**Summer Internship Report**

**Introduction**

India is expected to have 236 million mobile Internet users by 2016 : IAMAI-KPMG report.

Internet is usually used by layman (common people) to ask some common questions (who's answers should be ready and available). But even most efficient search engines give their users a hard time to find apt answers. So fast and efficient FAQ(full form) retrieval system can solve the problem.

The database used in this system is the Forum for Information Retrieval Evaluation (FIRE) database, which is a standard database used for RND which contains many questions from various domains.\*\*\* describe something about FIRE.

Our primary objective is to retrieve the most suitable question from our corpora of FAQs.

Top 5 questions will be provided to the user then he can select the most accurate one.

Knowing that our users wont spare time correcting their English we assume that our queries will be in “SMS language”. SMS language is a form of language used in communication which isn't grammatically correct (basically used by user to compress his text and type faster), hence the queries will be noisy,that is it will contain slang words,spelling mistakes ,abbreviations and SMS limitations -160 characters -Google).

These queries may also contain other unintended errors which are caused due to typing. The presence of such noise will make answering such questions hard .

***So our primary challenge is to de-noise the SMS query entered by our user and map it to the most appropriate question on our database (if there exists a match).***

The system is mainly focused on monolingual FAQ retrieval and divided into ….. modules.

**Module 1:** Preprocessing ---

In this module, we have arranged our corpus in such a way to make our processing easier.

Calculating stop-words\*.

Making slang dictionary.

Maintaing own corrected word database.\*

**Module 2:** Processing SMS query---

Refining question to ease processing.

Break down statement into tokens(noisy words).

Removing slang words.

Processing each word and according returning most accurate answer from corpus.

Further processing is done in a sequence of two steps …...

The paper is organized into following sections:(divide in sections)

**Literature review**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Problem Statement**

The database contains a set of frequently asked questions and their respective answers from various domains. These questions in the database will be represented as Q.

The SMS query entered may be noisy. The goal is to find a question Q\* from this question set Q that is the best possible match for the SMS query S.

For this we have used various techniques like Longest Common Subsequence,word-question ratio,removal of stop words,removal of slang words etc..Removal of stop-words from query and form our dataset questions while processing will avoid any unintended match of key words.

Each match of the keyword is assigned weight according to a values obtained by various techniques based upon general observations .The question having maximum keywords will have a high score. For this techniques like Longest Common Subsequence are used.

Questions containing these keywords will have high Question Similarity score.

These questions Q` are narrowed down to new set containing top matches according to Question Similarity score.

These questions Q` are taken and further checked for key words contained and sequence of keywords contained.

Combining these two scores will give the best possible match/matches Q\*.

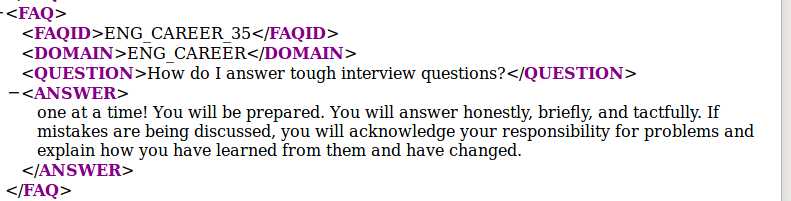
**Dataset used for SMS based FAQ retrieval**

The “SMS based FAQ Retrieval” task\* of FIRE ….... the datasets used as the training set for our system.

There are a total of...............(statistics of the database).

The format of the data is in XML and is shown in fig.....

**Table .** Sample XML-formatted training data.



**Implementation**

**.1 Preprocessing**

In this section, we have arranged our corpus in such a way to make our processing easier.

Hence, reducing work required to process our SMS query.

A list of stop-words\* is created.

A slang list is created manually.

**.2 Calculating weight of each word**

For each keyword *(s)* weight is calculated corresponding to (w) word in corpus. The weight is given by equation:

Weight (w,s)=LCSR(w,s)\*S.M Ratio(w,s)\*IDF(w)

LevDistance(w,s)

Where,

LCSR(w,s)-Longest Common Subsequence Ratio of words s and w

S.M Ratio(w,s)-Similarity Ratio using Ratcliff/Obershelp algorithm

IDF(w)-Inverse Document Frequency of w.

