still then, predicting the best matching word is challenging as there can be many variants of the same term or token according to different users.

Agarwal et al. [3] de-noises the SMS token by calculating a similarity measure based on prefix matching, and suffix matching that was computed using prefix and suffix tries respectively and using the similarity ratio. Prefix trie was made to store all the dictionary words, and the noisy word is then searched in the trie to calculated matching prefix length for all the dictionary terms. Similarly, suffix matching is done by reverse storing the dictionary terms in the trie. Then the noisy token is reversed and is matched with all the words in the trie. Suffix trie was not created directly because, in that case, the searching will increase the complexity to a great extent. So, the words are stored by reversing the keys to match the complexity with the prefix search. Then, use all the three factors together and calculate an overall score to find the substitute for the noisy SMS token. The overall score between dictionary term t and SMS token s is given as:

$$Overall_{Score}(t) = 0.25 * Prefix_{Score}(t,s) + 0.25 * Suffix_{Score}(t,s) + Sim_{Score}(t,s) \quad (2.6)$$

Where prefix score is the length of matching prefix in the noisy term and the dictionary term, suffix score is the length of matching suffix in the noisy token and the dictionary term, and the similarity score is taken as the ratio of twice the LCS ratio to the sum of the length of noisy token and the dictionary word.

CHAPTER 3

TERMINOLOGIES USED

The noisy SMS query hinders the automated FAQ retrieval system. So, there is a need to efficiently de-noise the SMS text. De-noising of SMS text requires finding the closeness of the SMS token with all the dictionary terms. In previous work, trie is used to find the length of prefix and suffix matching and Longest Common Subsequence to find a similarity score. In our dictionary, the distribution of words starting from “b” is shown in the figure 3.1.

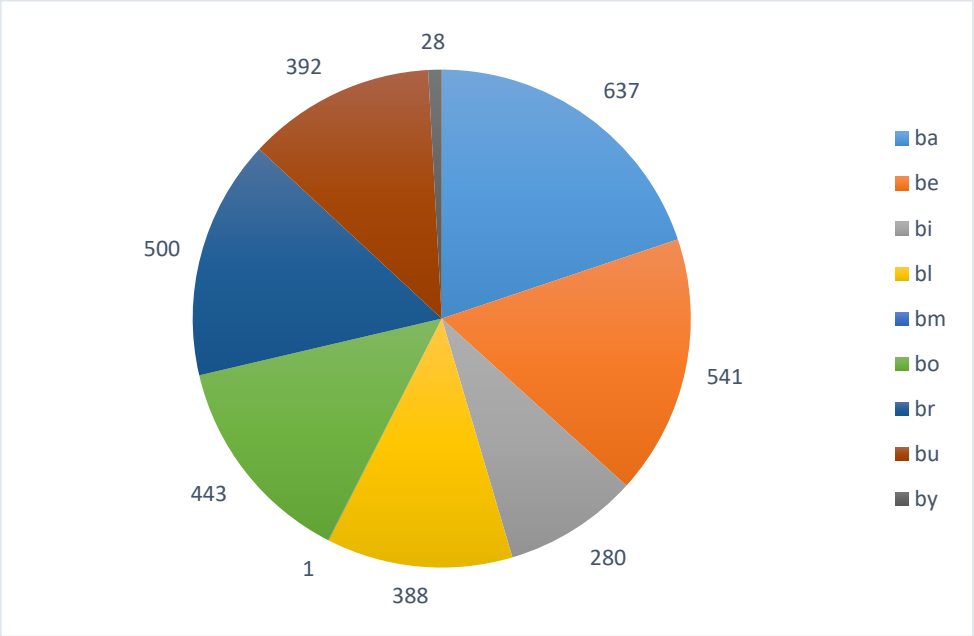

Figure 3.1. Distribution of words starting from character “b”

In the case of trie, for a node having first character ‘b’, the node will have 26 pointers to the nodes at the next level irrespective of the fact that 26 nodes are required or not. As we can see from the figure 3.1, the letter “b” does not follow every 26 characters. The dictionary contains the words starting with “ba”, “be”, “bi”, and so on and

the count of words starting with “bb”, “bc”, and many others are 0. In that case, the node for “b” in trie will point to 26 other nodes, and many unnecessary nodes are created. But in a ternary tree, only the required nodes are constructed. The memory requirement increases to a great extent in the trie. So, in such cases, the Ternary Tree is useful as the memory requirement of the ternary tree is very low as compared to trie.

To reduce the memory space, the ternary search tree is used to calculate the length of prefix and suffix matching. Then find the similarity of every noisy word with all the dictionary words and compute their respective similarity score. Combining all the three factors: prefix score, suffix score, and similarity score, we efficiently de-noises the SMS token.

3.1 TERNARY SEARCH TREE

A ternary search tree (TST) is a type of trie data structure that uses three-pointers instead of 26 (low memory space). Every node of the ternary tree has three-pointers (left, middle and right) along with the EndOfword bit and the value field where the character is stored. A node represents each character of a string. If the character is less than the current node, then it goes to the node pointed by the left pointer. Similarly, if the character is higher than the current node, it goes to the node indicated by the right pointer. And if it is equal to the current node, it goes to the middle one. It is used for various applications such as for storing string prefix, for autocomplete suggestions, etc.

The average time complexity for traversal, insertion, and deletion is O ($\log_3$ h), where h is the height of the TST. Ternary tree combines the advantage of less space of the binary tree and little character search time for the tries. Ternary tree search is faster than hashing for the words not present in TST.

Table 3.1. Time Complexities of TST

	Average	**Worst**
insert	O ($\log_3$ h)	O (h)
delete	O ($\log_3$ h)	O (h)
search	O ($\log_3$ h)	O (h)

3.2 TRIE DATA STRUCTURE

Trie is a data structure used for information retrieval efficiently and stores keys. Every node of trie consists of a maximum of 26 pointers and an EndOfWord field. The insert, search, and delete operation take O (h) time where h is the height of the trie. The length of the key determines the height of the trie. The space requirement in case of the trie is more and is O (s*n*m) where s is the size of the alphabet, n is the length of the key, and m is the number of keys stored in the trie.

For every character in a key, a new node is created. If the new key is the prefix of the existing one, mark the EndOfWord for the key's last node. Also, if the new key is the addition of the older key, create nodes for the new characters and mark EndOfWord for the last node. Searching in the trie is similar to the insertion. It is better than hashing and binary search trees. It is used for prefix search, auto-complete feature, and to print words in lexicographical order.

