# CHAPTER 1

# INTRODUCTION

Nowadays information retrieval is very frequent and active process for every other person. There are many modes for accessing information such as telephones, mobiles, internet etc. Advancement of mobile and internet technology gave rise to mobile and web search retrieval system respectively which compliments to traditional retrieval system, it follows completely different execution process [1]. Telecommunication and mobile growth made finding information easy and this is done via short text that is also known as SMS (Short message service) or texting. Mobiles are everybody's necessity. Everywhere mobile networks are available. It's really popular among young generation. There are approx. 811 million subscriber of mobile in India [4]. The number of messages sent in 2010 was 6.9 trillion and in 2011 it reached over 8 trillion [5]. SMSes technology has achieved heights in wireless world. This has restriction for number of characters for communication because of bandwidth, writing skill of text and memory [2]. SMS does not abide by spelling standards, pronunciations and conventional grammar. Intentionally words are compressed by phonetic spelling, transliteration and abbreviations are used. It just deals with text data. This fortify service provider to develop information retrieval services on the basis of SMS technology.

For example, SMS translation from one language to recipient mother language, where the sender doesn’t know recipient language and it’s automatically converted [3]. There are services such as “ChaCha” [6], “AQA 63336” [8] by Issuebits Ltd, “Texperts” [9] by Number UK Ltd. and GTIP [10] by AlienPant Ltd., in which the user can send query to service centre but it is handled by a human expert, it’s not an automatic response service. Here user is free to use SMS language without the need of any specified format for asking query because a human expert is sitting there to understand it and to respond it. But limited number of queries can be handled at a time because of limited experts, so it’s not perfect way of handling queries. If it becomes automated system to give response to query then the service will become more efficient. This automated SMS based question-answering system has been proposed in “SMS based interface for FAQ retrieval” by Venkatesan T. Chakaravarthy et al. in Aug’09, IBM India Research lab [7]. Transliterations, short form, spelling mistakes, homophonic words, abbreviations and different noise in SMS can be handled by matching SMS query with FAQ database. User sends query to system in form of text message and get reply. In this user need not to have multimedia or high-tech cell phones, this can be done by basic handset which doesn’t even have internet facility on it. Generally search is done on computers but it’s not necessary that computer system is always with you. Mobile phones are handy and easy to carry. To check query from FAQ database is bit tedious for user, so to make convenient to user SMS based FAQ retrieval came into the light. There can be one or more response to the query. Till now we discussed basic for SMS based FAQ retrieval, further in this chapter motivation, related work and research objective has been discussed.

## 1.1. MOTIVATION OF THE WORK

Mobile technology gave contribution to the progress of media of communication for example: chats, emails and short message services (SMS). The Popularity, utility and simplicity of SMSes is encouraging people to access information via SMSes, accessing information via internet creates hassle, it’s not necessary that internet connection is always available. So user can clarify their query, make complaint and get updates of result etc by sending SMS. Accessing information in such a manner makes information access very economic and easy for everybody from rural to metro city people. Portability of mobile is easier and mobile networks are present everywhere, so by clubbing these two gave motivation to information retrieval using mobile devices. In today’s date many such services are in use such as activating/deactivating any service, voting, e-commerce and many more. Now we are having high resolution mobile phones, full keypads and web browser in it but still at many places people are using basic mobile phones with limited keypad and without web browser, so this type of information retrieval is easy to access and encouraged providers to develop SMS related services. Different and frequently used such SMS based FAQ retrieval services are:

- Organization related Enquiry system
- Activating/Deactivating services
- Small domain search engines
- Voting through SMS
- Customer care facilities provided by software, hardware and telecom services
- FAQ support for various organization automatically.

## 1.2. RESEARCH OBJECTIVE

“FAQ retrieval” means there is corpora of frequently asked questions, and user sends a query in SMS language to retrieve some information. Such systems finds best match from FAQ corpora for given user defined query written in SMS language. The main problem in SMS language is the noise associated with it. Spelling mistakes, transliteration, phonetic spellings, abbreviations and short forms create difficulties in string matching, when query is matched to FAQ dataset and the main task is to find FAQ from FAQ corpus which is matching best with input SMS query.

In this thesis, objective is **to present atypical approach for getting best matched FAQ from FAQ dataset with respect to SMS query,using three similarity measures: Soundex similarity, Jaccard’s similarity and Cosine similarity. These similarity measures has been used in hybridization form** i.e. soundex similarity is used for token by token matching or conversion of SMS keywords into English language. Rest of two, Cosine similarity and Jaccard’s similarity is to get matching percentage for sentence of FAQ corpus with respect to sentence of SMS user query. These similarity measures work on sentence by sentence matching.

## 1.3. RELATED WORK

The research on SMS based question- answering system has been started in last few years. As user uses SMS noisy language i.e. shortcuts and phonetic spellings, this matching task becomes more tedious. SMS based QA system is in which the user can ask or send a query in

SMS language, which may contain noise in it. System contains FAQ database in which frequently asked questions are present. For handling noise in query, a formulation of query similarity over FAQ questions is done as combinatorial search. A search algorithm that can handle semantic variation and does not require SMS normalization has been proposed [7].

In SMS based FAQ system for calculating the scores of questions N-grams counts are used, which improves efficiency of system. Length based score and proximity based score takes length and tokens into consideration for calculating score of every question in FAQ database [11]. By clustering previous query log, recovering FAQ [12]. [7] Is further entailed by latent semantic analysis and latent term weight [13].

Bilingual statistical dictionary is made for expanding queries and for alignment of questions and answers of FAQ database, machine translation techniques are used in information retrieval system [14].

FAQ retrieval based on domain using "independent aspect". K-means algorithm and latent semantic analysis used for clustering question-answer. Probabilistic mixture model so that retrieval can be considered as maximum likelihood estimation problem [15].

## 1.4. SCOPE OF THE WORK

In recent time SMS based QA servicing is rising but earlier very little work has been done on SMS based system for different topics because of lack of knowledge of well define business cases. However, automation of such systems is challenge due to noise in the query. Tremendous growth in popularity of affordable low-end cell phones throughout the world has conceived a grand market for mobile information services. SMS has become primary data transport for mobile users in many parts of world, who are using low-end mobile phones.

In this thesis, Automatic SMS based search response system is proposed from FAQ dataset to improve the information retrieval model. Three different similarity measures have been used, which can provide best search responses for SMS queries. These Similarity measures make matching of query and FAQ database scalable and improve accuracy. Various other techniques are consolidated to SMS based FAQ retrieval system. It can be further extended to use in different areas conveniently and efficiently. There is a vast scope of improving the accuracy of the system so that user get correct answer for query and performance of the system can be improved to get answer for query in real time.

## 1.5. ORGAINIZATION OF THESIS

This chapter initially discusses the SMS based FAQ retrieval systems. In same motivation and objective of thesis is discussed. Brief of prior work that have been done and scope of doing work in the same field. It also includes SMS processing difficulties associated and techniques that have been proposed to resolve those difficulties.

***Chapter 2*** gives a comprehensive study of SMS based QA system; different types of approaches to implement SMS based QA system. Benefits and importance of SMS based QA system is described. In processing SMS queries and process of implementing the system challenges faced are described. Also includes problem formulation and system implementation details.

***Chapter 3*** focuses on proposed framework for SMS based FAQ retrieval using hybrid similarity measure. One similarity algorithm is applied to match token by token and other two similarity algorithms uses previous similarity and applied over it, for giving concept of hybrid similarity measures.

***Chapter 4*** gives the implementation details and experimental setups. Corpora is defined which is used while experimental result.

***Chapter 5*** presents conclusion of thesis and some suggestions future work.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1. BASIC CONCEPTS OF SMS BASED QA SYSTEM

### 2.1.1. SMS BASED QUESTION ANSWERING SYSTEM

In this globalization, retrieval of information has become an essential part of everybody's life. Due to that, approaches which make information retrieval easier or convenient became interesting research area. Everybody wants easy and better way to retrieve information. Information access can be achieved by several means such as telephones, mobile phones, internet etc. Because of swift development in mobile communication, cell phones became important part of our day to day life. Nowadays, Short Messaging Service (SMS) is very frequent and popular service for accessing information. There are approx. 811 million mobile subscribers in India and it still growing rapidly [4]. So taking this in consideration, it can be said that for accessing information SMS based QA system can be simple and economical option for mobile users. On mobile phones most crowd pleasing data application is texting messages. After voice calls and e-mails, texting messages is most preferred medium of communication, as the cost of sending messages and mobile phone are very reasonable accordingly. The number of messages sent in 2010 was 6.9 trillion and in 2011 it reached over 8 trillion [5]. This ease of use and popularity encouraged the information providers to provide information retrieval services through mobile phones. Now SMSes are also used for polling, enquiry, payment, advertisement, marketing, banking etc not just for personal communication. Thus, SMS based QA system can be used for accessing information with the help of questions and answers using SMS services on mobile phones.

### 2.1.2. TECHNIQUES FOR SMS BASED QA SYSTEM

These are basically data mining techniques, which are used for SMS based QA system discussed as follows:

#### 2.1.2.1. Human Intervention Based System:

In this system, the user sends query in natural language. This query is reached to service centre where human agents are present, they understands the query, find answer for that and reply to the user. Human intervention based system involves human community to answer the queries sent by user. Such systems generally suggest similar questions resolved in the past, it is a interesting part of the system. There are few systems in which human experts answer questions in a time efficient manner e.g. Askme [16] and Chacha [6]. Recently some businesses have allowed users to send query through SMS in SMS language. In many service centres the facility is provided to SMS their requests or complaints then these complaints or queries are handled or understood by human experts and responds to users accordingly e.g. “AQA 63336”

[8] by Issuebits Ltd, “ChaCha” [6], “Texperts” [9] by number UK Ltd and GTIP [10] by Alienpant Ltd.

#### 2.1.2.2. Natural Language Processing Based System

In natural language processing systems user sends query via SMS in natural language. At server’s side, system by using natural language processing techniques the query is evaluated and processed, produce answer and finally sent to the user. But the problem is that every time the natural language processing algorithm’s complexity is used. As SMS language contains abbreviations, misspellings, transliterations, omissions and phonetic substitutions, it may create difficulty for system to successfully evaluate and understand the query. So it is a tedious task to build such automated system around SMS technologies. Such service is used in access to yellow page services [19].

#### 2.1.2.3. Information Retrieval Based System

As the name indicates the question is treated as information retrieval query, user wants to access some information. Pre-designated format of short message can be sent to get specific information. Such techniques are used in short code services. For example to retrieve your current account balance, the user has to type ACCBAL NO. XXXXXXXX. This system deals question answering as an information retrieval problem. Disadvantage of this technique is user has to remember the specific code related to every service. This puts unnecessarily constraint on user who finds SMS language Smooth for such services. Other services are Google SMS, where user can use SMS enabled phone without requiring them to upgrade their hand set and subscribe to mobile data services for searching information [18].

#### 2.1.2.4. Frequently Asked Question Based System:

In such systems, FAQ gives already available set of questions and answers for using it as database for system. Here database maintains all possible queries. When system receives a query in the form of SMS sent by user, it scrolls database for that query or any other query which is near to it and answer of that query is sent back to user, when suitable match is found. Its basic aim is to search nearest match to get apt answer. For example, system has a database of frequently asked question answers. In FAQ database the questions and answers are structured by an expert. For the user query, system just has to find the appropriately matching questions.

## 2.2. SMS based FAQ retrieval system

### 2.2.1. BASELINE

This thesis has practiced in area of SMS based FAQ retrieval. In such system, user sends query written in SMS language and then system find a match for query from the set of frequently asked question. Queries written in texting language have a problem that it contains lot of noise because of abbreviations, short forms, spelling mistakes, transliterations and phonetic spelling etc. It makes processing SMS difficult. System has to find best matched FAQ from FAQ database with respect to user SMS query with preciseness. System has corpora of FAQ can be of various domain or single domain related question answers. Corpus of question answer in database is represented by Q and SMS query is represented by S. Aim for system is to find best match for S from set of Q.

### 2.2.2. TEXT NOISE

As discussed earlier, SMS became very popular and frequently used service. Unlike normal text SMSes are not abide by spelling standards, conventional grammar and punctuations. Intentionally words are compressed by abbreviations, non standard spellings and phonetic transliterations. Such abbreviations, phonetic spellings, speling mistakes and transliterations inconsistencies in SMS query are known as SMS noise or text noise. Basic question-answering systems are free from such errors. To build an automated system with SMS technology and identifying the best matching question to retrieve the relevant answer is difficult task. High level of noise makes these problems difficult [20] [21] [7].

Few examples of SMS noise, which is faced during day to day life. Given problems pertaining in the SMS make it very noisy. This makes SMS processing a tedious task:

- Spoken words are substituted with actual spelling of word. This is also known as phonetic substitution. For example, “pasprt” for passport in passport domain.
- In different domain, few abbreviations are present related to that domain. For example, in passport domain ECR(emigration clearance), PCC(police clearance certificate) etc are used.
- Commonly observed patterns include deletion of vowels, addition of repeated character and truncation. For example, “ttkal” after removing vowels for tatkal service in passport domain, “sooo” repeating characters and “asst” after truncating.

