A TF-IDF Method for Automatic Query Answering

[1]Anju K S, [2]Neethu George, [3]Surekha Mariam Varghese
[1]Student, [2]Student, [3]Professor
[1]Computer Science and Engineering,
[1] Mar Athanasius College of Engineering, Kothamangalam, India

***Abstract*: A Fully functional query answering system has been quite popular among researchers. In this paper the design and implementation of an automatic query answering system is presented. In the proposed system, queries are given as input to the model and all related questions and possible answers related to that query above a threshold value is retrieved. In this approach the input query given is first preprocessed for normalization and standardization of data. Each questions and their answers are considered as individual documents with each questions linked with corresponding answer (answers are not preprocessed). Tokenization is performed on data and TF-IDF(Term Frequency-Inverse Document Frequency) is used to generate representation vectors of documents. Input query is also converted to its vector form. In the proposed system, cosine similarity is used to find the similarity between the input query with the datas found in the dataset. This information can be used to give instantaneous reply to queries. The system tested and implemented successfully on kissan call centre data.**

***Index Terms*: tokenization, TF-IDF, Matrix, cosine, similarity,**

I. INTRODUCTION

The task of retrieving most relevant data for a user defined query has become so common and natural in recent years. However, this growing use of query retrieval warrants continued research and enhancements to generate better solutions to the problem. Informally, query retrieval can be described as the task of searching a collection of data, be that text documents, databases, networks, etc., for specific instances of that data. In recent years, the Query Answering systems have acquired very special place as they are very useful in allowing users to enter query based on their needs or point of view and the system tries to answer the queries effectively instead of just giving out the best suited keywords. First, some limit is set for searching a collection of certain documents. The refined problem then becomes the task of searching this corpus for documents that the query retrieval system considers relevant to what the user entered as the query. Suppose, if there are N text documents. When the user inputs a query to the system, system replies with the most similar documents from the database. In this approach the input query is in the form of short texts.

The input query given by the user is converted to its vector form to segregate semantic similarity effectively. Here TF-IDF method is used to convert queries to its vector form and cosine similarity is used to compare two vectors. Vectored queries by TF-IDF is very important to point specific textual similarity to segregate different contextual information. This is required since different person can ask same questions on different context, so main aim of this project is to identify all such similar text segments which mean almost the same, this will increase efficiency and accuracy of the model.

II. RELATED WORKS

Several NLP models are currently used in the field of question answering. One of the major models is [3s]. Most current Question Answering systems attempt to find answers to input questions by cleverly selecting portions of related documents. In this model, an alternative approach for taking advantage of the large amount of pre-answered questions available on the web and finding a similar question that has already been answered is discussed. This approach involves query transformation as breadth-first search with expansion by Wordnet and pruning by language model, in order to transform the question into the language of the question corpus. Combination of NLP and IR techniques are used to return the most relevant pre-answered question. In [4] POS tagging concept is proposed in order to match the query with questions already present in Yahoo Answers. One major issue with this approach is that not all the questions may be present on the internet and as a result such questions may not be answered. In work in [5] to retrieve all the relevant sentences from the text, a score is given to each of the sentences. Then to solve the problem, then a similarity matching is done between the sentences thus retrieved and user queries. This proves to be a fairly good approach but the answers are not always appropriately framed in order to solve the query.

III. DATA COLLECTION AND PREPROCESSING

Dataset for the system is prepared manually by the information collected from official site of Kissan Call Centre. For the portal we collected data for year 2015, 2016 and 2017. Data collection is done from 4 southern states including Kerala, Tamilnadu, Karnataka and Andra Pradesh. The dataset created manually is in the form of text files. Each questions and their answers are considered as individual documents with each questions linked with corresponding answer. Here only the questions are preprocessed and converted to vector and not the answer.

IV. SYSTEM MODEL AND METHODOLOGIES

The most common text mining approach involves a representation of text that is based on keywords. A keyword based methodology can be combined with other statistical elements (machine learning and pattern recognition techniques, for example) to discover

relationships between different elements in text by recognizing repetitive patterns present in the content**.** These approaches do not attempt to understand language, and may only retrieve relationships at a superficial level**.** The query text is in the form of short text segments and with different contexts. The query text is first preprocessed for relevant data. The preprocessed data is then converted to its vector representation using TF-IDF method.

An Overview of TF-IDF

To examine the structure and implementation of TF-IDF for a set of documents, mathematical background of the algorithm is introduced first. A quick informal explanation of TF-IDF is given before proceeding. TF-IDF stands for term frequency-inverse document frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining. Essentially, TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TFIDF numbers than common words such as articles and prepositions.

Typically, the TF-IDF weight is composed by two terms: the first term computes the normalized Term Frequency (TF) which is defined as the number of times a word appears in a document divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- **TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).