Table 3.2. Time Complexities of Trie

	Average	**Worst**
insert	O (h)	O (h)
delete	O (h)	O (h)
search	O (h)	O (h)

3.3 LONGEST COMMON SUBSEQUENCE

Given two strings, it gives the largest subsequence common (LCS) to both strings, i.e., the numbers of common characters. Let SMS token is s and dictionary term is t of length i and j respectively, then their LCS is given by :

$$LCS\,[i][j] = \begin{cases} 0 & ,if\ i = 0\ or\ j = 0 \\ a + 1 & ,if\ s[i-1] = t[j-1] \\ \max\,(b, c) & ,otherwise \end{cases} \quad (3.1)$$

Where $$a = LCS\,[i-1][j-1] \quad (3.2)$$

$$b = LCS\ [i-1][j] \tag{3.3}$$

$$c = LCS\ [i][j-1] \tag{3.4}$$

The higher the value of the LCS, the higher is the matching of the SMS token and the dictionary word.

3.4 EDIT DISTANCE

To modify a string to another, various operations such as insertion, substitutions, and deletions of characters is required, which is given by Edit Distance. Let s and t be the SMS token and dictionary term of length i and j, respectively. Then Edit Distance is given as:

$$ED\ [i][j] = \begin{cases} i & ,j = 0 \\ j & ,i = 0 \\ t3 & ,s[i-1] = t[j-1] \\ \min(t1, t2, t3) + 1 & ,otherwise \end{cases} \tag{3.5}$$

Where

$$t1 = ED\ [i-1][j] \tag{3.6}$$

$$t2 = ED\ [i][j-1] \tag{3.7}$$

$$t3 = ED\ [i-1][j-1] \tag{3.8}$$

The lower the edit distance, the lesser is the changes required to convert one word to another. So, prefer words with less value of edit distance.

CHAPTER 4

PROPOSED WORK

4.1 PROBLEM STATEMENT

People may type wrong words intentionally or unintentionally while texting, which adds noise to the SMS text. Such noisy texts are the combination of non-intentional errors and a few intentional dropping of characters or words, truncations, phonetic substitutions, and abbreviations. Non-intentional errors include typing errors and errors due to the small screen, damaged display, multi-tap keyboard, QWERTY keyboard, etc. There is a need to find the correct word to substitute such words to develop an automated FAQ retrieval. Therefore, a model is developed to efficiently de-noise the SMS texts using Ternary Tree.

4.2 PROPOSED METHOD

To develop an automated FAQ retrieval model, there is a need to remove noise from the SMS query. To get the correct word for the noisy SMS token, consider three factors: prefix, suffix, and similarity score. Prefix and suffix matching are done using a TST. We use a TST to store all the words of the English dictionary. Each node represents a single character and points to three children- left, middle, and right. TST is used to calculate the matching prefix length, which helps find the closeness of the noisy SMS token with the dictionary terms. Similarly, to find the matching suffix length, store all the words of the English dictionary in the reverse order in TST and similarly calculate the closeness as the prefix. The limitation of this model is that it works well for the single noisy token but fails to de-noise the two or more terms abbreviated together. For example, abbreviations such as “BRB” for “Be Right back” are not de-noised by the proposed model.

4.2.1 Prefix Score

Prefix ternary tree is created to store all the words of the dictionary. The noisy SMS token is then matched with all the dictionary terms. For the terms having minimum prefix matching of two (i.e., if two or more starting characters are matched) is stored in a hash table with key as dictionary term and the value as the prefix length score. It takes into account the noisy words that are truncated, i.e., the words whose trailing characters are cut off. For example, "nov" for "november". Here, the prefix score for the word "november" is 3. These words are correct in the beginning, so finding the prefix score helps us to get the right alternative for such noisy terms. For example, the prefix score of disagree, dislike, dimension for disable is 4, 3, and 2, respectively.

4.2.2 Suffix Score

For suffix matching, we store all the dictionary terms in the reverse order and reverse the noisy SMS token. Then this modified noisy word is matched with all the words in the suffix ternary tree, and the words with matching suffix are stored in the hash with key as dictionary term and value as the matching suffix length. The suffix ternary tree is created in the reverse order of the prefix ternary tree to reduce the time complexity as finding suffix length by directly creating a suffix tree (without reversing the dictionary terms) would be very complicated and time-consuming. Minimum matching of length one is considered in case of suffix matching.

4.2.3 Similarity Score

The third factor is the similarity score that takes into account the common characters between the noisy term and the dictionary term. The noisy word is matched with all the dictionary terms, and then the similarity score is calculated for all terms present in the dictionary. The LCS concept is used to find the similarity score between the noisy token and the dictionary term. The similarity of SMS word s and the dictionary word t is given as:

$$Sim(t,s) = \frac{2 * LCS\ (t,s)}{length(t) + length(s)} \tag{4.1}$$

Further, if the first character of noisy token and dictionary term is matched, we increase the similarity score by 0.1 times of Sim (t,s). Therefore, the similarity score is given as:

$$Sim_{Score}(t,s) = \begin{cases} Sim(t,s), & t[0]! = s[0] \\ 1.1 * Sim(t,s), & ,otherwise \end{cases} \quad (4.2)$$

4.2.4 Overall Score

Calculate the overall score for each word of the top seven terms in the prefix score list, suffix score list, and similarity score list. Prefix score of 0 is given for the term not present in the prefix-list but is present in the other two to calculate the overall score. Similar to the suffix score and similarity score. To obtain the overall score for each word of the dictionary corresponding to a noisy SMS word, combine the above three score calculated as:

$$Overall_{Score}(t) = w_1 * Prefix_{Score}(t,s) + w_2 * Suffix_{Score}(t,s) + w_3 * Sim_{Score}(t,s) \quad (4.3)$$

Where t is the dictionary term, s is the SMS token, and w_1, w_2, and w_3 are the constant weights assigned to prefix score, suffix score, and similarity score and are given values of 0.25, 0.2, and 1, respectively. The values of these weights are assigned using the experimental results that are discussed in the next section.

More weightage is given to the similarity score because the similarity score considers the ratio, while the prefix and suffix score takes the number of characters matched into consideration. We sort the entries by value (score) from the hash table for the overall score and choose the top 7 results in the output list.

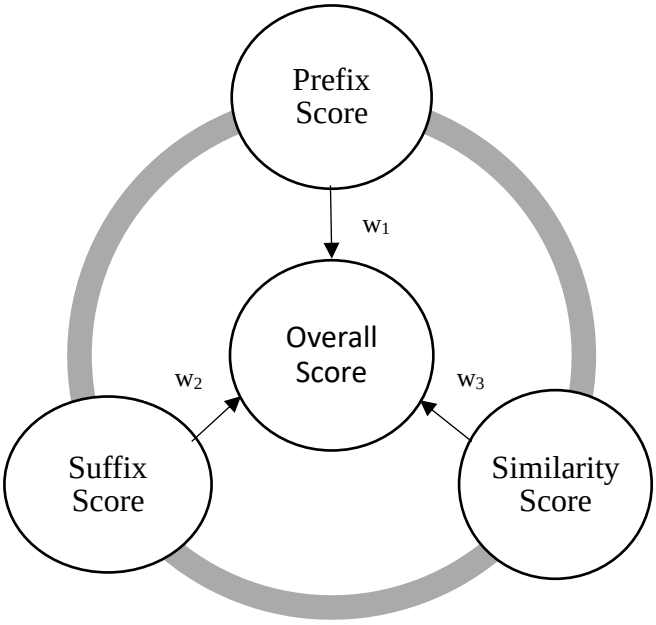

Figure 4.1. Diagrammatic Representation of Overall Score

4.3 SUMMARY OF PROPOSED ALGORITHM

Step 1. Create a list for the noisy terms and the dictionary terms.