- SMS query sometimes give keywords and miss the prepositions, conjunctions and other words which connect the key words. These missing words in the sentences are due to the limitation of text message. For example, "aply pasprt" instead of procedure of applying for passport, etc.

Categories of SMS based FAQ retrieval on the basis of language of given SMS query and FAQ database, system has to find best matched FAQ for SMS query:

- Mono-lingual Retrieval: Retrieval in which FAQ corpus and SMS queries are expressed in the same language.
- Cross-lingual Retrieval: Retrieval in which system has to find matching FAQ in a language different from the SMS queries language. For example, FAQs are written in Hindi and SMS queries are in English.
- Multi-lingual Retrieval: System has to find matching FAQs for SMS queries where both can be in any language. For example, FAQs are written in any language e.g. Hindi, English, Marathi etc and SMS queries may or may not belong to this set.

In this thesis Mono-lingual SMS based FAQ retrieval is used, as the FAQs and given SMS queries both are in same language.

### 2.2.3. SEARCH ALGORITM

Occurrence of the problems that are believed to be hard in general perspective, by search algorithm it can be efficiently exploring the usually large solution space of these instances. Combinatorial search studies such search algorithms. By reduction in effective size of the search space or by employing heuristics, combinatorial search algorithm can attain this efficiency. Many algorithms only returns the best solution but some algorithms assured to find the optimal solution found in the part of the state space that was observed. NP- hard problems comes under the combinatorial search algorithms. To boost combinatorial search, study of computational complexity theory is done. Finding moves in games with a game tree, e.g. chess. It is also include solving the eight queen's puzzle. These problems are not believed to be efficiently solvable, generally. But some instances of such problem can be solved efficiently as suggested by various approximations of complexity theory. Such instances often have important practical ramifications. These algorithm are generally implemented in an expressive declarative and in an efficient imperative programming language such as Prolog, or by functional programming language such as Haskell, or multi- paradigm language such as LISP [23]. In this work, search space for matching query in the FAQ database and SMS query is large. Naive algorithm is employed by combinatorial search technique to reduce the search space of finding the maximum scoring question.

## 2.3. PROBLEM FORMULATION

Objective of this heading is to define distinct functions and phases used in the overall system. Pre-processing functionalities which has to be processed before the actual proposed work is done. The database is pre-processed in order to make system processing fast at the time of actual computation. Various algorithms that have been used for the basic module of the work are discussed.

### 2.3.1. XML DATABASE

In the system, the database that has been used is in XML format. In wide variety of fields XML has emerged as a powerful format for representing data. XML is more forgiving and flexible for changes than other formats. It has hierarchical structure that can be used to virtually model any type of data. It presents number of opportunities for data storage.

To design specifically to manage XML content, a new class of database has emerged. XML databases are typically called "XML Native Database" [24], they incorporate functionality that highly improves the searching, management and manipulation of XML to develop the most effective XML data management solution. It is eminently true with very large collections of documents, very large documents and applications where complex document need complex parsing, manipulation, and querying. So All the questions and answers of the FAQ database are stored in XML form, where tag for questions is <question> and tag for answers is </ans>. Whole database is stored in XML format. It is easy to access and provide functionality.

### 2.3.2. TOKENIZATION

All the terms in the FAQ database and in SMS queries are brought into token format. Each word or term is converted into a token to apply every level of processing. User sends query to the system the first step to process it is tokenizing it. All the FAQ that are stored in Database are tokenized for each and every question.

### 2.3.3. STOP WORDS

Stop words are very common words that contain less important meaning than keywords. Generally search engines expunge stop words from the keywords phrase to riposte the most relevant result. Stop words are the part of human language.

In this presented system, experimentation is main motive, so rather than removing the stop words, system has taken it under consideration. One another file is maintained to hold the stop words which are encountered in questions that are present in database and in the SMS query. File for stop words related to questions and SMS queries are maintained separately and the questions and SMS queries that does not contain the stop words are maintained separately. FAQ corpus is filtered out. From the SMS query S stop words are removed. It became processed query. The list of stop words which is used in the system are the most common short function such as is, at, the, on, etc, and common lexical words as well. In the file which only contains stop words not even single key word.

### 2.3.4. STEMMING

"Stemming is a process for reducing inflected(or sometimes derived) words to their stem, base or root form- generally written in word form, in linguistic morphology and information retrieval [25]." It is not necessary that stem and morphological root of the word should be identical. Even if the stem is not a valid word, only related word map to the same stem is usually sufficient. Stemming algorithms have been studied in computer science since 1968 [25]. Words with same stem are treated as synonyms as a query broadening in many search engines, a process is called as **conflation**.

Basic words contain many variants of it, which are present in natural text. Morphological variants (e.g. computing, computer, computers, computational, computers etc.) are most common, and other source including valid alternative misspellings, spellings and variants made up from transliteration and abbreviation. The effectiveness of searching would be expected to increase if it were possible to conflate (i.e. to bring together) [26] the variants of a given word so that they could all be retrieved in response to a query that specify just a single variant. In English and many related languages, morphological transformations takes place at the right- hand end of a word- form [27], and this has accelerated the user directed right hand truncation for information retrieval. In many cases, morphological variants of words are taken approximately equal and have similar semantic interpretations for information retrieval applications. Because of this reason, lots of stemming algorithms or stemmers has been developed, which are used to reduce the words to their stem or root forms. So in our system the keywords of SMS query and questions in database are represented by stems rather than original words. By this, different variants of a word can be conflated to a common form as well as it minimizes the size of dictionary, i.e. the number of distinct words needed for representing a set of questions. Smaller dictionary size results in a saving of processing time and storage space. For information retrieval, it is not necessary that the stem generates genuine word e.g. comput is stemmed word for “computer”. Here the different words with same base word meaning are conflated and distinct meaning words are kept separate. “lemmatiser” is an algorithm which converts a word to its linguistically root word [28]. For information retrieval purpose the automatic removal of suffixes is very useful. Words with common root usually have same meanings. To improve the performance of system this stemming is done in which such terms are conflated into a single word or term. For example, if COMPUT is taken as root term then the words for this are – COMPUTE, COMPUTER, COMPUTATION, COMPUTATIONAL, COMPUTATIONS, COMPUTERS, and extreme basic meaning of all is same. Similarly for argue, terms are argument, argued, arguing, argues. It reduces number of words or terms, hence size and the complexity is also reduced.

For suffix stripping many strategies are added in literature. The features of exercise changed considerably based whether on suffix list used, on dictionary for stemmed words and on the actual purpose of suffix stripping. If stem dictionary is not used and have to improve information retrieval performance, then explicitly a suffix list is given to the program, so that it can give the stem word for the given term. Stemming programs are also known as **stemmers or stemming algorithms**. First stemmer was written by Julie Beth Lovins in 1968 [29]. It was a phenomenal paper and had immense effect on later work in this area. After this the next influencing stemmer was given by Martin Porter, published in July 1980 issue of the journal. This stemmer was very widely used and became the de facto standard algorithm used for English stemming. Dr. Porter received the Tony Kent Strix award in 2000 for his work on stemming and information retrieval [25].

#### 2.3.4.1. Porter's Algorithm

There are two major differences in Porter's algorithm and Lovin's algorithm. First, for associated rules with suffix removal there is convincing reduction in complexity.

Lovin's algorithm contains 294 suffixes, each of which is associated with one of 29 context sensitive rules that determine when that suffix can or cannot be removed from the end of a word; the algorithm also contains 35 recoding rules [29]. Recording rules and suffixes suggest that the Lovins algorithm has been designed principally for the processing of scientific texts [30]. Second, consolidated approach for context handling. Lovin's rule of context sensitive is related to the remaining length of stem after removing suffix. If two or less than two character left after stemming may lead to over stemming. For suffix stripping automated alternative is given by a stemming algorithm [31]. By this algorithm all the terms are reduced to a single form, in this basically all the right hand side end of the root term are removed, so that it can come to basic root word. Prefix removal is less studied in this area and this suffix removal is also applied for other languages as well [32]. Porter's algorithm not only achievement in itself, it is also an inspiration for consequential algorithms. It have many variants but with minor changes. In the presented system original program of Porter's is used. It is using simple conceptions with only 60 suffixes, 1 type of context sensitive rule and 2 recording rules to find out suffix should be removed or not rather than on number of character left as Lovin's algorithm says. It is based on number of vowel and consonant-vowel characters (Measure) remaining after suffix removal from the given term must be greater than 1 for every rule applied. This technique is widely used. Algorithm knows that it does not have to remove suffix if the length of term is too short. Stem length is calculated by its measure, n. No linguistic approach is applied.

ALGORITHM:

A term can have one of the form out of {C...C,V....V, C....V, V....C}, this can be represented as $\{C\}(VC)^n\{V\}$. There are 5 steps in Porter's algorithm:

Step 1: Plurals and past participles are taken into consideration.

E.g. educated – educate, teaching- teach

Step 2: On some common suffix Pattern matching is done.

E.g. calculation- calculate, happy- happi.

Step 3: Special ending words.

E.g. joyful – joy, cyclical- cyclic

Step 4: If words are compounded, check each word for more suffixes

E.g. allowance – allow, retrieval – retriev.

Step 5: check after removal of suffix if it ends with a vowel and fixes if accordingly.

E.g. possible – possibl, controll – control

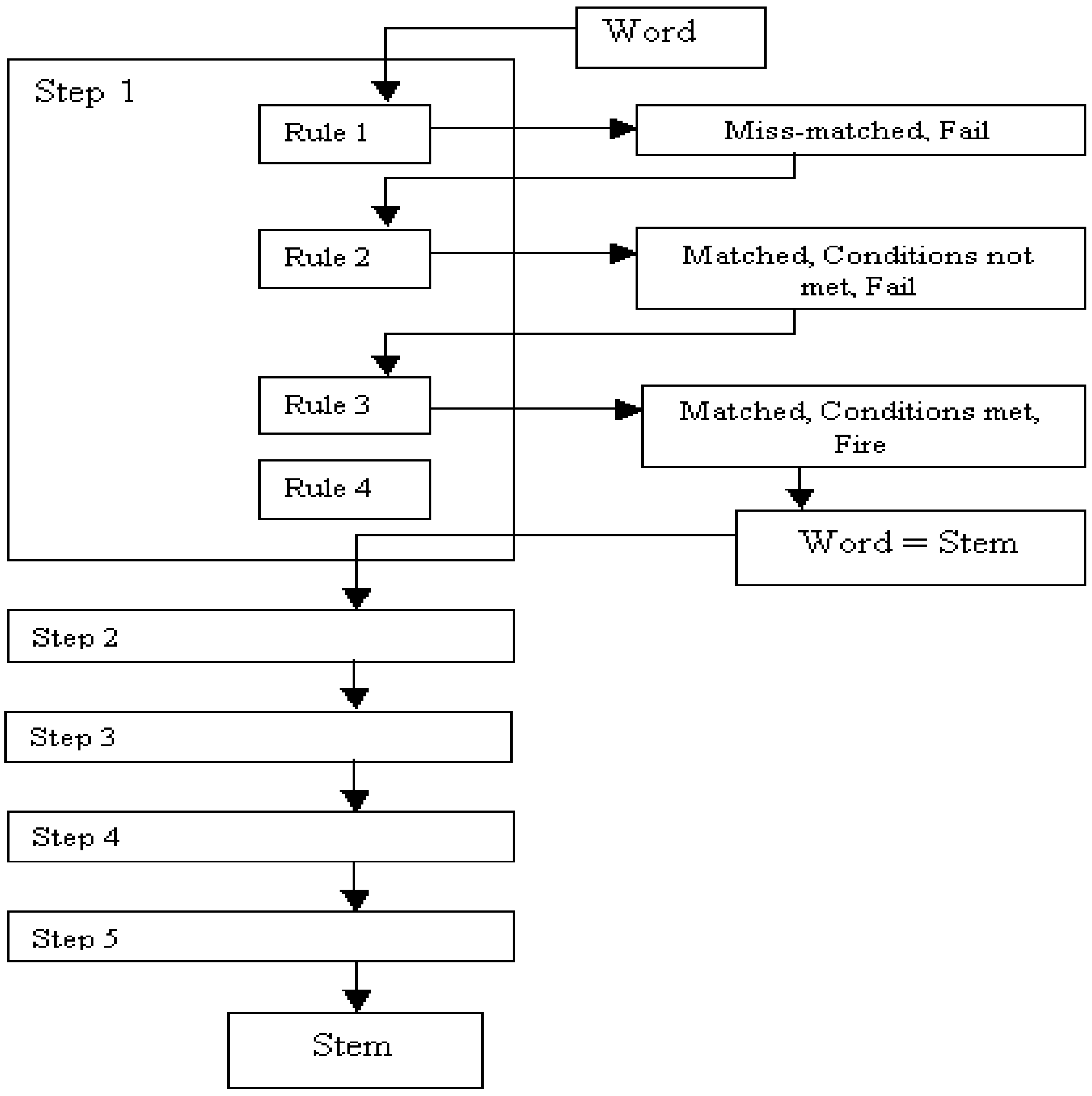

**Figure 2.1: Porter's Algorithm**

# CHAPTER 3

# SIMILARITY MEASURES

Fundamentally the concepts of similarity are very important and used in almost every field, e.g. in homothety, geometric methods for similarity, mathematics and in trigonometry. Fuzzy theory has their own similarity measures as well as in molecular biology to find pair of proteins. Similarity measures are functions, which are used to calculate the degree to which two compared objects are similar to each other. Functions should satisfy few similarity conditions. Similarity is the degree of closeness or separation between two objects. Many types of Similarity and distance measures have been proposed and widely used.

