DEVELOPING UNDERSTANDABLE & EFFICIENT RETRIVAL SYSTEM FOR MEDICAL QUERIES

ABSTRACT

Evidence based medicine is widely used for making decision and suggestion for practitioners. There is lot of medical data available on web, but it is difficult to understand for everyone or making decision from them. Evidence based medicine is used by practitioners to identify disease and cause for medical condition and suggest treatment. This paper present efficient way to retrieve data for medical query. The main difficulty is to complex relationship in medical record and need domain expert, difficult biological terminology anyone cannot get or understand info and efficiently process their query. In this paper i am presenting to process query through vector model and retrieve disease and possible treatment for them.

INTRODUCTION

There is lot of clinical and medical data available on internet medical data extraction tool have extracting data which is very difficult to understandable that data is not in structured form. It is not every one can understand and relate relevance of data according to their query. This research paper present that find meaningful and understandable result from large amount of medical data. The main difficulty is to complex relationship in medical record and need domain expert, difficult biological terminology anyone cannot get or understand info and efficiently process their query.

Some ambiguous queries (1) A query expresses a clearly defined sense, but the genuine needs under this sense may cover a broad range. Taking a common scenario where an ordinary user performs medical search for example, he feels excruciating (he has a leg pain when walking and rash erupts on his body) but is dubious about his exact medical problems, so he inputs ‘‘leg pain’’ and ‘‘rash’’ as keywords into a search engine. In this case, as many diseases may cause these symptoms, the user may prefer to learn knowledge about all these diseases and possible treatment, so as to have a preliminary understanding about his situation and better prepare for the interview with doctors. (2) Query terms themselves are ambiguous, as most users have little medical knowledge. For instance, a pregnant woman feels pain in her stomach, so she submits a query composed of ‘‘pain in the stomach’’ and ‘‘pregnant’’. In this case, the term ‘‘pain’’ is ambiguous, which may mean ‘‘stabbing pain’’, ‘‘distending pain’’, ‘‘labor pain’’, etc. The user-cared reasons causing these different kinds of pain, however, may be totally different.

RELATED WORK

Disease classification system the authors classify feature extraction based on disease train system for 6 different diseases and machine learning algorithm to classify. They use key-phrase extraction technique to identify useful information

The proposed model encompasses six major processes such as pre-processing, Diversification strategy, Meta Map, ontology, Vector Space Ranking Model and Neural Network based Classifier.

Pre-processing strategy is used to analyze the stop word, synonym, and white space present in the user query and the appropriate keyword is extracted from the input medical query. The diversification strategy involves four steps: Query understanding, query transformation, candidate concept mapping and derived query generation.

Each group of words consists of multiple concepts and words. The documents are ranked based on its importance. The importance of Neural Network based Classifiers used to train and test the medical documents and queries to predict the output based on the similarity between the document vectors and query vectors.

METHODOLOGY

The proposed methodology consist on four steps. First step is to analyzing of dataset and then second step is to extracting symptoms, diseases and prescriptions. In third step is to preprocessing, in fourth step is to calculating TF-IDF, In fourth step is to ranking using matching score, shown in Fig.1

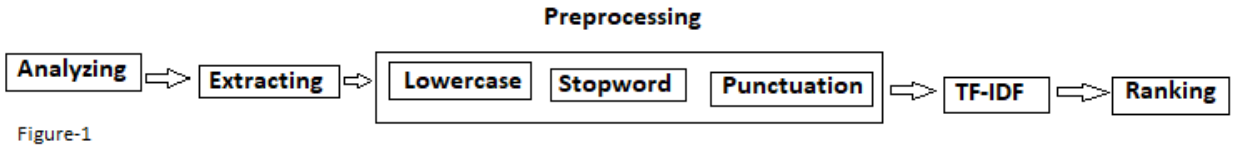

Figure-1

Step-1: Analyzing

The first step is to analyze the data, if we look at the dataset at first glance, we see all the documents with words in English. Each document has three major text which is separated by comma first thing is list of symptoms and second thing is disease and third is treatment.

Step-2: Extracting

This is major step to crawling data in that way this will help us to further process we will tokenize all data into words and separate symptoms , diseases and treatment that correlate with treatment, then we will create data dictionary and fill with all tokens.

Step-3: Preprocessing

Preprocessing is one of the major steps when we are dealing with any kind of text models. During this stage we have to look at the distribution of our data, what techniques are needed and how deep we should clean.

This step never has a one hot rule, and totally depends on the problem statement. Few mandatory preprocessing are converting to lowercase, removing punctuation, removing stop words. In our problem statement it seems like the basic preprocessing steps will be sufficient.

Lowercase

During the text processing each sentence is split to words and each word is considered as a token after preprocessing. Programming languages consider textual data as sensitive, which means that The is different from the. we humans know that those both belong to same token but due to the character encoding those are considered as different tokens. Converting to lowercase is a very mandatory preprocessing step

Stopword

Stop words are the most commonly occurring words which don't give any additional value to the document vector. in-fact removing these will increase computation and space efficiency. I have manually download all stop words list and then remove from data dictionary.

Punctuation

Punctuation are the unnecessary symbols that are in our corpus documents, we should be little careful with what we are doing with this. There might be few problems such as U.S—us "United Stated" being converted to "us" after the preprocessing. hyphen and should usually be dealt little carefully. but for this problem statement we are just going to remove these.

We are going to store all our symbols in a variable and iterate that variable removing that particular symbol in the whole dataset.

Preprocessing

Finally, we are going to put in all those preprocessing methods above in another method and we will call that preprocess method.

Step-4: Calculating TF-IDF

By taking a multiplicative value of TF and IDF, we get the TF-IDF score, there are many different variations of TF-IDF but for now let us concentrate on the this basic version.

tf-idf(t, d) = tf(t, d) * log(N/(df + 1))

use dictionary as with (document, token) pair as key and any TF-IDF score as the value. We just need to iterate over all the documents, we can use the Coutner which can give us the frequency of the tokens, calculate tf and idf and finally store as a (doc, token) pair in tf_idf.

```
tf_idf = {}
for i in range(N):
    tokens = processed_text[i]
    counter = Counter(tokens + processed_title[i])
    for token in np.unique(tokens):
        tf = counter[token]/words_count
        df = doc_freq(token)
        idf = np.log(N/(df+1))
        tf_idf[doc, token] = tf*idf
```

Step-5: Ranking using Matching Score

Matching score is the most simplest way to calculate the similarity, in this method, we add tf_idf values of the tokens that are in query for every document. for example, if the query "hello world", we need to check in every document if these words exists and if the word exists, then the tf_idf value is added to the matching score of that particular doc_id. in the end we will sort and take the top k documents.

```
def matching_score(query):
    query_weights = {}
    for key in tf_idf:
        if key[1] in tokens:
            query_weights[key[0]] += tf_idf[key]
```

key[0] is the documentid, key[1] is the token.

CONCLUSION

In the medical field mostly the problems need their solutions for the betterment of the society at a broader level. The natural language processing can help many things related to the text. The patient descriptions in our local context are written in the form of textual format. By using vector space model calculating tf-idf weights we got good result in little dataset.

REFERENCES

[1] "Developing Disease Classification System based on Keyword Extraction and Supervised Learning"

[2] "https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089"