Step 2. Create a prefix tree by inserting all dictionary terms in the ternary search tree.

Step 3. Create a suffix tree by inserting all dictionary terms in reverse order in the ternary search tree.

Step 4. For each noisy token:

a. Calculate prefix score:
   i. By finding the matching length of noisy token with dictionary term having at least two prefix characters in common in prefix tree.
   ii. Create a hash table to store these scores.

b. Take the top 7 retrievals sorted by the prefix score.

c. Calculate the suffix score:

i. By finding the matching length of noisy token with dictionary term having at least two suffix characters in common in suffix tree.
ii. Create a hash table to store these scores.

d. Take the top 7 retrievals sorted by the suffix score.

e. Calculate Similarity score:
i. By finding LCS and length of dictionary term and SMS token using (4.1).
ii. If the first character of the noisy term and dictionary term is the same, update the score using (4.2).

f. Take the top 7 retrievals sorted by similarity score.

g. Calculate overall score:
i. For words retrieved by the above three methods, calculate the overall score using (4.3).

Step 5. Return the top 7 retrievals as the best matching candidates for the noisy token.

4.5 FLOW CHART OF PROPOSED MODEL

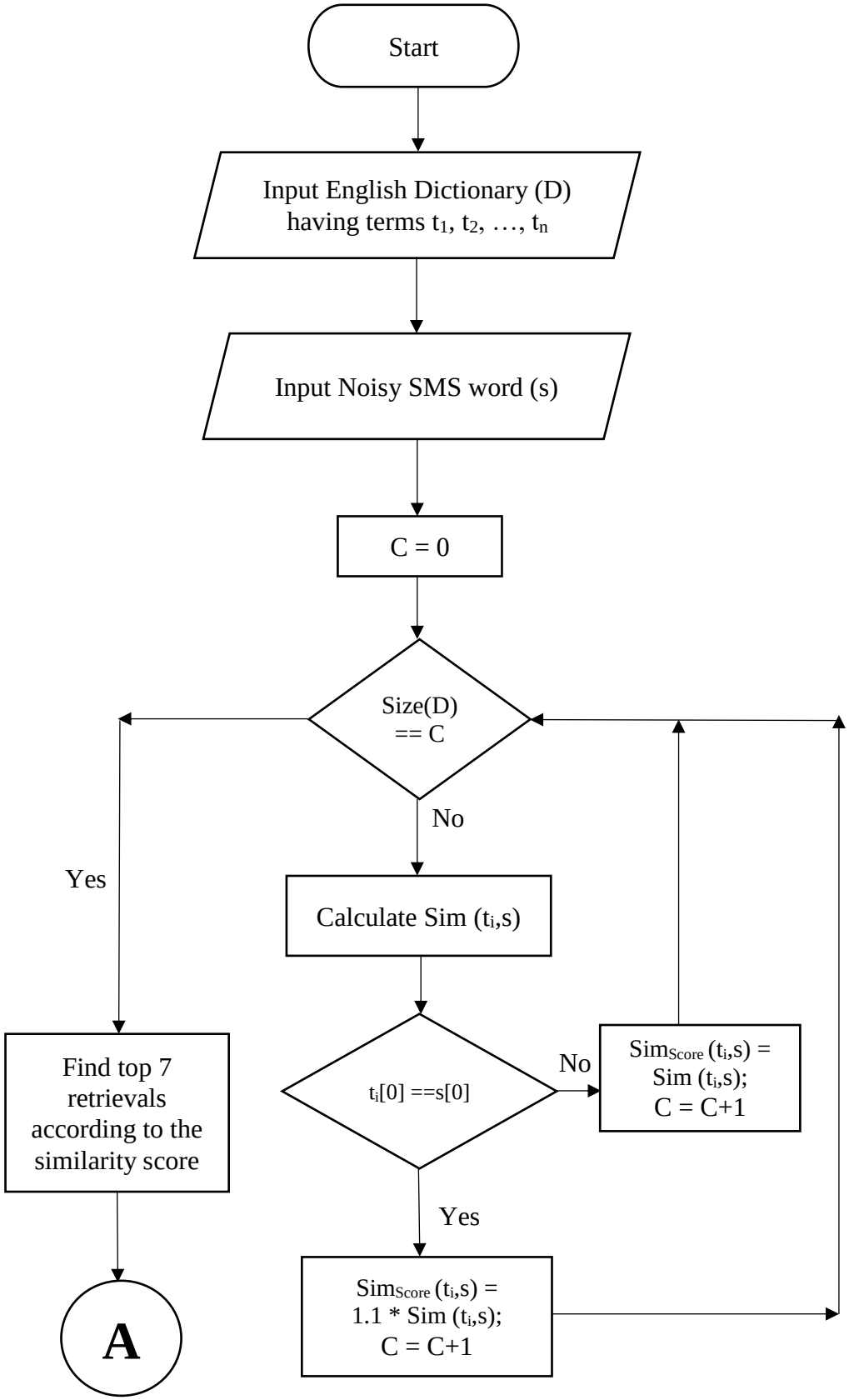

Figure 4.2. Calculation of Similarity Score

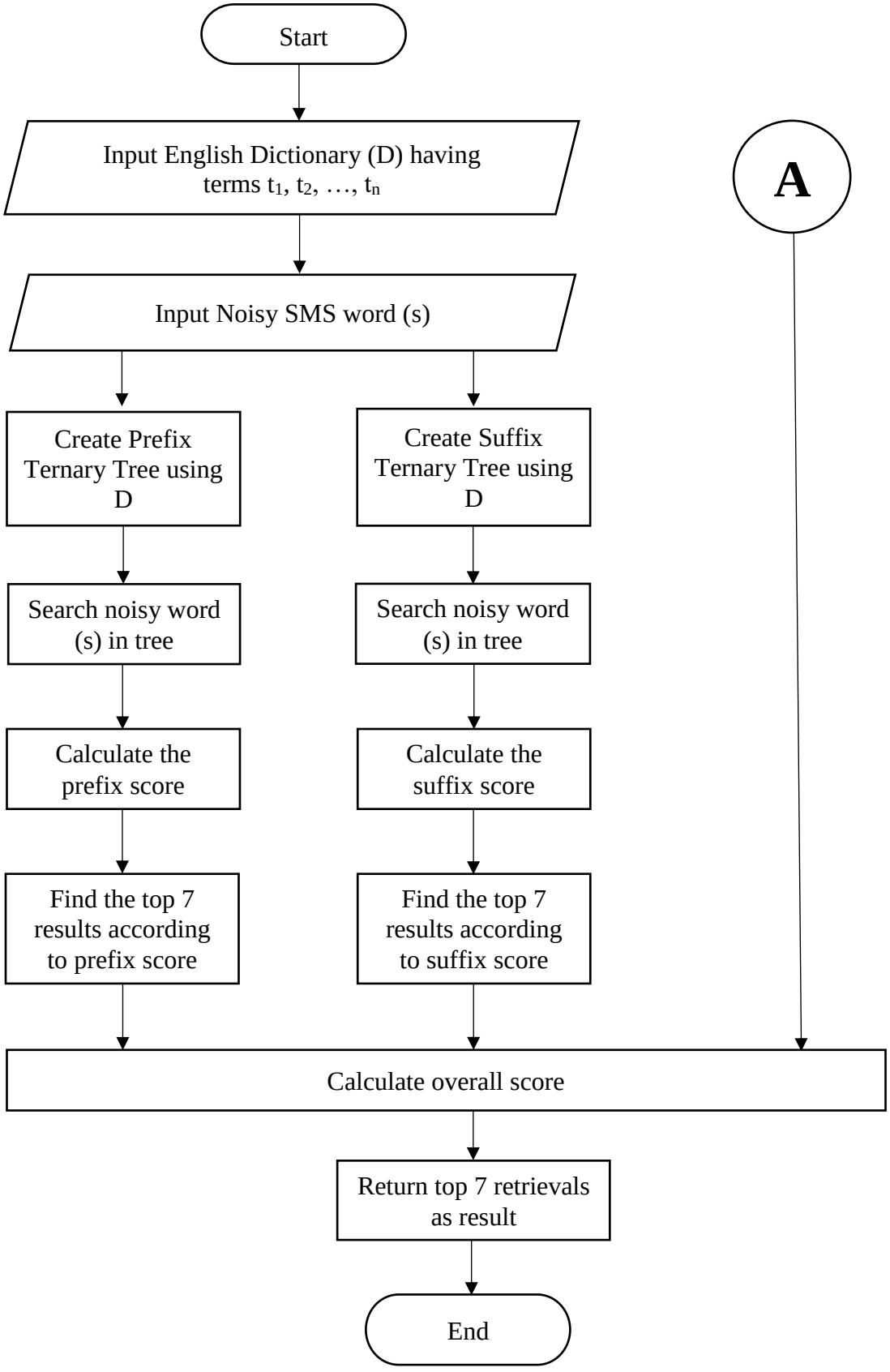

Figure 4.3. Working of the proposed model

CHAPTER 5

WORKING AND ANALYSIS

To develop an efficient system to de-noise the noisy SMS text, the ternary tree is used. Calculating prefix score based on matching prefix length and suffix score based on matching suffix length are the two main concepts used. Apart from these two, the similarity score is taken into consideration to find the best match for the noisy token. Prefix ternary tree and suffix ternary tree are created to calculate prefix and suffix scores.

5.1 PREFIX TERNARY TREE

Taking all the words of the dictionary, a prefix ternary tree is created. Then, a noisy SMS word is matched against each word in a tree, and the prefix score is calculated for dictionary word having minimum two characters matching and are stored in a hash as key-value pair where the key is dictionary term, and the value is the calculated prefix score. The prefix scores sort the hash table, and the top seven outputs are taken to calculate the overall score.