Where sim(x, y) ∈ [0,1] is some similarity function defined on the collection of objects. Such a scheme leads to a compact representation of objects so that similarity of objects, similarity sim(x, y) is a function that maps pairs of objects x, y to a number in [0, 1], measuring the degree of similarity between x and y. sim(x, y) = 1 corresponds to objects x, y that are identical while sim(x, y) = 0 corresponds to objects that are totally different. Measures are commonly used in the fields of information retrieval and compare two text documents.

There are many similarity available functions “Characters-based (e.g., Levenshtein distance, n-gram distances), context-based, tokens (e.g., Jaccard distance, TF-IDF (cosine similarity combined with the tf.idf weighting scheme to compute the similarity of two fields), phonetics (e.g., soundex distance) [40].

In this section we define some of the existing similarity measures and also our weighted similarity model.

## 3.1. SOUNDEX SIMILARITY MEASURE

Phonetic, misspellings, transliterations, typographical errors, abbreviations, integration of multiple data sources and cultural variations obscure information retrieval. Without lowering the precision improving recall is a challenging task. Sub language is created because of the limitation on SMS length which includes phonetically similar word to natural language, words vocabulary but eliminates punctuation marks, vowel and grammatical forms [33]. Performance of search engine is analyzed when representing questions related to information retrieval system with phonetic code. In the SMS based question answering system, input is SMS query and output is set of codes. Main motive is to find family of terms that matches with the same codes.

For example, names can be of different length, can have different spellings and not unique but have same pronunciation. To solve the problem phonetic algorithms are needed, which can find similar sounding terms and names. By Phonetic algorithm (combination of fuzzy matching algorithms) two different terms with same pronunciation in matched and gives same code for both of them, which allows setting comparison between phonetically similar words. In such cases, to simplify the problem phonetic algorithms are used. Many applications are there for phonetic representation. It encourages to find concepts based on pronunciation rather than spellings. There are different algorithms for codification of text on the basis of phonetic pronunciation. Information retrieval often use Soundex code, when it is based on spoken language transcriptions, because it is known as speech recognition transcription errors i.e. orthographically dissimilar and phonetically similar [33].

**Soundex** function is used to find terms when only the sound is known but not spelling. It finds out the terms which have similar sound based phonetic assumptions, the pronunciation of word is considered. E.g. searching of names in a database, where spellings may very but pronounced in a same manner. It has become source of many algorithms. It is a long established, fast, effective, non-proprietary and efficient procedure for certain types of spelling variation. Soundex covers many variations; firstly it was developed by patented by Robert C. Russell in 1918. It developed for converting surnames into a code of single character and three digits. Basic concept of Soundex is a relationship between characters and sound confirms that similar sounding terms are assigned the same code [34]. Take a word as a input and outputs a code that identifies a group of words that are approximately phonetically same. This algorithm is handy for processing large database and huge database.

Main aim is to encode homophones to the same code so that they can matched even with dinky differences in spellings [35]. It encodes only consonants and vowels are dropped unless those are first letter of string.

The basic two steps for every phonetic algorithm are: Consecutive consonants are removed, removal of non initial vowels and leading characters are stored rest non leading characters are digits.

Regulations for it are as follows:

- Each character is input and assigned a value.
- Convert each term into a code of four elements, out of which 1 is character rest three are numeric values.

- Except first letter, all letters are replaced by their phonetic code.
- Eliminate adjacent repetitions of codes.
- Eliminate all non initial vowels.
- First four letters of resulting string is returned.
- If word ends before four letters of code then pad 0s to it.
- If two letters belong to same code consecutively then replace it by single code.

Features of Soundex :

- Words sounds similar or are exactly same it gives high tolerance for variation.
- According to objective easy to change
- Affective in terms of operation.
- Processing time is less.

| CODE | LETTER |
|---|---|
| 1 | B,F,P,V |
| 2 | C,G,J,K,Q,S,X,Z |
| 3 | D,T |
| 4 | L |
| 5 | M,N |
| 6 | R |

**Table 3.1: SOUNDEX TABLE**

In the presented system, Soundex phonetic algorithm for information retrieval task, SMS queries comes, it uses Soundex algorithm to convert it into natural language so that, the output can be matched with the database question i.e. one text is in SMS sub language and another text is in natural language has t be compare so Sub- language has to be converted in natural language.

Example: H326 for Hatzar, Hedger, Hadgraft, Hadcroft.

## 3.2. JACCARD'S SIMMILARITY MEASURE

The **Jaccard Similarity** measure is similarity between sample sets. It is complementary to the Jaccard distance and is obtained by subtracting the Jaccard distance from 1.The Jaccard coefficient or Jaccard Similarity or Jaccard Index can be calculated by size of the intersection divided by the size of the union of the sample sets. Jaccard Cofficient is denoted by J(A,B). For the presented system, A and B are two sets of string or tokens.

Jaccard Cofficient or Tanimoto Coeficient is formulated as below:

$$J(A,B) = \frac{A \cap B}{A \cup B}$$

**Equation 3.1: Jaccard's Similarity**

So The Jaccard Distance can be formulated as below:

$$J_\delta(A,B) = 1 - J(A,B)$$

**Equation 3.2: Jaccard's Dissimilarity**

Some verified facts of Jaccard Similarity Measure are:

1. Jaccard Similarity J(A,B) is nonnegative because the size of the intersection cannot exceed the size of the union.
2. J(A,B) = 1 if A = B, because A ∪ B = A ∩ B = A. However, if A = B, then the size of A∩B is strictly less than the size of A ∪ B, so J(A, B) is strictly positive.
3. J(A,B) = J(B,A) because both union and intersection are symmetric i.e., A ∪ B = A ∪ B and A ∩ B = A ∩ B.

Example: Set A : where is nearest PSK

Set B : Where is nearest PSK and how to reach.

A ∩ B = where, is, nearest, PSK.

= 4 Tokens.

A ∪ B = where, is, nearest, PSK, and, how, to, reach, there.

= 9 Tokens.

Jaccard Coefficient = A ∩ B / A ∪ B

= 4/9 = 0.4444.

Jaccard Distance = 1- Jaccard Coefficient

= 1-0.4444

= 0.45454545454

Similarity measures are specific functions used to approximate the degree to which two compared objects are similar to one another. This measure has a great potential to be used in solving multi-criteria decision making problems where the weight factor of parameters (criteria) involved is of great importance and must be taken into consideration in the evaluation process. The Jaccard similarity can be used, when interested in binary differences between two or more objects, can be used for numeric set and for string set system is using [38].

## 3.3. COSINE SIMILARITY MEASURE

Cosine similarity is a one of the most popular similarity measure. The measure of similarity between two vectors of *n* dimensions by finding the cosine of the angle between them, often used to compare documents in text mining [39]. Given two vectors of attributes, *A* and *B*, the cosine similarity, θ, is represented using a dot product and magnitude.

The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$a.b = \left|\left|a\right|\right|\left|\left|b\right|\right| cos\theta$$

**Equation 1.3: Cosine Formula**

Two vectors of attributes, *A* and *B*, the cosine similarity, θ, is represented using a dot product and magnitude as

$$Sim(Cos\theta) = \frac{A.B}{||A||||B||} = \frac{\sum A_i \times B_i}{\sqrt{\sum A_i^2}\sqrt{\sum B_i^2}}$$

**Equation 3.4: Cosine Similarity**

The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude: two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0, 1].

System measure degree of similarity of two questions as correlation between their corresponding vectors, which is designated as the cosine of the angle between two vectors.

# CHAPTER 4

# PROPOSED HYBRID SIMILARITY

The objective of this approach is to introduce hybridization of similarity measure for SMS based FAQ retrieval system. Hybridizing can be done in many ways:

- Different or same measures are merged to give another measure by using additional information.
- Another way is, one measure's output is given as input of another measure, and the combined form of these two is called as Hybridizing. A hybrid similarity mixes several singlr measures with a combination method to achieve retrieval results.

In the proposed work, the system is implementing both types of hybridizing. Firstly, using one similarity measure for another one and second is deriving a formula by using additional information over a single similarity measure. Cosine and Jaccard are some of the best corpus based measures. Hence, these measures are taken as baselines to proposed model.

As discussed above sections the system is for retrieving an appropriate answer for a given SMS query, which is sent by user in text- language. System has a database of Frequently

Asked Questions, when the query arrives at the system; query is matched with the FAQs in the database then the best matched question's answer is reverted to user. Main focus is done on the matching of the user SMS query with FAQs. This work is all basically related to that similarity between SMS query and the FAQ. Let SMS token be $S_t$= {S1, S2, S3.........} and for each FAQ database question tokens be $F_t$= {F1, F2, F3.........}.

## 4.1. PROPOSED METHODS

In presented work, a novel approach has been presented by developing Hybrid similarities which evaluates similarity scores with the questions in the corpus for SMS query. In this way, we can further improve the accuracy of the SMS based FAQ system significantly by refining the results of the system using different hybrid similarity scores. All similarity measure have range between 0<Similarity measure<1.

### 4.1.1. Hybrid Soundex Measure

In this method, when system gets a SMS Query in text- language, then it contain abbreviations, short forms, spelling mistakes, transliterations and phonetic spellings create lot of noise. Normal text similarity cannot be applied on it, such text is not even recognized by the system apart from matching, and SMS language has to convert into natural language for applying similarity measures on it.

Soundex algorithm is first applied on the SMS text query to convert it into a correct form or natural language form then Jaccard similarity measure is applied for matching the corrected SMS text and the questions in the FAQ.

SMS query tokens, $S_t$ = {S1, S2, S3, S4...........}

Database question tokens, $F_t$ = {F1, F2, F3, F4.......}

Soundex algorithm is applied on SMS tokens and gives tokens in natural language, by converting phonetic words in English language. Let new corrected tokens be $N_t$= {N1, N2, N3, N4......}.

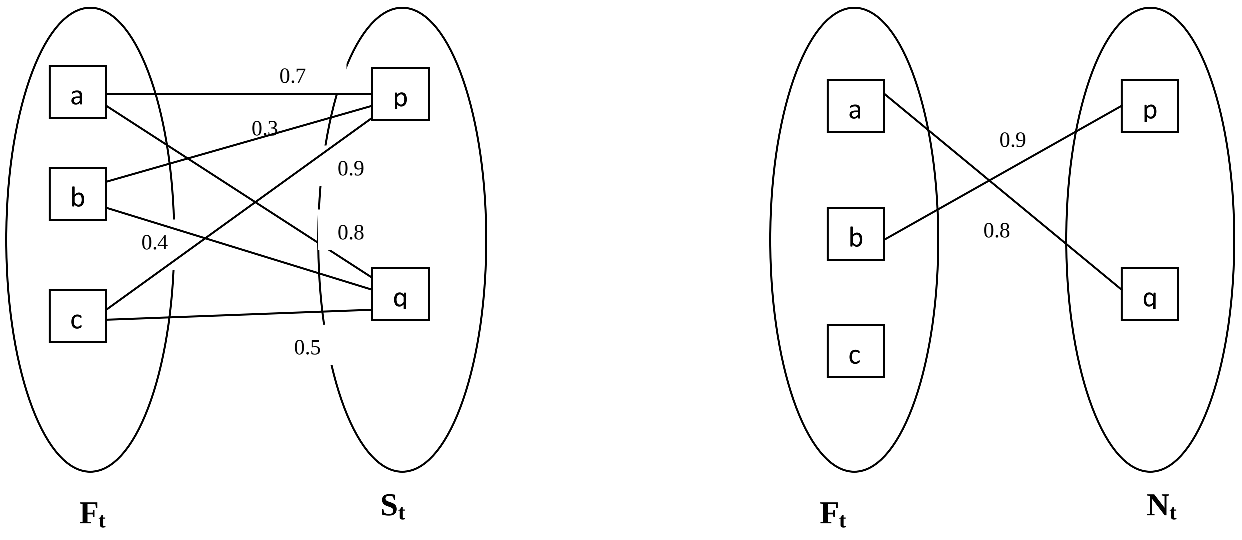

**Figure 4.1: Soundex Matching**

Each question in FAQ Database is denoted by "x". Here $S_t$ , $N_t$ and $F_t$ are tokens of old SMS, new SMS converted in natural language and questions in FAQ database respectively. For conversion each word in SMS is matched with every word in FAQ database question. With the help of Soundex algorithm, it is done token by token. As soundex works on tokens, not appropriate for whole string. As shown in the fig.2 each token of $S_t$ is matched with every token of $F_t$ by soundex similarity measure. Then the maximum scored weight is stored and rest are discarded, and that best matched word is replaced by the original word which is present in $F_t$ because it contain tokens of FAQ data set and that is in natural language. Let the output of this procedure be "$S_x(S_t, F_t)$".