Consider a document containing 100 words where in the word *paddy* appears 3 times. The term frequency (i.e., TF) for *paddy* is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word paddy appears in one thousands of these. Then, the inverse document frequency (i.e., IDF) is calculated as log (10,000,000 / 1,000) = 4. Thus, the TF-IDF weight is the product of these quantities: 0.03 * 4 = 0.12.

Cosine Similarity

Cosine similarity technique is used to find the similarity of the extracted feature vector with the available data set and the most appropriate remedy to the problems will be chosen in such a way that the answers are retrieved from the data set with queries having higher cosine similarity value with the feature vector. When the farmer enters the query, the system finds all the similar queries and displays the answers associated to those queries. For example, if the farmer give a query as "*paddy disease*" the query text will find the similar queries relating to "*paddy*", "*disease*", and "*paddy disease*". In this system cosine similarity is used for the evaluation of similarity measure for the query vectors with similarity matrix vectors.

The table 1 is the result of query "paddy disease". Here the query with high cosine similarity value will be selected and answer corresponding to that query will be displayed. The number of similar queries that is to be displayed is already set. For example if the limit is set to 3, then the three top most similar queries will be selected and their answers will be displayed.

Table 1. Evaluation of similarity

Query text	Query from data file	Cosine Similarity index
Paddy Disease	How to control white or yellow leaf of **paddy** ?	0.2601
Paddy Disease	How to control fungal **disease** in **paddy**?	0.643561

This system consists of mainly three modules. They are:-

1. Data preprocessing
2. Vector generation
3. Similarity evaluation

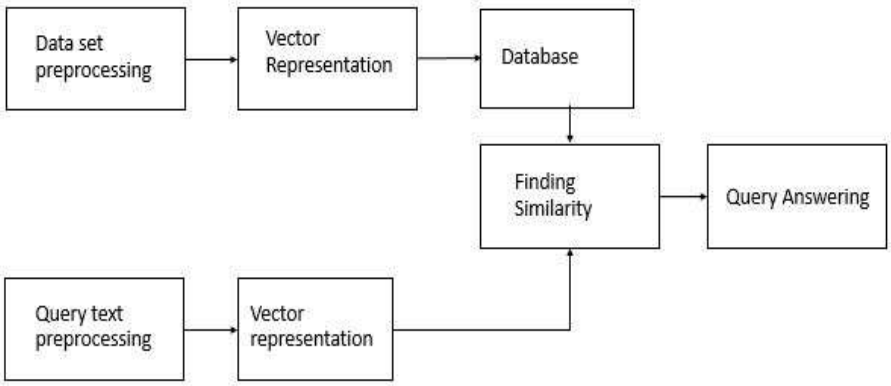

Figure 1. Block diagram

The proposed system is based on the following ideas:- The input query given by the farmer is preprocessed by removing stop words and irrelevant words. Stop words which are present in the sentence will be eliminated to give precedence to important words. Proper sanitization is applied to the sentence to remove punctuations and stemming is performed. TF-IDF is used to construct vector and cosine similarity is used to select the feature vectors. Figure 1 shows the block diagram of our system.

- **Corpora generation**: Tokenization is the main step in feature vector construction which will help separate out the words. Such words are then known as tokens. Skip gram is used for tokenization. A genism dictionary is constructed and this data is saved as dictionary text. Dictionary text is then converted to vectored for using bag-of words format and saved in MM (market matrix) format.
- **Finding Similarity Matrix:** TF-IDF method is used for converting data to its vector format.TF-IDF works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Each document is converted to its vector form and similarity matrix is created.

The formal procedure for implementing TF-IDF has some minor differences over all its applications, but the overall approach works as follows:-

Given a document collection D, a word w, and an individual document d ϵ D, we calculate w_d= f(w, d) * log (|D|/f(w, D)) (2), where f(w, d) equals the number of times w appears in d, |D| is the size of the corpus, and f(w, D) equals the number of documents in which w appears in D. There are a few different situation that can occur here for each word, depending on the values of f_w, d, |D|, and f_w, D, the most prominent of which is examined.

- **Query generation:** Similarity matrix is loaded from database. Query is fed in to the model after converting in to vector space using the dictionary stored. Cosine similarity technique is used to find the similarity of the extracted feature vector with the available data set and the most appropriate remedy to the problem will be chosen in such a way that the answers are retrieved from the data set with queries having higher cosine similarity value with the feature vector.

V. Performance Analysis

The system is tested on kissan call centre data. The returned documents were arranged in descending order of weights, with documents with higher weight sums appearing first. To compare obtained results, Naïve method for retrieving query is performed parallely. Naturally, this naive method is intuitively flawed for the larger problem of query retrieval, since this approach would simply return documents where non relevant words appear most (i.e. long documents with plenty of articles and prepositions that might not have any relevance to the query,also the stop words). Thus it provides an evidence that TF-IDF, though relatively simple, is a big improvement over this naive approach.