For example, if the list of dictionary term consists of words such as [“disappear”, “dismiss”, “dislike”, “damage”, “dialect”] and the noisy token is “distance”, then the calculated prefix scores are given as: { “disappear” : 3, “dismiss” : 3, “dislike” : 3, “dialect” : 2}. The prefix “dis” in noisy token “distance” matches with the dictionary term “disappear”. So, we get length of “dis”, i.e., 3 as the output. Similar is the reason for output 3 for words like “dismiss” and “dislike”. For dictionary word “dialect”, the prefix “di” matches with the noisy token “distance”. So gives output 2 for the same.

In the case of words like “decem” for “demember” which are truncated, the words in the final top 7 retrievals are because of the prefix score. For such terms, the

prefix score is dominant. The prefix score output for “decem” noisy word is shown in figure 5.1.

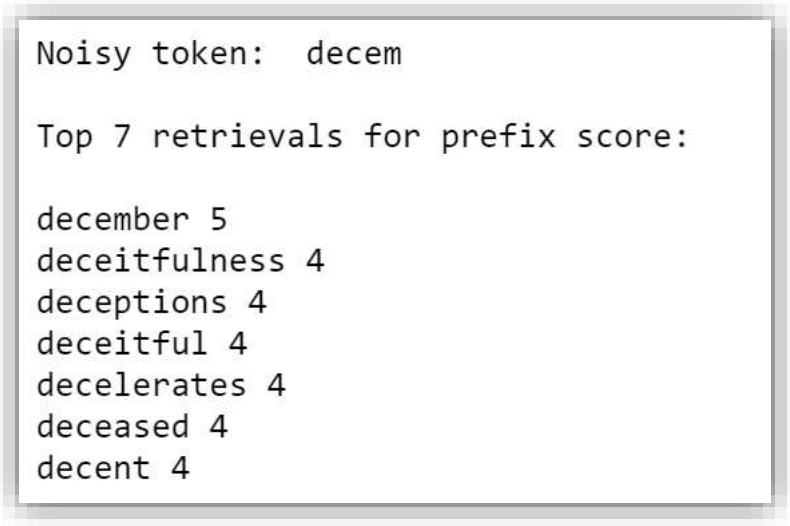

Figure 5.1. Top 7 retrievals for prefix score using the proposed model

5.2 SUFFIX TERNARY TREE

To construct a suffix ternary tree, all the words of the dictionary are reversed and inserted into the ternary tree. Then the noisy token is reversed and is then searched in the suffix ternary tree to calculate the suffix score. The reversed noisy token is matched against all words in the tree, which are the reversed dictionary words. The suffix score is calculated and stored in the hash with the dictionary term as key and suffix score as value. Creating a direct suffix tree will increase the complexity to a great extent. So, create a suffix tree by reversing the dictionary term to match the complexity with the prefix tree. Sort the hash by suffix value and take the top seven retrievals.

If the list of dictionary term consists of words such as [“helper”, “butcher”, “dancer”, “preacher”] and the noisy token is “teacher”, then the calculated suffix scores are given as: {“preacher” : 6, “butcher” : 4, “helper” : 2, “dancer” : 2}. Suffix “eacher” is common in both the noisy word “teacher” and the dictionary term “preacher”. So, the word “preacher” is given suffix score 6. In the case of a noisy word like “dfusion”, suffix score is dominant. So, giving weightage to suffix score helps in getting the correct word “diffusion” as the output. The top 7 retrievals using the suffix score is shown in figure 5.2.