### 4.1.2. Hybrid S-Jaccard's Similarity Measure

Stop words has been removed and stemming of words is done, now Jaccard's similarity measure is applied on the output of Soundex procedure, So named Hybrid S-Jaccard. Using this accuracy of the output will increase, because now it matches with a sentence which is already in natural language .So the similarity score will increase. As it is mentioned earlier, old SMS tokens are denoted by "$S_t$", $F_t$ for tokens of FAQ data set and each question in FAQ Database is denoted by "x " and the output of Soundex procedure be "$S_x(S_t, F_t)$".

So now derived formula is given below:

$$J_x(S_x, F_t) = \frac{F_t \cap S_x(S_t, F_t)}{F_t \cup S_x(S_t, F_t)}$$

**Equation 4.1: Hybrid S-Jaccard**

Where, $J_x(S,F_t)$ is similarity measure of Jaccard's in hybrid form.

In Hybrid Jaccard's formula, the number of common terms is divided by union of all terms or tokens present in each question and in new converted SMS, which has been processed by Soundex algorithm.

### 4.1.3. Hybrid S-Cosine Similarity Measure

As the system works for Jaccard's similarity measure, in the same way Cosine Similarity measure is hybridized. Stop words has been removed and stemming of words is done on the output of Soundex procedure, now Cosine similarity measure is applied. Using this accuracy of the output will increase, because now it matches with a sentence which is already in natural language. So the similarity score will increase. As it is mentioned earlier, old SMS tokens are denoted by "$S_t$", $F_t$ for tokens of FAQ data set and each question in FAQ Database is denoted by "x " and the output of Soundex procedure be "$S_x(S_t, F_t)$".

So now derived formula is given below:

$$C_x\,(F_t, S_x) = \frac{\sum F_t \times S_x(S_{t,} F_t)}{\sqrt{\sum F_t^2}\,\sqrt{\sum S_x^2\,(S_t, F_t)}}$$

**Equation 4.2: Hybrid S-Cosine**

Where, $C_x(F_t, S_t)$ is Hybrid similarity by Cosine.

### 4.1.4. Hybrid Jaccard's Similarity Measure

In this method, the Jaccard's formula is modified on the basis of involvement of stop words. Generally in any of the similarity measures first stop words are removed then measure is applied on text but here the stop words are considered but with less weight age in compared to weight age given to keywords. It increases systems precision but measure of similarity decreases with correct answer, because stop words create variation in measure. Unlike implementation of other similarity measure in this method stop words are stored in different file and used at the time of applying formula.

Two additional terms added $S_p$ and $F_p$, one for prepositions in SMS and in FAQ dataset respectively.

Formula is given below:

$$H_x\left(S_x, F_t, S_p\right) = \frac{F_t \cap S_x(S_{t\,,}F_t)}{F_t \cup S_x(S_{t\,,}F_t)} + (\varphi)\frac{S_p \cap F_p}{S_p \cup F_P}$$

**Equation 4.3: Hybrid Jaccard**

Where, $H_x(S_x, F_t, S_p)$ is Hybrid Jaccard Similarity Measure and $\varphi$ is preposition coefficient for giving weight to preposition set ($0< \varphi <1$).

NOTE: As two sets are added with respect to keywords and prepositions, both have 0 to 1 range in itself, so the range of similarity is increased i.e. 0<Similarity Measure<2.

# CHAPTER 5

# IMPLEMENTATION

# AND EXPERIMENTAL RESULTS

## 5.1. ENVIRONMENTAL SETUP

The system used the following configuration while finding the experimental results.

### 5.1.1. HARDWARE CONFIGURATION

| | | |
|---|---|---|
| Processor | : | Intel Core 2 Duo Processor |
| Speed | : | 2.20GHz |
| Main Storage | : | 4GB RAM Hard Disk |
| Hard Disk Capacity | : | 80GB |
| Monitor | : | Dell 17”5 Color |

### 5.1.2 SOFTWARE CONFIGURATION

| | | |
|---|---|---|
| Operating System | : | Windows 7 |
| Front end | : | Java |
| Back end | : | Datasets in XML (explained in 5.2) |

## 5.2. Datasets

Dataset used for evaluation was taken from Passport Government website [37]. It is official website of government for passport related FAQs. It contains data from many Passport related domains viz. – tatkaal, user registration, user assistance, miscellaneous, etc. The SMS queries were also provided by the user in order to test the system accuracy.

**Dataset format:** The data is in an XML-based format. FAQs are placed in the input FAQ xml file and SMS queries are placed in SMS query xml file. The two formats are:

```
<xmldata>
  <doc id="2">
      <question>
What types of passports are issued in India?
      </question>
       <ans>
Ordinary Passport: An ordinary passport appears with a blue cover, and consists of 36 or 60 pages. It is valid for 10 years from the date of issue and can be renewed for another 10 years.  </ans>
   </doc>
 </xmldata>
```

**Figure 5.1: FAQ Format**

```
<SMS>

<SMS_Text>

What’s need to change name on pport after a marriage

</SMS_TEXT>

</SMS>
```

**Figure 5.2: SMS Format**

## 5.3. ANALYSIS AND RESULTS

In current experiments, Baseline is considered of S-Jaccard’s Similarity Measure, S-Cosine Similarity Measure and Hybrid Jaccard’s Similarity Measure occurrences as the attributes for evaluation. Three different experiments were conducted and one experiment is inbuilt of Soundex Similarity Measure which executes for correction of SMS query. In first experiment, only S-Jaccard’s Similarity is considered the between SMS token and FAQ terms, In Second experiment -Cosine Similarity was considered and in Third experiment Hybrid Jaccard’s Similarity measure considered. Finally I have observed that all the results are giving highest score to relevant question, just the value for Hybrid Jaccard is low but with correct answer, that means it gives precision to the output. The precision of the system improved.

All three experiments are repeated three times by taking three different score thresholds. This score threshold determines whether to consider the match of a FAQ with SMS or not. In this way, the percentage correctness of queries is observed in each case. For a given SMS query accuracy was tested based on the top three FAQ returned by the system, if the required FAQ is present in the top three then the answer is marked as true.

In table, maximum similarity score is assigned to corresponding similarity measures.

In graph, X- axis is for Mean Marginal Relevance (MMR) and Y- axis is for Question Entries in FAQ.

**Threshold 1:**

Given SMS query: "Wat r d vrious psport srvcs and whch form hs to be fild in?"

| SIMILARITY MEASURE | SIMILARITY SCORE |
|---|---|
| S-JACCARD | 0.6863636255264283 |
| S- COSINE | 0.8577777922153473 |
| HYBRID JACCARD | 0.6363636255264282 |

**Table 5.1: Experimental Result for T1**

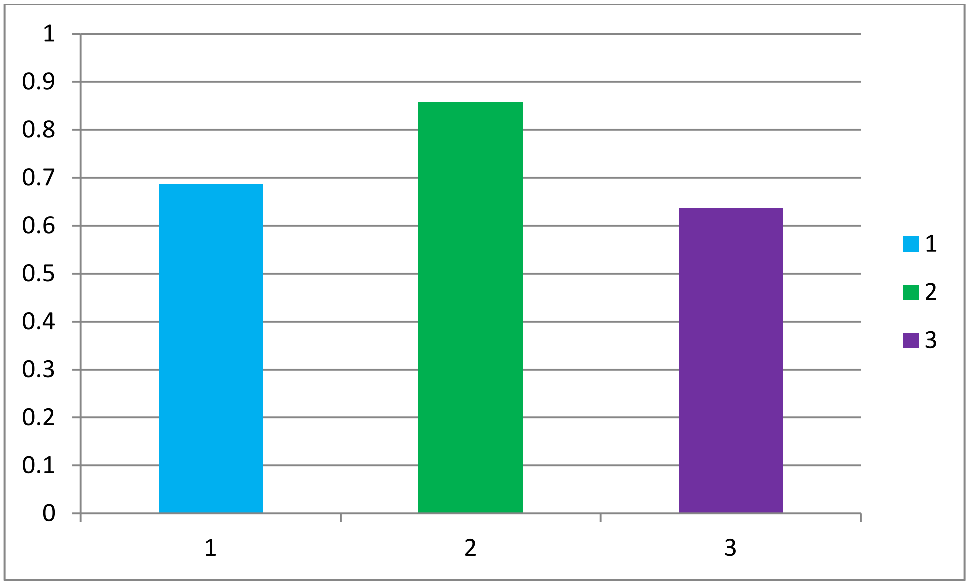

**Figure 5.3: Graph for T1 with maximum values**

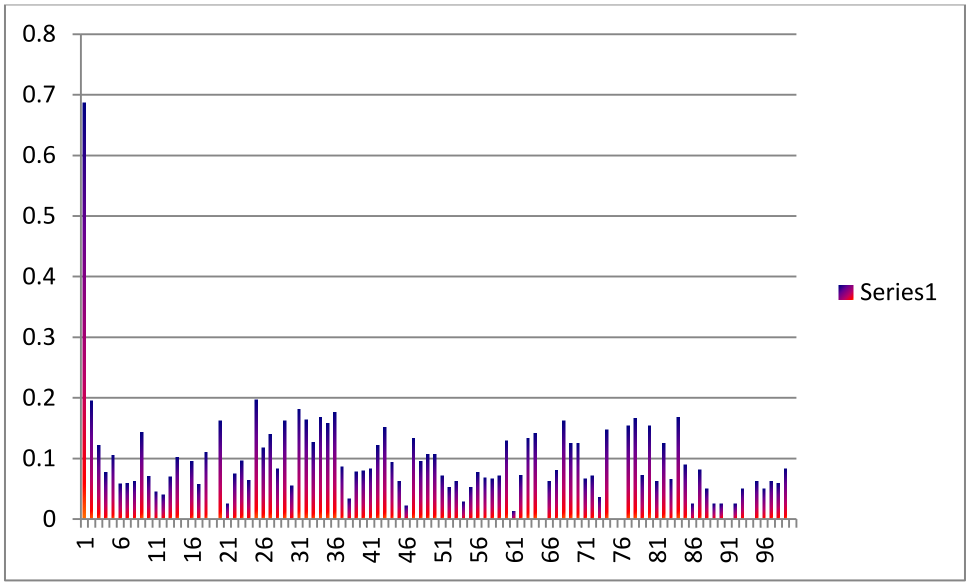

**Figure 5.4: Graph for T1 with S-Jaccard**

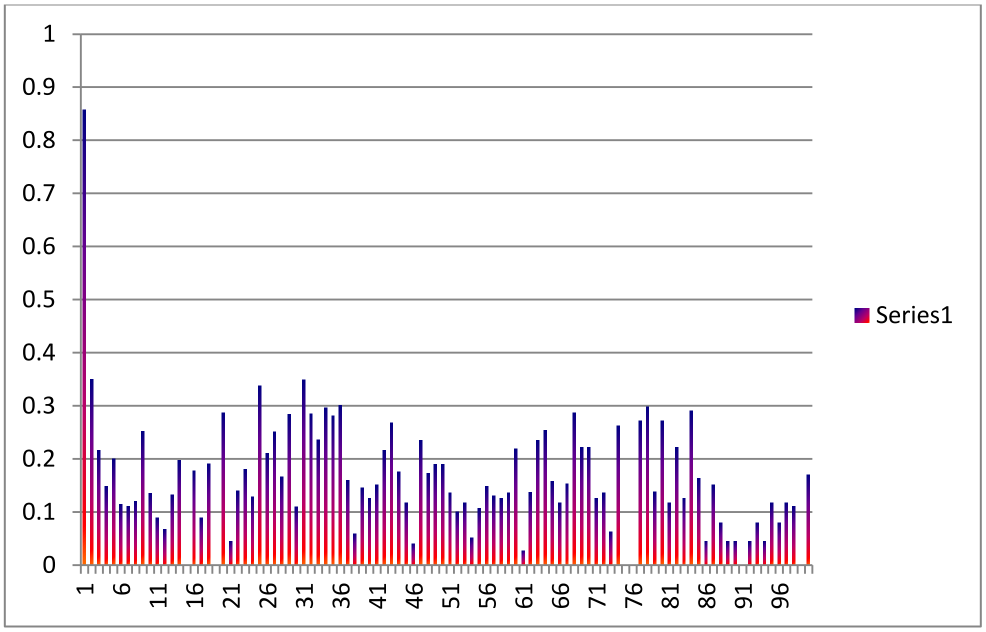

**Figure 5.5: Graph for T1 with S-Cosine**

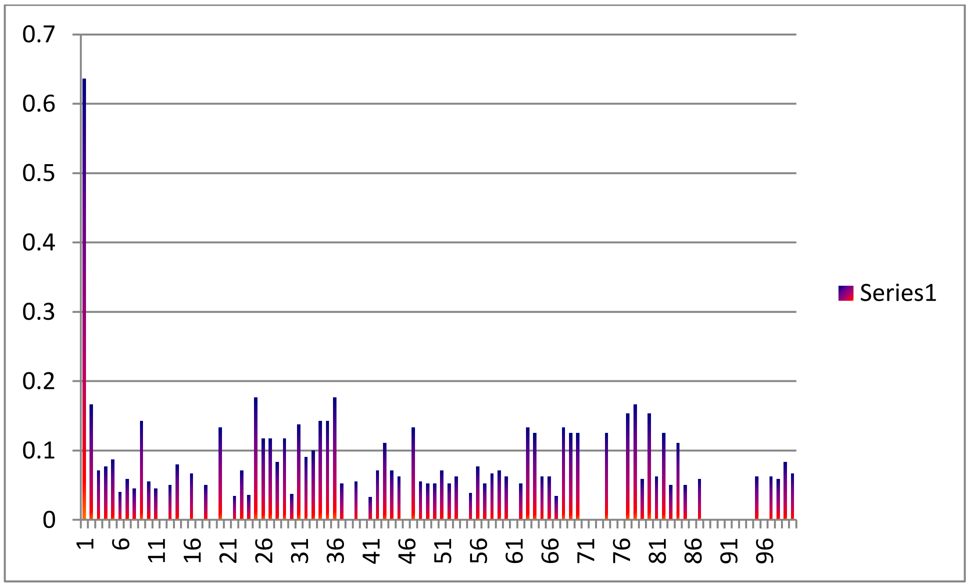

**Figure 5.6: Graph for T1 with Hybrid Jaccard**

Result: Top three results for T1 are Question number 1, 2, 25 according to S-Jaccard,S-Cosine and Hybrid Jaccard. These are correct and relevant questions according to dataset.