A lot of work still remains to be done in the field so as to achieve full reasoning capability but there is also a need to device a model that can serve as multilingual question answering system so not to give limit within a specific language but to handle the queries in all the languages. Fig 2 shows the result of running naïve query retrieval on our data. Notice an algorithm does not consider $w_{d,}$ but rather returns documents based solely on $f_{w,d}$. Relevant documents are scattered sporadically, so simply returning the top documents as done by this algorithm returns irrelevant documents.

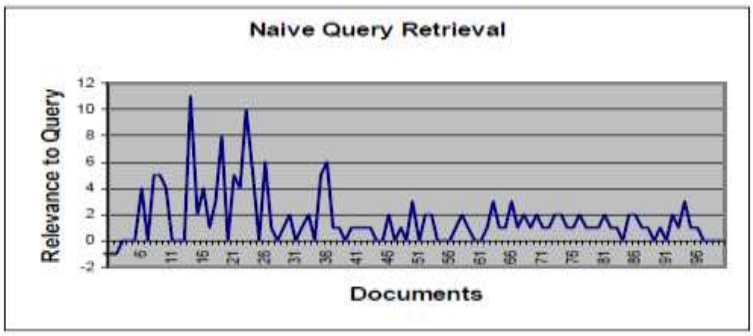

Figure 2. Naïve query retrieval

Figure 3 shows the result of retrieval with TF-IDF on our data. High values of $w_{d,}$ are concentrated on the beginning of the graph, so basing query retrieval on the top words here will likely return relevant documents. The two extra graphs indicate upper and lower bounds found by the retrieval engine.

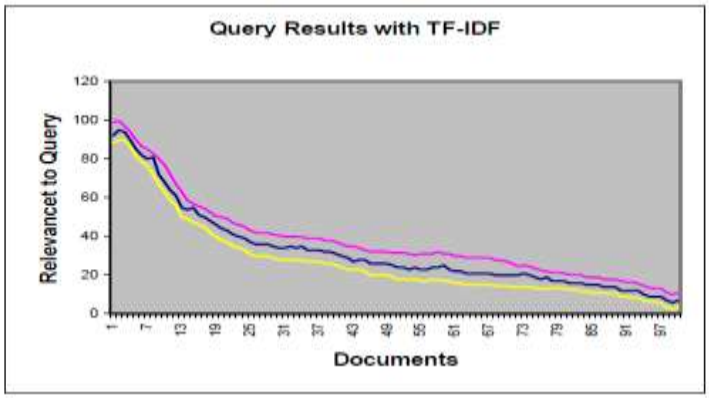

Figure 3. Query results with TF-IDF

VI. Conclusion

TF-IDF is proved as an efficient and simple algorithm for matching words in a query to documents that are relevant to that query. From the data collected, we see that TF-IDF returns documents that are highly relevant to a particular query. If an user tries to input a query for a particular topic, TF-IDF can be best used to find documents that contain relevant information on the query. Furthermore, encoding TF-IDF is straightforward, making it ideal for forming the basis for more complicated algorithms and query retrieval systems.

Despite its strength, TF-IDF has its limitations. In terms of synonyms, notice that TF-IDF does not concern about the context of the sentence. TF-IDF cannot be used to find the semantic meaning of the text. If the user wanted to find information about, say, the word 'Obama' , TF-IDF would not consider documents that might be relevant to the query but instead use the word 'USA' their by slightly decreasing the word's $w_d$value. In our own experiment, TF-IDF could equate the word 'drug' with its plural 'drugs' considering only the stemmed words, categorizing each instead as separate words. For large document collections, this could present an escalating problem.

References

[1] Tu Shouzhong, Huang Minlie, "Mining microblog user interests based on Text Rank with TF-IDF factor*,"* The Journal of China Universities of Posts and Telecommunications October2016, 23(5):40–46

[2] *https://radimrehurek.com/gensim/apiref.html*

[3] Tait Larson, Johnson(Heng) Gong, Josh Daniel, "Providing a Simple Question Answering System By Mapping Questions to Questions." Technical report, Department of Computer Science,Stanford University, 2006

[4] Leon Derczynski, Alan Ritter, Sam Clark, Kalina Bontcheva,(2013), "Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data," Proceedings of Recent Advances in Natural Language Processing, pages 198–206

[5] Sanglap Sarkar, Venkateshwar Rao, Baala Mithra SM, Subrahmanya VRK Rao(2015), "NLP Algorithm Based Question and Answering System*,"* Proceedings of 2015 Seventh International Conference on Computational Intelligence, Modelling and SimulationPages 97-101

[6] Berger, A & Lafferty, J. (1999)," Information Retrieval as Statistical Translation," In the proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval. (SIGIR' 99), 222-229.

[7] Berger, A et al (2000), " Bridging the Lexical Chasm:Statistical Approaches to Answer Finding," In Proc. Int.Conf. Research and Development in Information Retrieval, 192-199.

[8] Berry, Michael W. et al. (1995)," Using Linear Algebra for Intelligent Information Retrieval,"*SIAM Review*, 37(4):177-196.