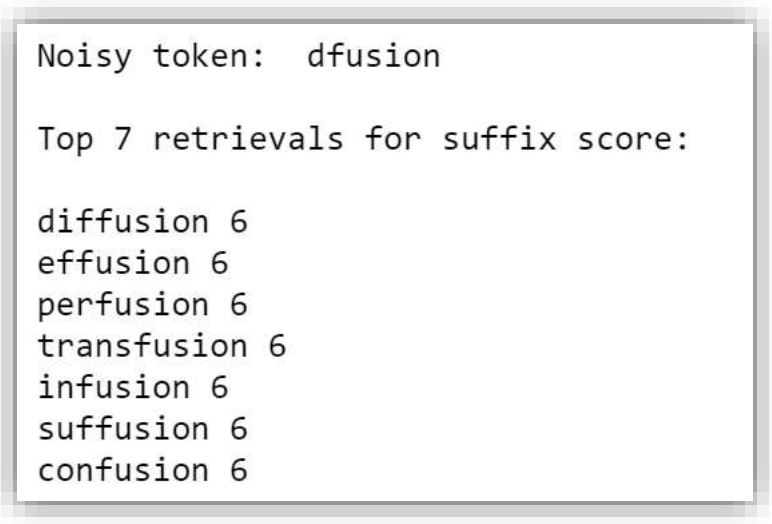

Figure 5.2. Top 7 retrievals for Suffix Score using the proposed model

5.3 SIMILARITY SCORE

The similarity score between the noisy token and the dictionary term is calculated to find the similarity. It is calculated using the longest common subsequence, which is based on the number of common characters between the two strings. A similarity score is the ratio of twice the number of common characters between the two strings to the sum of length of the two. The similarity score is calculated between the noisy word "hxgn" and the dictionary terms. The score is calculated using (4.1) and (4.2), and the results are sorted based on the values. The top seven retrievals are taken and shown in figure 5.3.

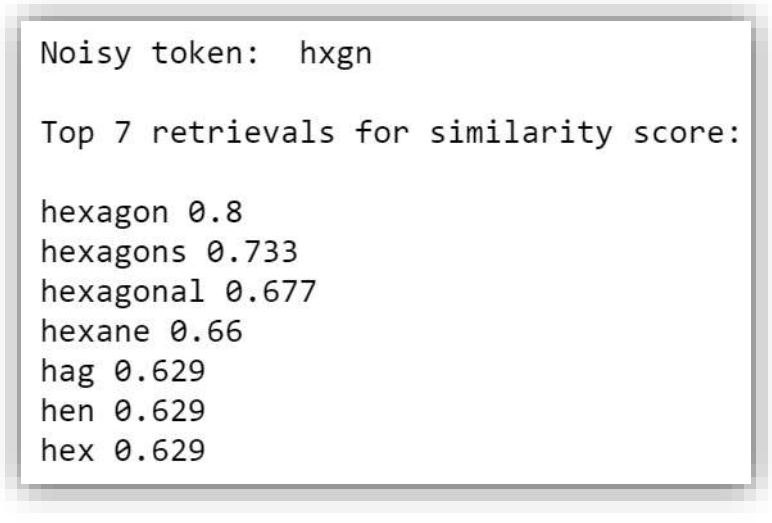

Figure 5.3. Top 7 retrievals for similarity score using proposed model

5.4 OVERALL SCORE

Until now, we have the top 7 retrievals for the prefix score, the suffix score, and the similarity score. For each of the words in the three lists, calculate the overall score using (4.3). Sorted the results by values and retrieved the top 7 outcomes. These seven words are the best candidates for the noisy token. The overall score is calculated for each of the words in prefix hash, suffix hash, and similarity table, and the top 7 results are taken after sorting it by overall score. Figure 5.4 shows the top 7 retrievals for the noisy word “thur” using Ternary Tree.

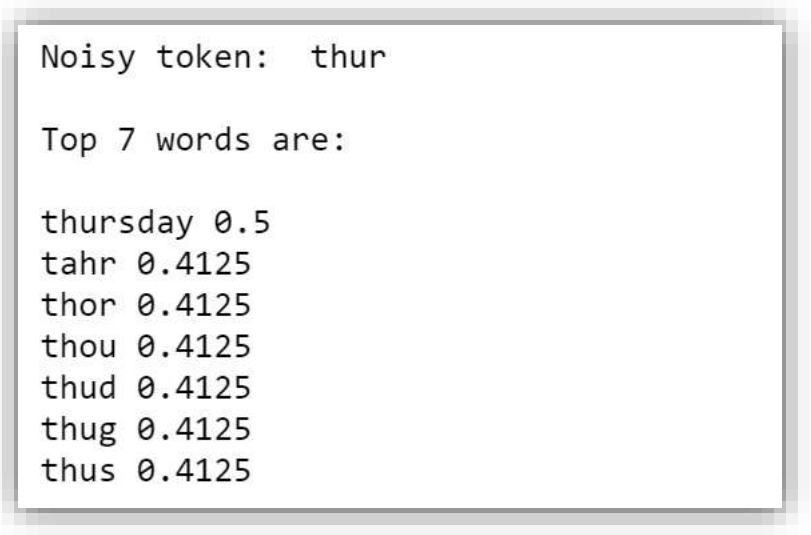

Figure 5.4. Final top 7 retrievals using the proposed model

5.5 AGARWAL’S MODEL AND KOTHARI’S MODEL

Apart from implementing the ternary search tree, two other models [3, 4] are also implemented to compare the result of the proposed model. Agarwal et al.’s model uses the trie to calculate prefix and suffix scores. Kothari et al.’s model use only a similarity calculation to de-noise the SMS token. The similarity score is calculated using (4.1). The values sort these results, and the top seven results obtained after prefix score, suffix score, similarity score, and overall score calculations are shown in figures 5.5-5.8.

```
Noisy token:  decem

Top 7 retrievals using prefix trie:

december 5
decease 4
deceased 4
deceases 4
deceit 4
deceitful 4
deceitfulness 4
```

Figure 5.5. Top 7 retrievals for Prefix Score using Agarwal's model

```
Noisy token:  dfusion

Top 7 retrievals using suffix trie:

fusion 6
confusion 6
infusion 6
diffusion 6
effusion 6
suffusion 6
perfusion 6
```

Figure 5.6. Top 7 retrievals for Suffix Score using Agarwal's model

```
Noisy token:  hxgn

Top 7 retrievals using similarity score:

hexagon 0.727
hexagons 0.667
hexagonal 0.615
hexane 0.6
oxygen 0.6
shogun 0.6
gen 0.571
```