**Threshold 2:**

Given SMS query: “Wat iz the dfinitn of minr for issu of psprt?”

| SIMILARITY MEASURE | SIMILARITY SCORE |
|---|---|
| S-JACCARD | 0.5785714328289032 |
| S- COSINE | 0.7732051134109497 |
| HYBRID JACCARD | 0.4285714328289032 |

**Table 5.2: Experimental Result for T2**

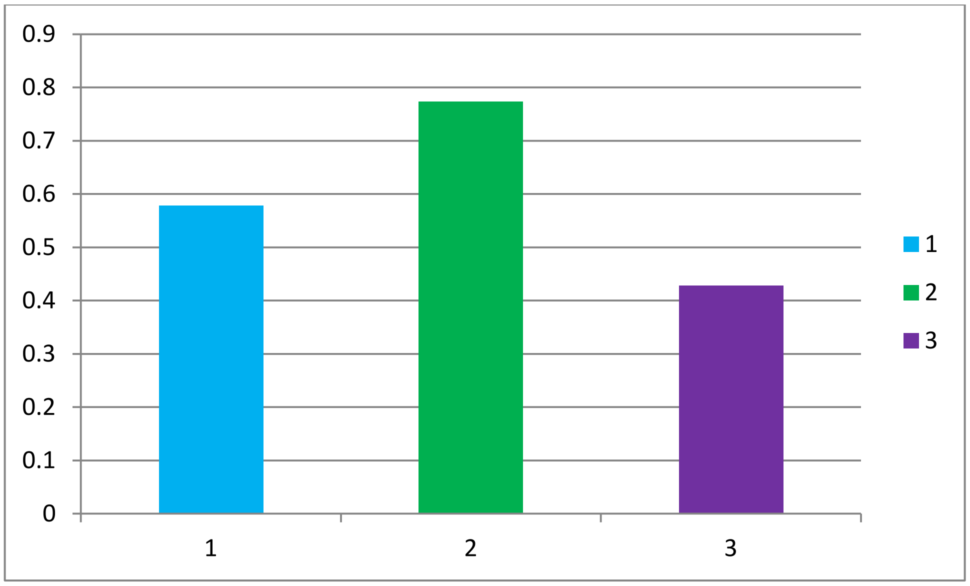

**Figure 5.7: Graph for T2 with maximum values**

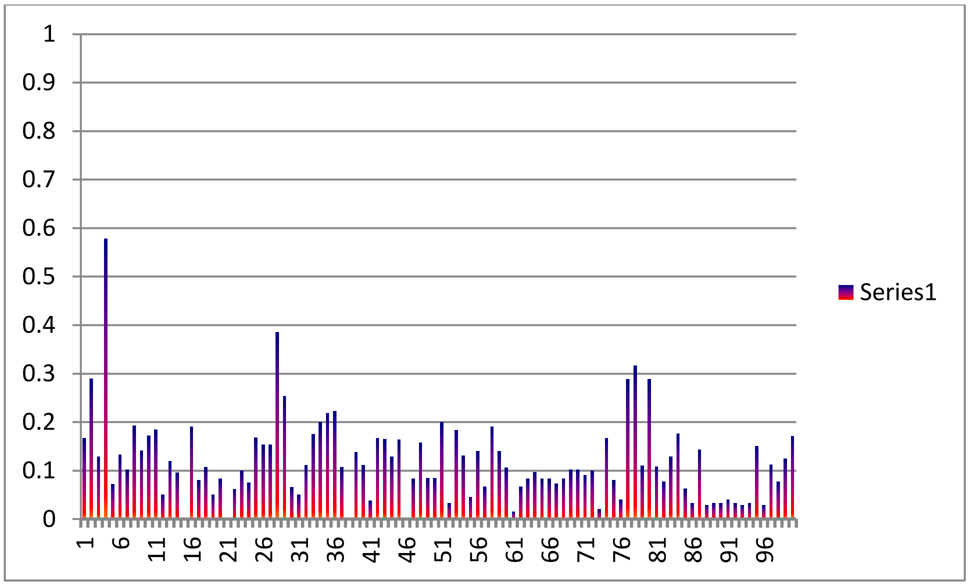

**Figure 5.8: Graph for T2 with S-Jaccard**

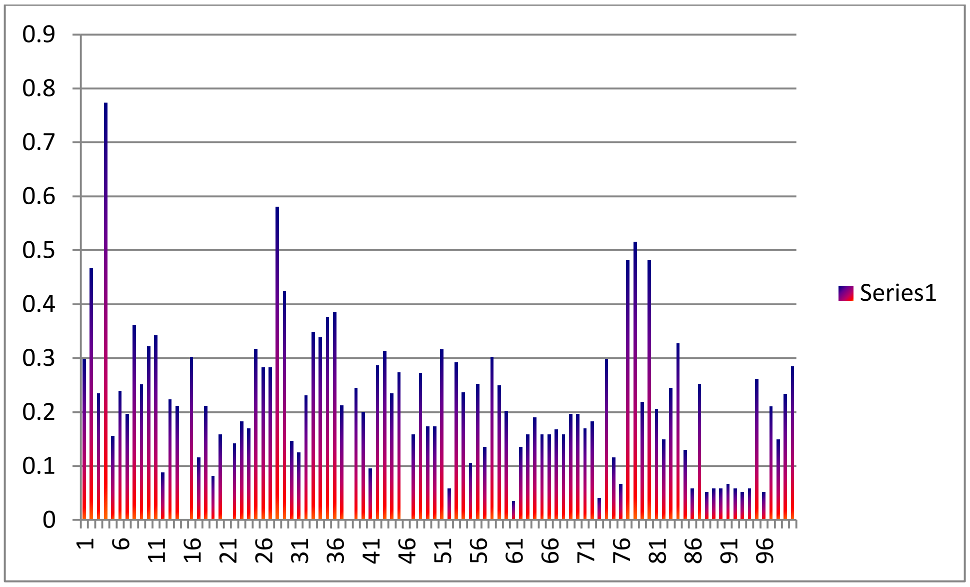

**Figure 5.9: Graph for T2 with S-Cosine**

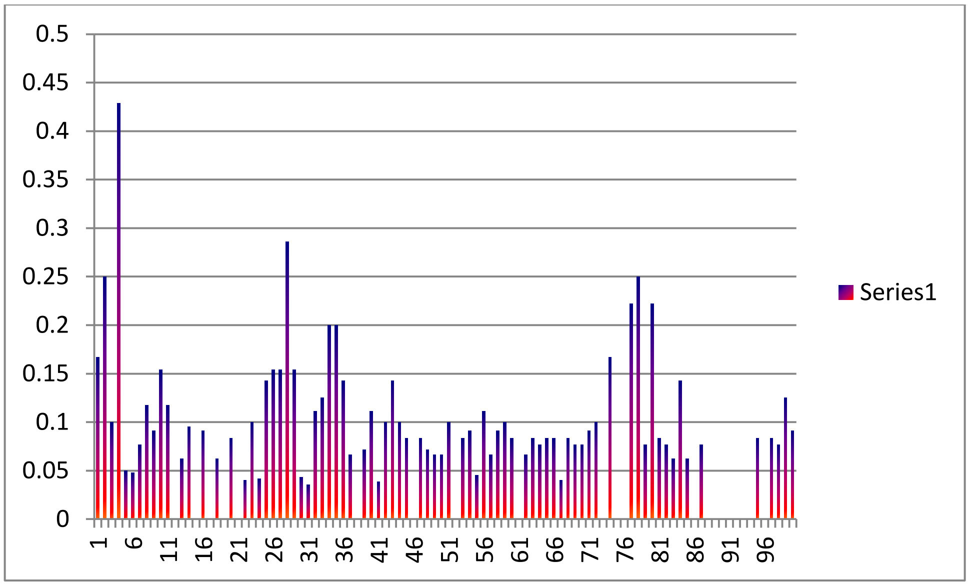

**Figure 5.10: Graph for T2 with Hybrid Jaccard**

Result: Top three results for T2 are Question number 4, 28, 78 according to S-Jaccard, S-Cosine and Hybrid Jaccard. These are correct and relevant questions according to dataset.

**Threshold 3:**

Given SMS Query: "Wat iz the dfinitn of minr fr issu of psprt?"

| SIMILARITY MEASURE | SIMILARITY SCORE |
|---|---|
| S-JACCARD | 0.46363637447357176 |
| S- COSINE | 0.6759438276290893 |
| HYBRID JACCARD | 0.3636363744735718 |