LevDistance(w,s)-Levenshtein Distance between w and s.

**Longest Common Subsequence Ratio**

The longest common sub-sequence(LCS) is to find the longest subsequence common to our entered words. A subsequence in a sequence that can be derived from removing few elements without changing the order of the the remaining characters.

Longest Common Subsequence Ratioof the keyword entered in our query (s) and the word present in our database(w) is the ratio of their LCS to the maximum length of the two.



Example:............................

**Similarity Ratio**



**Inverse Document Frequency**

If f number of documents in corpus in Q contain a term w and the total number of documents in Q is N, the Inverse Document Frequency(IDF) of w in W.



This means that a word which occurs less number of times in th corpus Q

will have a high IDF. The reason behind this logic is that queries are composed of more informative words.

We have used hash-table to store IDF for each word in W.

**Levenshtein Distance**

Levenshtein distance is a 'distance' between two strings is the minimum number of changes that has to be performed on one string so that it is transformed into the other.

These operations may are defined as insertion, deletion or substitution of a single character,or transposition of two charecters.

Example:............................

**\*Programming perspective:**

The weight of only those words were calculated whose first alphabet or last alphabet matches the words in our databases and also containing LCS is greater than 30% of the word in database so that number of computations are reduced.

All matches for our keyword are then arranged in descending order of their weights and stored. This avoids re-computation of already calculated weights.

Weight of stop words are not included.

Slang words are replaced from entered query.

**.3 Generating a Question List**

In order to calculate Question Similarity Score for questions in Q, we create a hash-table for each question having values set to zero.

Each question is checked for any possible matches for the key words and weight of the key word is added to the question corresponding to the ratio of the word's length to th length of th sentence.

length(question)

Weight(question)=Weight(question)+ length(w)

Hence, increasing question weight as the number of keywords matched on each sentence increases. Then the Questions are arranged according to decreasing order of their weight and top 25 questions are sent for further evaluation.

**\*Programming perspective:**

The questions hashed from database were checked if their length (excluding spaces) has length less than 400 % of length our entered query and length greater 75 % of our entered query to minimize computations.

**.4 Generating Similarity Score**

The top 25 questions are then checked for sequence of keywords .

This is done in two parts:

**Part 1:**(generating list1)

Each question is then checked for number of keywords that it contains hence giving it a new score for keywords (keyword score- k1). If the word contained in a question is a possible match to any of the keywords of the query the Keyword Score is incremented.

…..sudo code

**Part 2:**(generating list2)

Now, each question is checked whether these keywords occur in same sequence as in our SMS query.

For this an algorithm similar to LCS, but is applied to words. The keyword score- k2 is calculated for each question

…..sudo code

Finally scores from both parts are added for each individual question.

Score(question)=l1(question)+ l2(question)

**\*Programming perspective:**

The questions having weight below ….. we initially screened out as they are those which contain questions containing word/words which have little correlation to our entered keyword.

**.5 Generating best possible match**

The final score list is then checked for maximum score entries.

If there is only one entry that has the maximum score then it is taken as

the answer .Else the entries with maximum keyword score are further checked keeping their weight as criteria.

The best possible match is then returned.

**Experiments and their results.**

**Advantage over other implementations.**

The implementation just discussed is much simpler to implement and produces …% accurate results (in-domain).

The main advantage of this approach is that only one response is returned to the user.

The system has been tested for a wide set of testcases and proves to be verry efficient for in-domain problems.

…......show stats with comparison form other papers

**Further work**

The main drawback of this FAQ system is that it tries to infer from a given set of keywords and searches given domains. The following steps will be taken to improve system performance:

**\*adding a spell checker**- this may help the system to actually know what the user is and wants to know.

Once the question has been corrected its domain can be identified hence making our search even easier.

**\*rearrange questions domain wise**- this will help us arrange our list of possible questions quicker.

**\*improve speed-** improving overall system performance by parallelizing the code.

**References**