Figure 5.7. Top 7 retrievals for Similarity Score using Agarwal's model

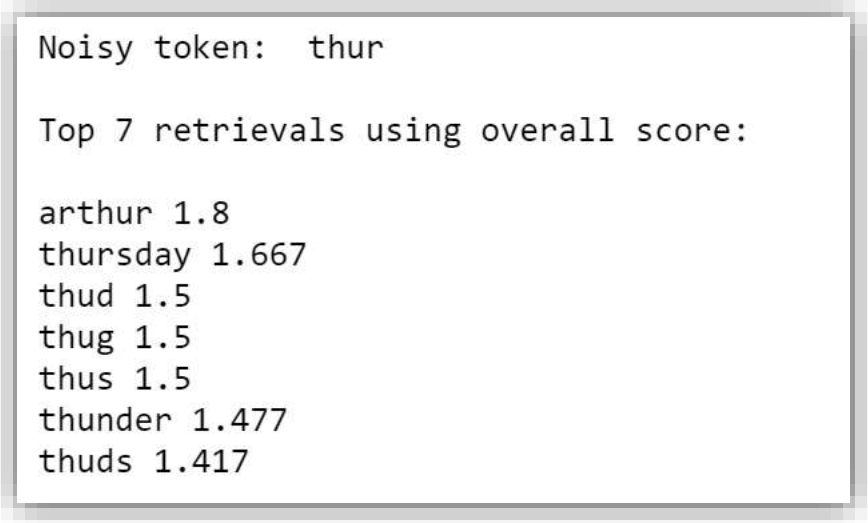

Figure 5.8. Final top 7 retrievals using Agarwal's model

Kothari et al. [4] in their model, calculated similarity score for each of the dictionary term with respect to the noisy token. The similarity score is given by (2.1). The similarity score is based on the longest common subsequence ratio and the edit distance.

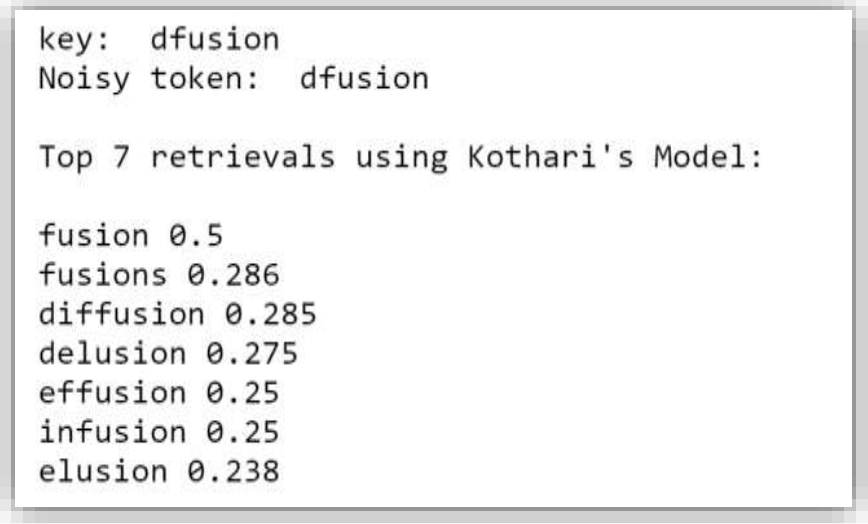

Figure 5.9. Top 7 retrievals using Kothari's model

CHAPTER 6

EXPERIMENTS AND RESULTS

The ternary tree was the key to the proposed model. Matching prefix length, matching suffix length were calculated using the ternary tree and the similarity score using the dynamic algorithm like longest common subsequence. Prefix score, suffix score, and the similarity score are the key factors to calculate the overall score to efficiently de-noise the noisy SMS token.

The model is implemented in Python 3.7 using the Anaconda platform. To verify the performance of the proposed model, it is tested on Mieliestronk's words list [11] as the dictionary. Mieliestronk's words list comprised of 58000+ English words and tested against 255 noisy words to test the performance of our system. These noisy words include the words which are misspelled, truncated, and whose vowels and repeated characters are dropped. The de-noising system is compared with Agarwal et al.'s [3] and Kothari's [4] method. To assess quantitatively, MRR (Mean Reciprocal Rank), and accuracy is considered to compare the proposed model's output with the previous ones. Accuracy is given by the number of correct output retrieved to the total number of output retrieved. MRR [12] is another evaluation measure that is used to calculate the systematic of the system by making possible response lists to the query in order of their probability of correctness. It is used to get the best relevant document. The reciprocal rank is given by $1/_{rank}$, where the first relevant document is found. Mean reciprocal rank gives the average of all the N-reciprocal ranks and is given by:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{6.1}$$

Rank = 1 is given when the correct word is found at the top of the list. Rank = 7 for the term at 7th position in the list. MRR is used because it is easy to compute and give focus on the first relevant item of the list; thus, it helps in finding the best match.

6.1 PARAMETERS DISCUSSION

There are 3 main parameters in the novel method, i.e., weight for prefix score w_1, weight for suffix score w_2, and weight for similarity score w_3. To study these parameters, different experiments were carried out, one parameter is made variable, and the other two as fixed. The results of the tests are displayed in table 6.1-6.3.

Table 6.1. Parameter w_1 study

Metric	$\boldsymbol{w_1}$**,** $\boldsymbol{w_2}$**= 0.2,** $\boldsymbol{w_3}$**= 1.1**				
	0.15	***0.20***	***0.25***	***0.30***	***0.35***
Accuracy	94.686	94.690	**95.294**	90.196	82.352
MRR	0.785	**0.801**	0.789	0.757	0.702

Table 6.2. Parameter w_2 study

Metric	$\boldsymbol{w_1 = 0.25}$**,** $\boldsymbol{w_2}$**,** $\boldsymbol{w_3}$**= 1.1**				
	0.15	***0.20***	***0.25***	***0.30***	***0.35***
Accuracy	94.509	**95.294**	92.549	91.372	84.705
MRR	0.788	**0.789**	0.750	0.759	0.681

Table 6.3. Parameter w_3 study

Metric	$\boldsymbol{w_1 = 0.25}$**,** $\boldsymbol{w_2}$**= 0.20,** $\boldsymbol{w_3}$		
	1.0	***1.0 + 0.10 If 1st char matches***	***1.0+ 0.20 If 1st char matches***
Accuracy	93.333	**95.294**	94.294
MRR	0.754	**0.789**	0.782

The model is run for different weightage for prefix, suffix, and similarity score and found that the best results are obtained when weightage for prefix score w_1 = 0.25, suffix score w_2 = 0.20, and a large value for similarity score w_3 = 1.1.

6.2 RANKED LIST RETRIEVAL

The results of our model are compared with the previous model [3] that uses the Trie data structure to calculate the prefix and suffix length and the model [4] that uses only similarity scores to find the correct candidate for the noisy token. Agarwal et al.'s [3] model uses the similarity score directly without considering whether the first character matches or not. The papers [3, 4] are implemented, and their results are compared with our proposed method. The same dataset is used to compare all the three approaches. The output of the model [4], model [3], and our proposed model for a few noisy SMS tokens are shown in table 6.4-6.9. These outputs are prioritized based on the top seven retrievals obtained from the respective overall score of the three models.