**Table 5.3: Experimental Result for T3**

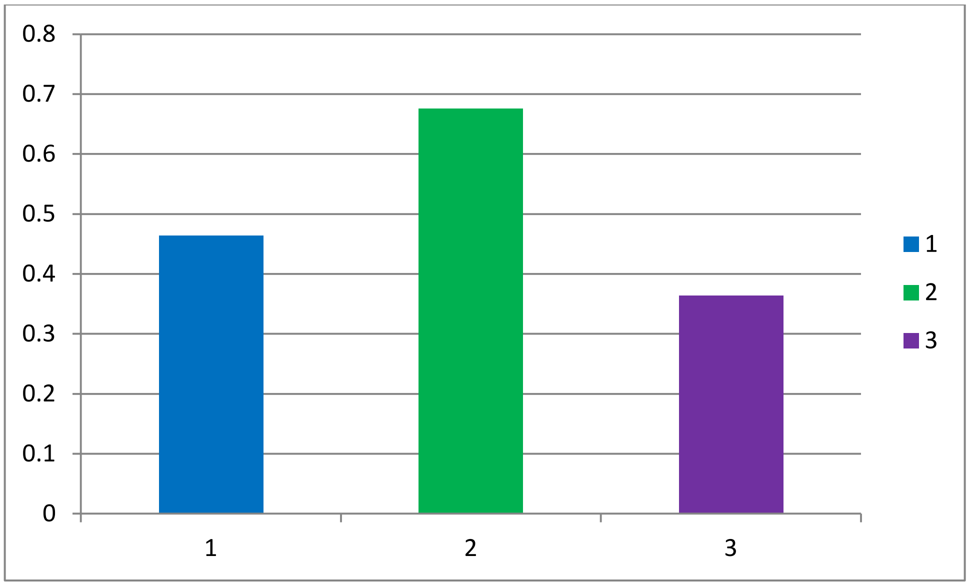

**Figure 5.11: Graph for T3 with maximum values**

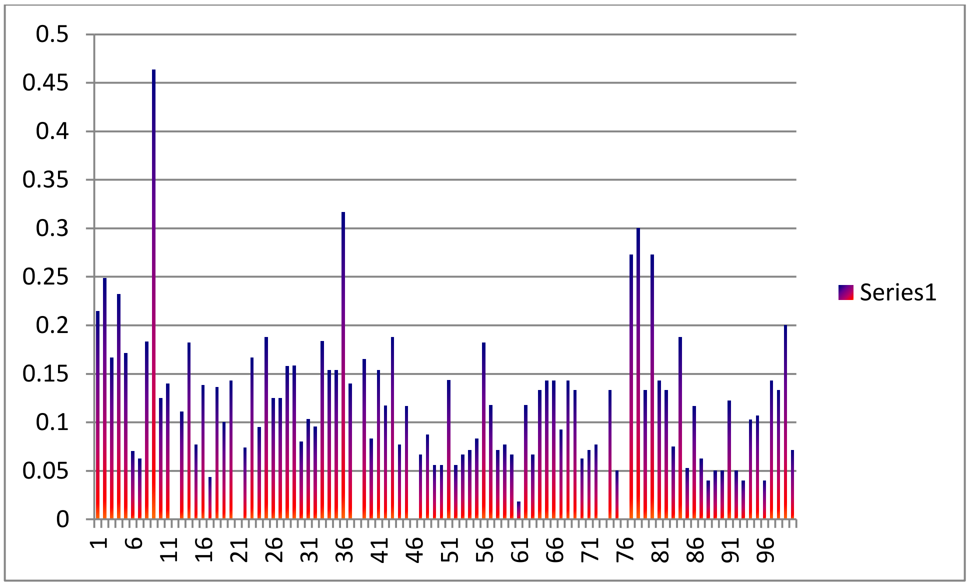

**Figure 5.12: Graph for T3 with S-Jaccard**

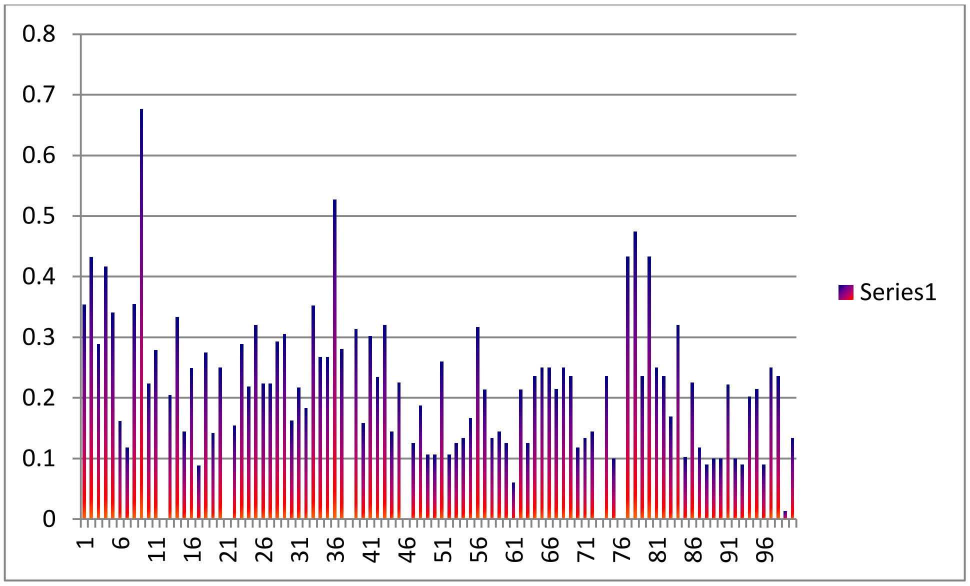

**Figure 5.13: Graph for T3 with S-Cosine**

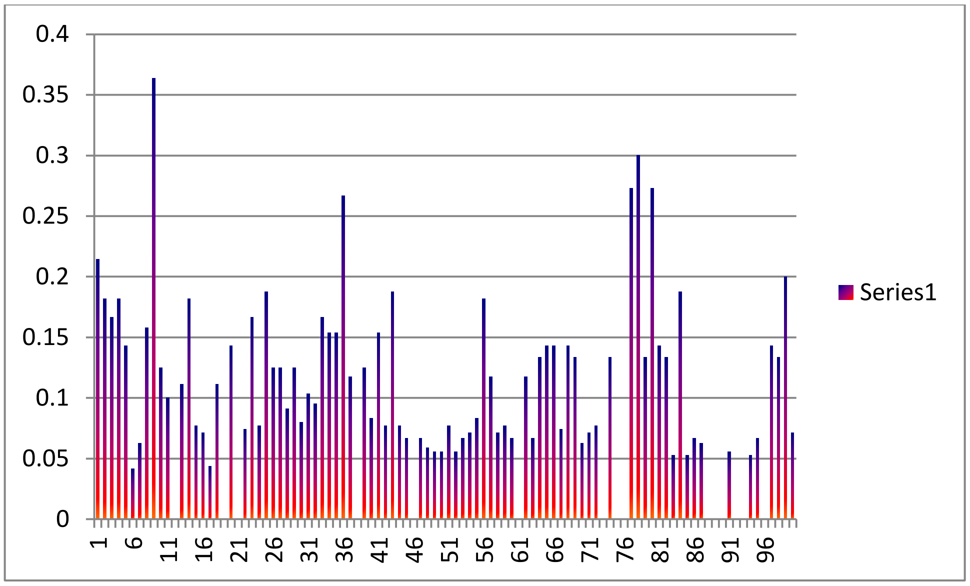

**Figure 5.14: Graph for T3 with Hybrid Jaccard**

Result: Top three results for T3 are Question number 9, 36, 78 according to S-Jaccard, S-Cosine and Hybrid Jaccard. These are correct and relevant questions according to dataset.

## 5.4. SUMMARY

This chapter introduced the environmental setup which System used while making the experimental results. In addition to this, type of dataset is explained which is used and explained it in detail. Various experiments were conducted to test the accuracy of various matching techniques. From the experimental results this can be conclude that this approach is able to significantly outperform the previous state-of-the-arts SMS based QA system, particularly in case of similarity measure, the results are precise

# CHAPTER 6

# CONCLUSION

# AND FUTURE SCOPE

## 6.1. CONCLUSION

There has been little work on SMS-based search for arbitrary topics due to the initial lack of a well defined business cases. The explosive growth in prevalence of affordable low-end mobile devices throughout the world has created a large market for mobile information services. Since mobile users in many parts of the world use low-end mobile devices with SMS as their primary data transport, therefore, SMS-based search becomes a critical problem to address on the path to enabling SMS-based services. In this Thesis, an automated SMS-based search response system that is tailored to work across arbitrary topics is presented. The use of different similarity technique adds to the scalability of the software without adding much to its complexity. A novel approach based on Hybrid Similarity measure scores is presented to improve the accuracy of the SMS based QA system. Since the SMS are short in nature, system uses Soundex Similarity measure to improve the precision of the system. Various experiments were conducted to test the accuracy of various matching techniques. From the experimental results, it can be conclude that this approach is able to significantly outperform the current state-of-the-art SMS based QA system, particularly in case of out domain queries the results are more accurate. This work represents an attainment into an open and practical research domain. This proposed similarity formulation is effective, comparable, easy to compute and can serve as an alternative approach to existing similarity measures.

## 6.2. FUTURE SCOPE

### 6.2.1. N-Grams Technique

By using an N-gram count based algorithm that takes the count of various N-grams (monograms, bigrams, trigrams, etc.) into the account in order to calculate the score of the questions in the corpus. In this way, it can further improve the accuracy of the SMS based FAQ system significantly by refining the results of the system using N-gram count based scoring.

### 6.2.2 Inverse Bigram Frequency

Like Inverse domain frequency and inverse bigram frequency can be used in the pre-processing stage. This may improve the similarity score of the question hence improve the accuracy of the system.

### 6.2.3 Caching the Results

Caching the results would help the system in answering the repetitive queries. In this case, system needs not to search the FAQ in the full corpus every time instead it can first check the question similarity the cache if not found then go to the corpus. In general, it is the common to have a particular set of queries at particular time. For example, during admission time in a university, most of the queries would be related to the admission. only.

### 6.2.4 Extension of work from monolingual (English) to multilingual (English, Hindi, Malayalam, Tamil, etc)

The same work may be applied for different languages. It will increase scalability of the system.

# REFERENCES

1. R. S. Monojit Choudhury, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar and Anupam Basu, "Investigation and modeling of the structure of texting language," IJDAR, vol. 10, pp. 157-174, 2007.
2. J. Chen, et al., "SMS-Based Contextual Web Search," presented at the Mobiheld '09 Barcelona, Spain, 2009.
3. S. K. Samanta, et al., "Automatic language translation for mobile SMS," International Journal of Information Communication Technologies and Human Development (IJICTHD), vol. 2, pp. 43-58,2010.
4. TRAI Annual report - http://www.trai.gov.in/annualreport/English_Front_Page.pdf.
5. Global mobile statistics 2012 -http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats
6. ChaCha - http://www.chacha.com/
7. GovindKothari , SumitNegi, Tanveer A. Faruquie, Venkatesan T. Chakaravarthy, L. Venkata, "SMS based interface for FAQ retrieval," Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 852–860, Suntec, Singapore, 2-7 August 2009.
8. http://www.aqa.63336.com/
9. http://www.texperts.com/
10. http://www.gtip.co.uk/

11. Anwar Shaikh, Rajiv Ratn Shah, Mukul Jain, Mukul Rawat, Manoj Kumar, "Improving accuracy of SMS based FAQ retrieval," Proc. Forum for Information Retrieval Evaluation (FIRE 2011), to be published in Springer LNCS unpublished.

12. Kim H., Lee H., Seo J., A reliable FAQ retrieval system using a query log classification technique based on latent semantic analysis, Inf. Process. Manage. 43, 2007, 420-430.

13. Kim H., Seo J., Cluster-based FAQ retrieval using latent term weights. IEEE Intelligent Systems 23, 2008, 58-65.

14. Riezler S., Vasserman A., Tsochabtaridis I., Mittal V., Liu Y., Statistical machine translation for query expansion in answer retrieval. In proceedings 45th Annual Meeting of the Association of Computational Linguistic, 2007, 464-471.

15. Wu C. H., Yeh J. F., Chen M. J., Domain Specific FAQ retrieval using independent aspects, ACM Transactions on Asian Language Information Proceeding (TALIP) 4, 2005, 1-17.

16. www.**askme**.com/

17. "Handling Noisy Queries In Cross Language FAQ Retrieval" Danish Contractor Govind Kothari Tanveer A. Faruquie L. Venkata Subramaniam Sumit Negi in IBM Research India Vasant Kunj, Institutional Area New Delhi, India

18. Rudy Schusteritsch, Shailendra Rao, Kerry Rodden. 2005. Mobile Search with Text Messages: Designing the User Experience for Google SMS.

19. Sunil Kumar Kopparapu, Akhilesh Srivastava and Arun Pande. 2007. SMS based Natural Language Interface to Yellow Pages Directory. In Proceedings of the 4th international conference on mobile technology, applications, and systems and the

1st international symposium on Computer human interaction in mobile technology, pp. 558-563.

20. Xue, J. Jeon, and W.B Croft. 2008. Retrieval Models for Question and Answer Archives. In Proceedings of SIGIR, pp. 475-482.

21. Sneiders. 1999. Automated FAQ Answering: Continued Experience with Shallow Language Understanding Question Answering Systems. Papers from the 1999 AAAI Fall Symposium. Technical Report FS-99- 02, AAAI Press, pp. 97-107.

22. http://www.isical.ac.in/~fire/faq-retrieval/2012/faq-retrieval.html

23. http://en.mimi.hu/artificial_intelligence/search_algorithm.htm

24. Using XML and Databases ,W3C Standards in Practice in February 2008 by Bill Trippe, Senior Analyst Dale Waldt, Contributing Analyst

25. http://en.wikipedia.org/wiki/Stemming

26. The Porter stemming algorithm: then and now Peter Willett, Head of the Department of Information Studies, University of Sheffield, Sheffield, UK.

27. Sproat, R. (1992), *Morphology and Computation*, MIT Press, Cambridge MA.

28. http://www.comp.lancs.ac.uk/computing/research/stemming/general/index.htm

29. Lovins, Julie Beth (1968). "Development of a Stemming Algorithm". *Mechanical Translation and Computational Linguistics* 11: 22–31

30. Porter, M.F. (2005), “Lovins revisited”, In Tait, J.I. (editor) *Charting a New Course: Natural Language Processing and Information Retrieval. Essays in Honour of Karen Spärck Jones*, Springer, Dordrecht, pp. 39-68

31. Hooper, R. and Paice, C. (2005), *The Lancaster Stemming Algorithm*. Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/

32. Ahmad, F., Yusoff, M. and Sembok, T.M.T. (1996), “Experiments with a stemming algorithm for Malay words”, *Journal of the American Society for Information Science*, Vol. 47 No. 12, pp. 909-918.

33. The Soundex Phonetic Algorithm Revisited for SMS Text Representation by David Pinto et.al. Faculty of Computer Scfience Benemérita Universidad Autónoma de Puebla, Mexico.

34. Is Soundex Good Enough for You? Frankie Patman and Leonard Shaefer © 2001-2003 Language Analysis Systems, Inc. 2214 Rock Hill Road, Suite 201 Herndon, VA 20170

35. "The Soundex Indexing System". National Archives and Records Administration. 2007-05-30. Retrieved 2010-12-24.

36. http://en.wikipedia.org/wiki/Soundex

37. http://passportindia.gov.in/AppOnlineProject/online/faqCallCentre

38. A Jaccard-based Similarity Measure for Soft Sets N or Hashimah Sulaiman and Daud Mohamad Mathematics Department, Faculty of Computer Sciences and Mathematics, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, MALAYSIA, **Published in:**Humanities, Science and Engineering Research (SHUSER), 2012 IEEE Symposium, in kaula lumpur, 2012 ieee Published.

39. U.Manber, “Finding similar files in a large file system,” Proc. USENIX Conference, pp. 1–10, 1994.

40. Clustering Based On Cosine Similarity Measure, K.P.N.V.Satya sree , Dr.J V R Murthy [IJESAT] International journal of engineering science & advanced technology.