Table 6.4. Top seven words for the noisy SMS word “thur”

SMS word: thur, Correct word: thursday		
Kothari’s model [4]	***Agarwal’s Model [3]***	***Our Proposed Model***
'thor'	'arthur'	**'thursday'**
'thud'	**'thursday'**	'tahr'
'thug'	'thud'	'thor'
'thus'	'thug'	'thou'
'tour'	'thus'	'thud'
'tahr'	'thunder'	'thug'
'thou'	'thuds'	'thus'

Table 6.5. Top seven words for the noisy SMS word “hxgn”

SMS word: hxgn, Correct word: hexagon		
Kothari’s Model [4]	***Agarwal’s Model [3]***	***Our Proposed Model***
'hag'	'sign'	**'hexagon'**
'hen'	'align'	'hexagons'
'hog'	'feign'	'hexagonal'
'hug'	'reign'	'hexane'
'hags'	'malign'	'hag'
'hewn'	'assign'	'hen'
'hex'	'ensign'	'hex'

Table 6.6. Top seven words for the noisy SMS word “rmv”

SMS word: rmv, Correct word: remove		
Kothari’s model [4]	***Agarwal’s Model [3]***	***Our Proposed Model***
'rev'	'arm'	'ram'
'ram'	'ram'	'rem'
'rem'	'rem'	**'remove'**
'rim'	**'remove'**	'rev'
'rom'	'rev'	'rim'
'rum'	'rim'	'rom'
'arm'	'rom'	'rum'

Table 6.7. Top seven words for the noisy SMS word “dfusion”

SMS word: dfusion, Correct word: diffusion		
Kothari’s Model [4]	***Agarwal’s Model [3]***	***Our Proposed Model***
'fusion'	'fusion'	**'diffusion'**
'fusions'	**'diffusion'**	'infusion'
'diffusion'	'infusion'	'profusion'
'delusion'	'effusion'	'transfusion'
'effusion'	'confusion'	'effusion'
'infusion'	'suffusion'	'suffusion'
'elusion'	'perfusion'	'confusion'

Table 6.8. Top seven words for the noisy SMS word “sychtrst”

SMS word: sychtrst, Correct word: psychiatrist		
Kothari’s Model [4]	***Agarwal’s Model [3]***	***Our Proposed Model***
'schist'	'syndicalist'	**'psychiatrist'**
'scars'	'symbolist'	'schist'
'scots'	'sycophants'	'psychiatrists'
'sects'	'sycamores'	'sycophancy'
'shirt'	'sycophant'	'sycophant'
'short'	'thirst'	'sycamores'
'strut'	**'psychiatrist'**	'sycophants'

Table 6.9. Top seven words for the noisy SMS word “decem”

SMS word: decem, Correct word: december		
Kothari’s Model [4]	***Agarwal’s Model [3]***	***Our Proposed Model***
'deem'	**'december'**	**'december'**
'dee'	'deem'	'deem'
'deems'	'deceit'	'deceived'
'deuce'	'decent'	'deceases'
'dace'	'decease'	'deceptions'
'deck'	'deceits'	'decelerate'
'deco'	'deceive'	'decentralised'

As we can see from the comparison table 6.4-6.9, the assumption of increasing the similarity score by 0.1 times if the first character of noisy term and dictionary term are the same, outperforms the Agarwal et al.’s results. In table 6.4, the first character ‘t’ in “thur” matches with “thursday” but not with “arther”, so, the weightage of “thursday” is increased, and the proposed model gives it as the 1st retrieval. Similar to the word “hexagon” in table 6.5. Similarly, the correct terms “remove” in table 6.6 and “diffusion” in table 6.7 move one position up in our model due to increasing the similarity if 1st character matches in noisy SMS token and the dictionary word.

In table 6.8, we can see the better result because of decreasing the weight of the suffix score by 0.25 to 0.20. Prefix sore is also an essential factor. It works well for the truncated words. This can be seen in table 6.9, as for ‘decem’ noisy term, prefix score is the most dominant factor compared to other, and as in both Agarwal’s [3] method and our method, prefix score is given similar weight, i.e., 0.25. Both ways give the same position of ‘December’ word.

6.3 MRR AND ACCURACY

The MRR is introduced to check the effectiveness of the different models. A high value of MRR represents that the correct word corresponding to the noisy word is at the top level of the output list compare to the low MRR. High rank is given if the relevant

document is found the first time. Rank = 0, if relevant document is not found. MRR = 0.75 indicates that most of the results are obtained either at 1st or at 2nd position.

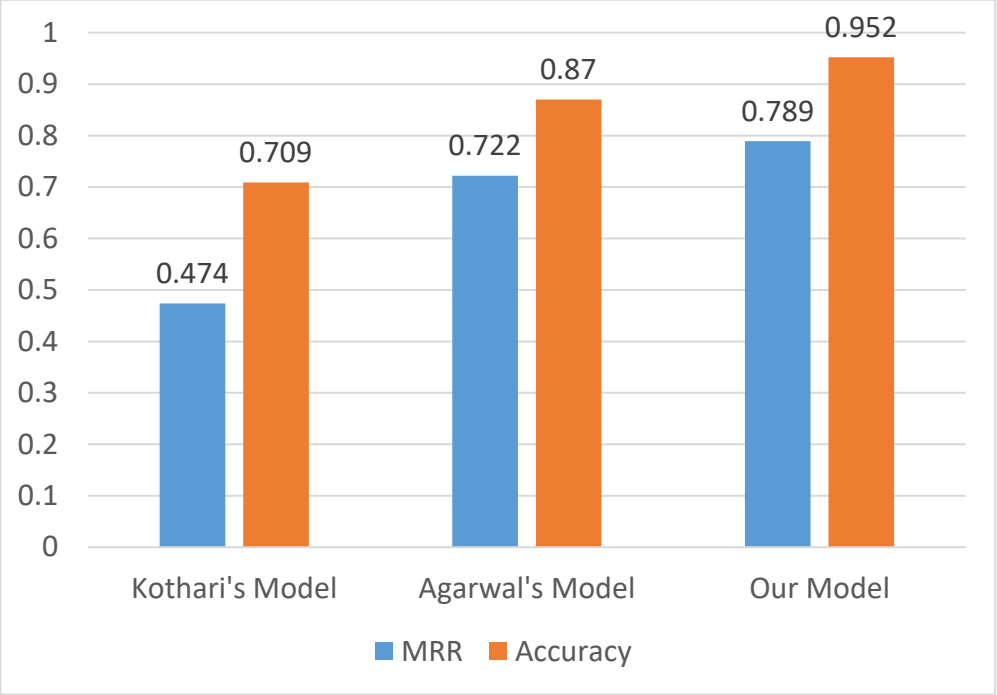

Figure 6.1. Mean Reciprocal Rank and Accuracy of different models

Figure 6.1 shows the different MRR values and accuracy of the three different models. MRR and Accuracy vary with the change in the noisy data. Still, since all the three models have experimented with the same dataset, we see that the model achieves the highest value of MRR and Accuracy compared to others every time, which justifies our proposed novel model's better performance.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

Mobile phones are the most widely used device due to their low cost, portability, and easy accessibility, and the SMS facility is available on every mobile device. Therefore, FAQ retrieval based on SMS is the area that is very useful for people to save their cost, time, and space. It prevents them from the hassle of going to a particular place and also to avoid waiting in a queue. It can be useful for any type of information retrieval for railway reservation, bills, railway inquiry, advertisement, banking, marketing, healthcare, health reports, medicines prescription, and other kinds of information access. Natural language processing, particularly text retrieval, is used to de-noise the words written in SMS language, i.e., the words which are truncated, abbreviated, misspelled, etc.