# APPENDIX A: CODING

## Login:

```
<%@page import="java.io.File"%>

<HTML>
    <HEAD>
        <TITLE>Login using jsp</TITLE>
    </HEAD>

    <BODY>
        <H1>LOGIN FORM</H1>
        <%

        String myname =  (String)session.getAttribute("username");

        if(myname!=null)
      {out.println("Welcome"+myname+",<a href=\"logout.jsp\" >Logout</a>");}
        %>
       <form action="checkLogin.jsp" method="post">
                <table>
                    <tr>
   <td> Username   : </td><td> <input name="username" size=15 type="text" /> </td>
                    </tr>
                    <tr>
  <td>Password  : </td><td><input name="password" size=15 type="password" /></td>
                    </tr>
                </table>
                <input type="submit" value="login" />
            </form>

    </BODY>
</HTML>
```

## Logout:

```
<%
     String username=(String)session.getAttribute("username");
    if(username!=null)
        {


           session.removeAttribute("username");
           out.println(" loged out, <a href=\"index.jsp\">Back</a>");

        }
     else
         {
         out.println("You are already not login <a href=\"index.jsp\">Back</a>");
     }
```

## Soundex:

```
<%@ page import="simmetrics.similaritymetrics.*" %>
<%@ page import="xmlmanip.* " %>
<%@page import="java.lang.reflect.Array"%>
<%@ page import="simmetrics.similaritymetrics.*" %>
<%@ page
import="javax.xml.parsers.DocumentBuilderFactory,javax.xml.parsers.DocumentBuilde
r,org.w3c.dom.*"%>
<%@page
import="java.io.*,org.w3c.dom.*,javax.xml.parsers.*,javax.xml.transform.*,
javax.xml.transform.dom.*,javax.xml.transform.stream.*"%>
<%
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse("http://localhost:8080/faqsearch/abhi.xml");
NodeList docu = doc.getElementsByTagName("doc");
    Soundex sn=new Soundex();
    String first=request.getParameter("query");
    boolean [] z=new boolean[100];
    boolean [] x=new boolean[100];
    String [] replace=first.split(" ");
    String [] arr=first.split(" ");
    int num= arr.length;
  /* replacing with dictionary word*/
   for(int i=0;i<arr.length;i++)
    {
    if(arr[i].equalsIgnoreCase("wat")||arr[i].equalsIgnoreCase("wt"))
          {
                arr[i]="what";
          }else if(arr[i].equalsIgnoreCase("wid"))
          {
                arr[i]="with";
          }else if(arr[i].equalsIgnoreCase("&"))
          {
                arr[i]="and";
          }
          else if(arr[i].equalsIgnoreCase("c"))
          {
                arr[i]="see";
          }else if(arr[i].equalsIgnoreCase("xplain"))
          {
                arr[i]="explain";
          }
          else if(arr[i].equalsIgnoreCase("xtrnl"))
          {
                arr[i]="external";
          }
          else if(arr[i].equalsIgnoreCase("partcpnt"))
          {
                arr[i]="participant";
          }
          else if(arr[i].equalsIgnoreCase("pcc"))
          {
                arr[i]="Pcc";
          }
          else if(arr[i].equalsIgnoreCase("shud"))
          {
                arr[i]="should";
          }
          else if(arr[i].equalsIgnoreCase("b4"))
          {
                arr[i]="before";
          }else
if(arr[i].equalsIgnoreCase("whos")||arr[i].equalsIgnoreCase("whse"))
          {
                arr[i]="whose";
          }else if(arr[i].equalsIgnoreCase("nt"))
          {
                arr[i]="not";
          }else if(arr[i].equalsIgnoreCase("den"))
          {
                arr[i]="then";
          }
          else if(arr[i].equalsIgnoreCase("dey"))
          {
                arr[i]="they";
          }else if(arr[i].equalsIgnoreCase("dis"))
          {
                arr[i]="this";
          }else if(arr[i].equalsIgnoreCase("ny"))
          {
                arr[i]="any";
          }else
if(arr[i].equalsIgnoreCase("dere")||arr[i].equalsIgnoreCase("thr"))
          {
                arr[i]="there";
          }else if(arr[i].equalsIgnoreCase("dat"))
          {
                arr[i]="that";
          }else if(arr[i].equalsIgnoreCase("m"))
          {
                arr[i]="am";
          }else if(arr[i].equalsIgnoreCase("ma"))
          {
                arr[i]="my";
          }else if(arr[i].equalsIgnoreCase("r"))
          {
                arr[i]="are";
          }else if(arr[i].equalsIgnoreCase("whr"))
          {
                arr[i]="where";
          }else if(arr[i].equalsIgnoreCase("ur"))
          {
                arr[i]="your";
          }else if(arr[i].equalsIgnoreCase("i'hv"))
          {
                arr[i]="i have";
          }else if(arr[i].equalsIgnoreCase("wen"))
          {
                arr[i]="when";
          }else if(arr[i].equalsIgnoreCase("cud"))
          {
                arr[i]="could";
          }else if(arr[i].equalsIgnoreCase("dan"))
          {
                arr[i]="than";
          }else if(arr[i].equalsIgnoreCase("e.g."))
          {
                arr[i]="example";
          }else if(arr[i].equalsIgnoreCase("no."))
          {
                arr[i]="number";
          }else if(arr[i].equalsIgnoreCase("nw"))
          {
                arr[i]="now";
          }else if(arr[i].equalsIgnoreCase("wud"))
          {
                arr[i]="would";
          }
    }
    /*code for matching using soundex algorithm and replacing with soundex
generated values. */
    for(int i=0;i<=num-1;i++)
{%>
<%
for (int j=0;j<=docu.getLength()-1;j++)
{
     String id= docu.item(j).getAttributes().getNamedItem("id").getNodeValue();
     String s=docu.item(j).getFirstChild().getNodeValue();
     String [] arr2=s.split(" ");
     int num1=arr2.length;
     for(int k=0;k<=num1-1;k++)
     {
          if(arr2[k].equalsIgnoreCase(arr[i]))
          {z[i]=true;}
          float n=sn.getSimilarity(arr2[k], arr[i]);
          // calculating soundex match.
// out.println(arr[i]+":"+arr2[k]+sn.getSimilarity(arr2[k],arr[i])+z[i]);
          if (n>=0.9)
          {replace[i]=arr2[k];}}}
if(!z[i])
{arr[i]=replace[i];
}}
String strn="";
for(int m=0;m<=arr.length-1;m++)
{strn=strn+" "+arr[m];}
out.print(strn);
%>
<script>location.href="testprocess.jsp?query=<% out.print(strn);%>"</script>
<!--  --script>location.href="testprocess.jsp?query=<%
out.print(strn);%>"</script-->
%>
```

## Test Process:

```
<%@page import="javax.activation.MailcapCommandMap"%>
<%@ page import="xmlmanip.* " %>
<%@page import="java.lang.reflect.Array"%>
<%@ page import="simmetrics.similaritymetrics.*" %>
<%@ page
import="javax.xml.parsers.DocumentBuilderFactory,javax.xml.parsers.DocumentBuilde
r,org.w3c.dom.*"%>
<%@page
import="java.io.*,org.w3c.dom.*,javax.xml.parsers.*,javax.xml.transform.*,
javax.xml.transform.dom.*,javax.xml.transform.stream.*"%>
<a href="admindashbord.jsp" >Home</a>&nbsp;&nbsp;&nbsp;<a href="test.jsp" >Back
to query</a>&nbsp;&nbsp;&nbsp;<a href="logout.jsp" >Logout</a>
<%
String query = request.getParameter("query");
//double max=0;
String maxid[]={"","",""};
String maxidj[]={"","",""};
String maxidc[]={"","",""};
/* value of query string will be removing prepositions and other stop words
here..*/
remStop m=new remStop();
String []value=m.removeStop(query);
out.println(value[0]+"/"+value[1]);

/* value of query string will be stemmed here..*/
StringStemmer stm= new StringStemmer();
String ne= stm.Stemmer(value[1]);

/*reading the stemkey.xml file for similiarity measure*/

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse("http://localhost:8080/faqsearch/stemkey.xml");
doc.normalizeDocument();
NodeList docu = doc.getElementsByTagName("doc");

Document pre = db.parse("http://localhost:8080/faqsearch/prep.xml");
pre.normalizeDocument();
NodeList pre1 = pre.getElementsByTagName("doc");


JaccardSimilarity jc=new JaccardSimilarity();
CosineSimilarity cs=new CosineSimilarity();
QGramsDistance gd=new QGramsDistance();%>

<table border="2px">
<tr style="background-color: green"><td>Question
id</td><td>Jaccard</td><td>Cosine</td><td>{Jaccard +
Cosine}*1000</td><td>qgram</td><td>Prep-Jaccard</td><td>Prep-Cosine</td><td>Prep-
qgram</td></tr>
<%
double maximum[]={0,0,0};
double maximumj[]={0,0,0};
double maximumc[]={0,0,0};

for (int i=0;i<=pre1.getLength()-1;i++){

    String s=docu.item(i).getFirstChild().getNodeValue();
    String st=pre1.item(i).getFirstChild().getNodeValue();




    double l=jc.getSimilarity(s, ne);
    double t=cs.getSimilarity(s, ne);
    double n=gd.getSimilarity(s, ne);
    double l1=jc.getSimilarity(st, value[0]);
    double t1=cs.getSimilarity(st, value[0]);
    double n1=gd.getSimilarity(st, value[0]);
    double a=l+0.2*l1;
    double b=t+0.2*t1;
    double d=n+0.2*n1;
    double c=1000*(a+b+d);
    /* if(c>max)
    {max=c;

    maxid=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();

    } */
    if(c>maximum[0])
    {
    double temp1=maximum[0];
    String tempid1=maxid[0];
    double temp2=maximum[1];
    String tempid2=maxid[1];
    maximum[0]=c;
    maximum[1]=temp1;
    maximum[2]=temp2;
    maxid[0]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
    maxid[1]=tempid1;
    maxid[2]=tempid2;
    }
    else if(c>maximum[1])
    {
    double temp1=maximum[1];
    String tempid1=maxid[1];
    maximum[1]=c;
    maximum[2]=temp1;
    maxid[1]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
    maxid[2]=tempid1;
    }
    else if(c>maximum[2])
    {
    maximum[2]=c;
    maxid[2]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
    }
    //for jaccard
```

```
if(a>maximumj[0])
{

double temp1=maximumj[0];
String tempid1=maxidj[0];
 double temp2=maximumj[1];
 String tempid2=maxidj[1];
 maximumj[0]=a;
 maximumj[1]=temp1;
 maximumj[2]=temp2;
 maxidj[0]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
 maxidj[1]=tempid1;
 maxidj[2]=tempid2;
}
else if(a>maximumj[1])
{
double temp1=maximumj[1];
String tempid1=maxidj[1];
maximumj[1]=a;
maximumj[2]=temp1;
maxidj[1]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
maxidj[2]=tempid1;
}
else if(a>maximumj[2])
{
maximumj[2]=a;
maxidj[2]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
}
// for cosine
if(b>maximumc[0])
{

double temp1=maximumc[0];
String tempid1=maxidc[0];
 double temp2=maximumc[1];
 String tempid2=maxidc[1];
 maximumc[0]=b;
 maximumc[1]=temp1;
 maximumc[2]=temp2;
 maxidc[0]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
 maxidc[1]=tempid1;
 maxidc[2]=tempid2;
}
else if(b>maximumc[1])
{
double temp1=maximumc[1];
String tempid1=maxidc[1];
maximumc[1]=b;
maximumc[2]=temp1;
maxidc[1]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
maxidc[2]=tempid1;
}
else if(b>maximumc[2])
{
maximumc[2]=b;
```

```
    maxidc[2]=docu.item(i).getAttributes().getNamedItem("id").getNodeValue();
    }

   %>
   <tr>
   <td><%= docu.item(i).getAttributes().getNamedItem("id").getNodeValue()%></td>
   <td><% out.println(l);%></td>
   <td><% out.println(t);%></td>
   <td><% out.println(c);%></td>
   <td><% out.println(n);%></td>
   <td><% out.println(a);%></td>
   <td><% out.println(b);%></td>
   <td><% out.println(d);%></td>

   <%}%></tr>
</table>
<div style="background-color: green">For hybrid similarity: </div>
<div style="background-color: green">Question Id with Maximum probability:<%
out.println(" "+maxid[0]+","+maxid[1]+","+maxid[2]); %></div>
<div style="background-color: green">Similarity Value of the question:<%
out.println(" "+maximum[0]+","+maximum[1]+","+maximum[2]); %></div>

<%
Document doc1 = db.parse("http://localhost:8080/faqsearch/faq.xml");
NodeList doc2 = doc1.getElementsByTagName("doc");
NodeList question = doc1.getElementsByTagName("question");
NodeList ans = doc1.getElementsByTagName("ans");
for (int j=0;j<=doc2.getLength()-1;j++)
{
String qid=doc2.item(j).getAttributes().getNamedItem("id").getNodeValue();
for(int t=0;t<3;t++){
if(qid.equals(maxid[t]))
{
    out.println("Question"+qid+":");
    String quest=question.item(j).getFirstChild().getNodeValue();
    out.println(quest+"<br/>");
    out.println("Answer:");
    String answer=ans.item(j).getFirstChild().getNodeValue();
    out.println(answer+"<br/><br/><br/>");

    }
}
}

// jaccard display
out.println("Jaccard Based result<br/><br/><br/>");
%>
<div style="background-color: green">For Jaccard:</div>
<div style="background-color: green">Question Id with Maximum probability:<%
out.println(" "+maxidj[0]+","+maxidj[1]+","+maxidj[2]); %></div>
<div style="background-color: green">Similarity Value of the question:<%
out.println(" "+maximumj[0]+","+maximumj[1]+","+maximumj[2]); %></div>

<%
for (int ja=0;ja<=doc2.getLength()-1;ja++)
```

```
{
String qidj=doc2.item(ja).getAttributes().getNamedItem("id").getNodeValue();

for(int t=0;t<3;t++){
if(qidj.equals(maxidj[t]))
{
    out.println("Question"+qidj+":");
    String quest=question.item(ja).getFirstChild().getNodeValue();
    out.println(quest+"<br/>");
    out.println("Answer:");
    String answer=ans.item(ja).getFirstChild().getNodeValue();
    out.println(answer+"<br/><br/><br/>");

    }
}
}

//cosine display



out.println("Cosine Based Result<br/><br/><br/>");
%>
<div style="background-color: green">For Cosine: </div>
<div style="background-color: green">Question Id with Maximum probability:<%
out.println(" "+maxidc[0]+","+maxidc[1]+","+maxidc[2]); %></div>
<div style="background-color: green">Similarity Value of the question:<%
out.println(" "+maximumc[0]+","+maximumc[1]+","+maximumc[2]); %></div>

<%
for (int js=0;js<=doc2.getLength()-1;js++)
{
String qidc=doc2.item(js).getAttributes().getNamedItem("id").getNodeValue();

for(int t=0;t<3;t++){
if(qidc.equals(maxidc[t]))
{
    out.println("Question"+qidc+":");
    String quest=question.item(js).getFirstChild().getNodeValue();
    out.println(quest+"<br/>");
    out.println("Answer:");
    String answer=ans.item(js).getFirstChild().getNodeValue();
    out.println(answer+"<br/><br/><br/>");

    }
}}

%>
```

## Upload:

```
<a href="admindashbord.jsp">Home</a>
<%
      String saveFile = "";
      String contentType = request.getContentType();
      if ((contentType != null) && (contentType.indexOf("multipart/form-data") >=
0)) {
            DataInputStream in = new DataInputStream(request.getInputStream());
            int formDataLength = request.getContentLength();
            byte dataBytes[] = new byte[formDataLength];
            int byteRead = 0;
            int totalBytesRead = 0;
            while (totalBytesRead < formDataLength) {
                  byteRead = in.read(dataBytes, totalBytesRead, formDataLength);
                  totalBytesRead += byteRead;
            }
            String file = new String(dataBytes);
            saveFile = file.substring(file.indexOf("filename=\"") + 10);
            saveFile = saveFile.substring(0, saveFile.indexOf("\n"));
                 saveFile = saveFile.substring(saveFile.lastIndexOf("\\") + 1,
                 saveFile.indexOf("\""));
            int lastIndex = contentType.lastIndexOf("=");
                  String boundary = contentType.substring(lastIndex+
                 1,contentType.length());
            int pos;
            pos = file.indexOf("filename=\"");
            pos = file.indexOf("\n", pos) + 1;
            pos = file.indexOf("\n", pos) + 1;
            pos = file.indexOf("\n", pos) + 1;
            int boundaryLocation = file.indexOf(boundary, pos) - 4;
            int startPos = ((file.substring(0, pos)).getBytes()).length;
            int endPos = ((file.substring(0,
                 boundaryLocation)).getBytes()).length;
            saveFile = "D:/eclipse/java/WebContent/" + saveFile;
            File ff = new File(saveFile);
            FileOutputStream fileOut = new FileOutputStream(ff);
            fileOut.write(dataBytes, startPos, (endPos - startPos));
            fileOut.flush();
            fileOut.close();
%><Br>
<table border="2">
      <tr>
            <td><b>You have successfully upload the file by the name of:</b>
            <%
                  out.println(saveFile);
                  }
            %>
            </td>
      </tr>
</table>
<%@include file="xmlparse.jsp" %>
<%@include file="keygen.jsp" %>
<%@include file="Stemmer.jsp" %>
```

## XML Parsing:

```
<%@ page contentType="text/html"%>
<%@ page
import="javax.xml.parsers.DocumentBuilderFactory,javax.xml.parsers.DocumentBuilde
r,org.w3c.dom.*"%>
<%@page
import="java.io.*,org.w3c.dom.*,javax.xml.parsers.*,javax.xml.transform.*,
javax.xml.transform.dom.*,javax.xml.transform.stream.*"%>
<a href="admindashbord.jsp">Home</a>
<%
String str="abhi";
DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = builderFactory.newDocumentBuilder();
Document doc1 = docBuilder.newDocument();
// questionset elements
Element rootElement = doc1.createElement("queryset");
doc1.appendChild(rootElement);

/*Reading faq.xml*/

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse("http://localhost:8080/faqsearch/faq.xml");
NodeList docu = doc.getElementsByTagName("doc");
NodeList question = doc.getElementsByTagName("question");
NodeList ans = doc.getElementsByTagName("ans");
%>
<html>
<head>
<meta></meta>
<title>XML with JSP</title>
</head>
<body>
<table border=0 bgcolor="green" width=80%  cellspacing=2 cellpadding=2
align=center>
<tr>
  <td style="font-family:arial,verdana;font-size:9pt;">
<font color=white face="courier new" face=1><strong>Reading XML Data from
JSP</strong></font>
        </td>
</tr>
</table>
<br>
<table border=0 width=80% style="border: thin groove white;" cellspacing=2
cellpadding=2 align=center>
<tr>
<td bgcolor="green"><font color=white face="courier new" face=1>Document
No</font></td>
         <td bgcolor="green"><font color=white face="courier new"
face=1>Question</font></td>
         <td bgcolor="green"><font color=white face="courier new"
face=1>Ans</font></td>
</tr>
```

```
<%int i;
  for (i=0;i<=docu.getLength()-1;i++)
      {
   // question elements
     Element document = doc1.createElement("doc");
     rootElement.appendChild(document);
     // Merging question and answer.
     //document.setTextContent(question.item(i).getFirstChild().getNodeValue()+"
"+ans.item(i).getFirstChild().getNodeValue());
     document.setTextContent(question.item(i).getFirstChild().getNodeValue());
     // set attribute to doc element
     Attr attr = doc1.createAttribute("id");

attr.setValue(docu.item(i).getAttributes().getNamedItem("id").getNodeValue());
     document.setAttributeNode(attr);%>
                 <tr>
                 <td>
       <font  face="courier new" face=1>
 <%= docu.item(i).getAttributes().getNamedItem("id").getNodeValue() %>
                                       </font>
                         </td>
                         <td>
 <font face="courier new" face=1>
 <%= question.item(i).getFirstChild().getNodeValue()%>
                         </font>
                         </td>
                         <td>
                             <font face="courier new" face=1>
                                       <%=
ans.item(i).getFirstChild().getNodeValue()%>
                                       </font>
                         </td>
                 </tr>
          <%}

  TransformerFactory factory = TransformerFactory.newInstance();
  Transformer transformer = factory.newTransformer();

  transformer.setOutputProperty(OutputKeys.INDENT, "yes");

  StringWriter sw = new StringWriter();
  StreamResult result = new StreamResult(sw);
  DOMSource source = new DOMSource(doc1);
  transformer.transform(source, result);
  String xmlString = sw.toString();

  File file=new File("D:/eclipse/java/WebContent/"+str+".xml");
  BufferedWriter bw = new BufferedWriter(new FileWriter(file));
  bw.write(xmlString);
  bw.flush();
  bw.close(); %>
</table>
</body>
</html>
```

## Upload XML:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<HTML>
<HEAD>
<TITLE>file upload</TITLE>
</HEAD>
<BODY>
<div align="center"><font size="20px">Upload Faq xml file</font></div>
<div align="center">
<a href="admindashbord.jsp">Home</a>
<FORM ENCTYPE="multipart/form-data" ACTION="upload.jsp" METHOD=POST>
<br />
<br />
<br />

<table border="0" bgcolor=#ccFDDEE>
      <tr>

            <td colspan="2" align="center"><B>UPLOAD THE FILE</B>

            </td>
      </tr>
      <tr>
            <td colspan="2" align="center"></td>
      </tr>
      <tr>
            <td><b>Choose the file To Upload:</b></td>
            <td><INPUT NAME="file" TYPE="file"></td>
      </tr>
      <tr>
            <td colspan="2" align="center"></td>
      </tr>
      <tr>
            <td colspan="2" align="center"><input type="submit"
                  value="Send File"></td>
      </tr>
      </table>

            </FORM>
            </div>
</BODY>
</HTML>
```

## Jaccard:

```
Package simmetrics.similaritymetrics;

import simmetrics.tokenisers.InterfaceTokeniser;

import simmetrics.tokenisers.TokeniserWhitespace;

import java.util.HashSet;

import java.util.Set;

import java.util.ArrayList;

import java.io.Serializable;


public final class JaccardSimilarity extends AbstractStringMetric implements

Serializable {

   private final float ESTIMATEDTIMINGCONST = 1.4e-4f;

   public JaccardSimilarity() {

      tokeniser = new TokeniserWhitespace();

   }

   public JaccardSimilarity(final InterfaceTokeniser tokeniserToUse) {

      tokeniser = tokeniserToUse;

   }

   public String getShortDescriptionString() {

      return "JaccardSimilarity";

   }

   public String getLongDescriptionString() {
        return "Implements the Jaccard Similarity algorithm providing a similarity
measure between two strings";
    }
    public String getSimilarityExplained(String string1, String string2) {
        return null;
    }
    public float getSimilarityTimingEstimated(final String string1, final String string2) {
        final float str1Tokens = tokeniser.tokenizeToArrayList(string1).size();
        final float str2Tokens = tokeniser.tokenizeToArrayList(string2).size();
        return (str1Tokens * str2Tokens) * ESTIMATEDTIMINGCONST;
    }
  //todo this needs checking
        final ArrayList<String> str1Tokens = tokeniser.tokenizeToArrayList(string1);
        final ArrayList<String> str2Tokens = tokeniser.tokenizeToArrayList(string2);

        final Set<String> allTokens = new HashSet<String>();
        allTokens.addAll(str1Tokens);
        final int termsInString1 = allTokens.size();
        final Set<String> secondStringTokens = new HashSet<String>();
        secondStringTokens.addAll(str2Tokens);
        final int termsInString2 = secondStringTokens.size();

        //now combine the sets
        allTokens.addAll(secondStringTokens);

        final int commonTerms = (termsInString1 + termsInString2) - allTokens.size();

//return JaccardSimilarity

        return (float) (commonTerms) / (float) (allTokens.size());

    }

public float getUnNormalisedSimilarity(String string1, String string2) {

        return getSimilarity(string1, string2);

    }

}
```

## COSINE:

```
package simmetrics.similaritymetrics;

import simmetrics.tokenisers.InterfaceTokeniser;

import simmetrics.tokenisers.TokeniserWhitespace;

import java.util.HashSet;

import java.util.Set;

import java.util.ArrayList;

import java.io.Serializable;


public final class CosineSimilarity extends AbstractStringMetric implements

Serializable

{private final float ESTIMATEDTIMINGCONST =

0.00000038337142857142857142857142857142f;

  private final InterfaceTokeniser tokeniser;

  public CosineSimilarity() {

    tokeniser = new TokeniserWhitespace();

  }

  public CosineSimilarity(final InterfaceTokeniser tokeniserToUse) {

    tokeniser = tokeniserToUse;

  }

  public String getSimilarityExplained(String string1, String string2) {

  }

  public float getSimilarityTimingEstimated(final String string1, final String string2) {
    final float str1Length = string1.length();
    final float str2Length = string2.length();
    return (str1Length + str2Length) * ((str1Length + str2Length) *
ESTIMATEDTIMINGCONST);
  }
  public float getSimilarity(final String string1, final String string2) {
    final ArrayList<String> str1Tokens = tokeniser.tokenizeToArrayList(string1);
    final ArrayList<String> str2Tokens = tokeniser.tokenizeToArrayList(string2);

    final Set<String> allTokens = new HashSet<String>();
    allTokens.addAll(str1Tokens);
    final int termsInString1 = allTokens.size();
    final Set<String> secondStringTokens = new HashSet<String>();
    secondStringTokens.addAll(str2Tokens);
    final int termsInString2 = secondStringTokens.size();
    allTokens.addAll(secondStringTokens);
    final int commonTerms = (termsInString1 + termsInString2) - allTokens.size()
    return (float) (commonTerms) / (float) (Math.pow((float) termsInString1, 0.5f) *
Math.pow((float) termsInString2, 0.5f));
  }
  public float getUnNormalisedSimilarity(String string1, String string2) {
    return getSimilarity(string1, string2);
  }
```

# APPENDIX B: SOUNDEX ALGORITHM

**Soundex Match**

**Soundex Algorithm** [*O'dell, Russel*]

a) Retain first letter of the word and remaining
letters are replaced by their codes

b) For the consecutive occurrence of the same
digit, drop all but the first

c) Drop all '0's

d) Convert to the form 'letter digit digit digit'
by dropping right most digits (if there are
more than three digits) or by adding trailing
zeroes (if there are less than three digits)

**Letter Code**

Instead of restricting to code size to **4** , we have taken the full
code i.e., the step d) is modified as

*d') Convert to the form 'letter digit digit …… '*