Developing such an automated system to de-noise an SMS text has always been a difficult task. The proposed model tries to efficiently remove noise from the SMS token and find the correct candidate for the noisy token from the right English words list. The model achieves a good result with the accuracy of 95.294% and the MRR of 0.789, which is far better than the previous models for de-noising of noisy SMS token. Different weights for the parameters are measured and found the values that give the best result. Results are compared with the previous approaches, and the proposed model outperforms the previous models. The ternary tree is used for calculating matching prefix length and matching suffix length as it takes less space to store the dictionary keys than the trie data structure and takes less time to search than the hash. Also, giving more weightage to the word having the first character was the right choice as it improves the result to a great extent.

As a part of future work, work can be done to further improve the Accuracy and MRR of the system by finding the new methodology. Also, it can be extended further to make the system work for the entire question using a different approach to get the best MRR score possible. As still today, the part of the country is illiterate, so instead of SMS text, the work can be done on spoken queries so that the illiterate person can get benefited from the system. A single system can be made for all the Indian languages so that people from all over the country can use the system.

REFERENCES

[1] P. Lalitha, "The impact of 'sms' language on standard english", IOSR Journal Of Humanities And Social Science, vol 1, 2013, pp. 50-51.

[2] L. V. Subramaniam, S. Roy, T. A. Faruquie, S. Negi, "A survey of types of text noise and techniques to handle noisy text", In Proceedings of AND, Barcelona, Spain, 2009, pp. 115–122.

[3] A. Agarwal, B. Gupta, G. Bhatt, and A. Mittal, "Construction of a Semi-Automated Model for FAQ Retrieval via Short Message Service". Proc of the 7th Forum for Information Retrieval Evaluation (FIRE2015) (2015), 35–38.

[4] G. Kothari, S. Negi, T. A. Faruquie, V. T. Chakaravarthy, and L. V. Subramaniam, "Sms based interface for faq retrieval", In the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, 2009, pp. 852-860.

[5] E. Sneiders, "Automated question answering using question templates that cover the conceptual model of the database", Natural Language Processing and Information Systems, Springer Berlin Heidelberg, 2002, pp. 235-239.

[6] D. Shaikh, M. Jain, M. Rawat, R. R. Shah, and M. Kumar, "Improving accuracy of sms based faq retrieval system", In Multilingual Information Access in South Asian Languages, Springer, 2013, pp. 142-156.

[7] D. Contractor, T. A. Faruquie, and L. V. Subramaniam, "Unsupervised cleansing of noisy text", In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, 2010, August, pp. 189-196.

[8] N. Shivhre, "Sms based faq retrieval", In Multilingual Information Access in South Asian Languages, Springer Berlin Heidelberg, 2013, pp. 131-141.

[9] S. Mhaisale, S. Patil, and K. Mahamuni, "Weighted edit distance based faq retrieval using noisy queries", Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation (FIRE2012&2013) (2013), 8:1–8:4.

[10] R. Batra, S. Sharma, A. Shrivastav, P. Goyal, “Efficiently de-noising sms text for faq retrieval”, IEEE International Conference on Data Mining and Intelligent Computing, Delhi, Sep. 2014.

[11] CornCorb. The Corncob list of more than 58000 English words. Available: http://www.mieliestronk.com/wordlist.htm

[12] J. Waghmare, and M. A. Potey., “Survey of sms based faq retrieval systems”, International Journal Of Engineering And Computer Science 4, 2 (2015), 10259–10263.

[13] A. O. Adesina, K. K. Agbele, A. P. Abidoye, and H. O. Nyongesa, “Text messaging and retrieval techniques for a mobile health information system”, Journal of Information Science, vol. 40, no. 6, 2014, pp. 736–748.

[14] Forum for information retrieval evaluation (FIRE) 2013 shared task detailed description: FAQ retrieval using noisy queries, http://www.isical.ac.in/fire/faq-retrieval/2013/faq-retrieval.html (URL verified 19 Nov. 2013).

[15] http://www.cfilt.iitb.ac.in/resources/surveys/Swapnil-Cross lingual-Information-Retrieval.pdf

LIST OF PUBLICATIONS

[1] Manoj Kumar, Vipra Bhatt, "Denoising SMS Text using Ternary Tree for FAQ Retrieval". Accepted at the **International Journal of Advanced Science and Technology (IJAST).**

Indexed by Scopus, EBSCO, ProQuest, ULRICH, J-Gate, OAJI**.**
Paper Id: IJAST_06_2020_507

Abstract- With the increase in mobile technology, everyone is using mobile phones for various purposes, from only making calls to using it for accessing the internet, gaming, etc. Internet service is not available on every mobile phone, but the facility of Short Messaging Services (SMS) is on every phone. So, many people choose SMS for information retrieval. The SMS can be used to get a quick response in various fields such as healthcare, railways, insurance, banking and travel. But the SMS texts are made noisy by intentional and non-intentional errors, and there is a need to find the correct word to substitute them. This paper proposed efficiently denoising the SMS term by finding matching prefix and suffix length using a ternary search tree and calculating similarity score between the noisy token and the correct English words using Longest Common Subsequence (LCS). We see that the ternary tree is space-efficient compared to the trie data structure used in the previous approach to efficiently denoise the SMS token. It also helps in completing the task in less time compared to using a hash table. We compute the overall score by combining the prefix matching length, suffix matching length, and similarity score. We experimented using some noisy SMS words generally used by the people and compare our model with some previous models. Our model outperforms all of these earlier approaches.

[2] Manoj Kumar, Vipra Bhatt, “A Review on Monolingual and Cross lingual FAQ Retrieval System”. Accepted and presented at the **5th International Conference on Communication and Electronics Systems (ICCES 2020)**.
Indexed by Scopus.
Paper Id: ICCES253

Abstract- With the growth of movable communication, cell phones have become the most widely used device for communication. Even mobile users are increasing at an amazing rate. Short messaging service is most convenient way to give access to information to the community over the world. Different automated FAQ retrieval systems are developed in which the SMS query is given by user and result is retrieved from the database of FAQs. FAQ database consists of questions and answers already designed by the experts. SMS query is matched from the set of questions from the database, and the question with highest score is selected. This paper reviews various approaches for FAQ retrieval based on SMS. Basically, we provide a review on monolingual and cross lingual FAQ retrieval and performance for various approaches are compared.