Spoken Query Processing-A Report

Kunal Gupta and Jeevan Joishi, MT13040,MT13036.
Indraprastha Institute of Information and Technology

---

1. INTRODUCTION

Traditional text based search engines has always been flowing in one direction where a user fires a query on some kind of graphical user interface provided and obtain results for the same. But with time, challenges arose in understanding the working of these search engines had the query been fired as a natural language monologue rather than precise search terms.

This project tries to study and deal with such natural context search. The queries can be long, verbose and may include many expressive terms not commonly found in general dictionary. As such, reducing the given query to a more appropriate query that does intend the same meaning is of utmost importance. Also, there is a need of a speech-to-text application that shall help in fetching the query and then further processing be applied on the fetched query.

2. RESOURCES

2.1 Search Engine

The project uses open search engine Galago 1.04. Galago takes 10 minutes on average to parse a given corpus. So building index and then performing search on it can be can be done quite efficiently.

2.2 Dataset

The project focuses on spoken query processing. So much of the work was focused on getting the most appropriate query from the input received from the speech app. Dataset chosen for the experiments were wiki-small and wiki-large as obtained from [1]. Besides, more static HTML pages dumps collected from [2] were also used in the experiments that were all converted to corpus format beforehand.

2.3 Speech-to-Text App

Speech is an indispensable part of the project. An android app was developed which uses the Google voice search API to convert speech to text. Also a server was developed that continues to listen to this speech-to-text app.The received text is then forwarded to the search engine where further processing takes place.

3. DESIGN CONCEPTS

3.1 Stopword Removal

A frequently occurring word but of no relevance like if, to, be etc. can have a huge impact on the document score if not taken proper care. Also for weighting schemes that gives due importance to term weights, not-removing such words may have a negative impact on the results obtained. So removal of these stopwords from the query is important. For a query, galago first creates a query tree and then

transforms the query tree. The transformed query tree will in turn fetch the results pertaining to the query.Removing stopwords just before the query tree creation starts will expedite the process as limited terms are then processed in subsequent stages.

3.2 Noun Phrase Detection

With stopwords gone, a query may still contain terms that are not so contributing factor to the search process and whose removal doesn't change the search results drastically. So detecting important words be it noun or noun phrases, foreign words, etc can be quite useful to further shorten the query. The project uses POS tagger from Stanford NLP group that helps in identifying nouns,verbs, adjectives,etc. The output from these taggers give an idea of the key concepts present in the query.

3.3 Key Concept and Quality Predictors

Once a set of key concept is obtained, the task is finding the most suitable query/sub-query is also important. The selected sub-query would be the one that will have more impact on the search process as compared to other sub-queries. Various query quality predictors have been discussed in [3]. The project implements two query quality predictors viz. IDF-based features and Query Scope. Besides, Simplified Clarity Score (SCS) was also studied and experimented during the process, but differences in the output were least significant.

IDF-based Features: The IDF of each query term w was calculated as

$$IDF_w = \frac{log_2 \frac{N+0.5}{N_w}}{log_2(N+1)}$$

where N_w is the document frequency of the the term w and N is the total number of documents in the collection.

Query Scope(QS): It is a measure of the size of the retrieved document set relative to the size of the collection. It is given by

$$QS = -log\frac{n_Q}{N}$$

where n_Q is the number of documents containing atleast one query term. A higher value of QS infers the query being of low quality.

Simplified Clarity Score(SCS): It is a pre-retrieval performance indicator and is given by

$$SCS = \sum_{w \in Q} P_{ml}(w|Q) \times log_2 \frac{P_{ml}(w|Q)}{P_C(w)}$$

where $P_{ml}(w|\mathrm{Q})$ is the probability of the occurrence of the word w in the query, $P_C(w)$ is the probability of occurrence of w in the collection.

3.4 Ranking

Galago has Dirichlet model implemented that plays major role in ranking the documents obtained for a given query. Dirichlet is more of a DAAT technique. So the project also implements Vector-Space Model which is more inclined towards TAAT technique. The user can choose between the two ranking techniques and also between the quality predictors technique to study the variation in output in different cases.

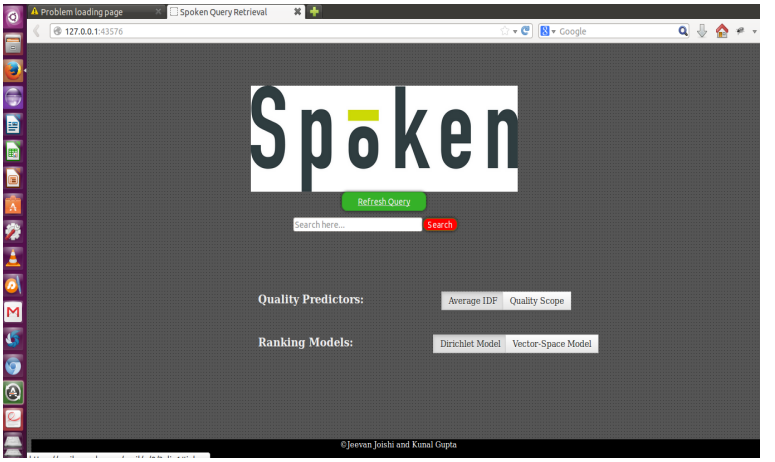
4. IMPLEMENTATION AND WORKING

Android Application: An android app was devevloped namely *SpeechRE* was developed to collect the spoken query from a mobile interface. It uses the google speech API to best identify the spoken terms. Once identified, the app automatically writes the words in a text file residing in the Android device itself. The app is provided with a *Send* feature that send the written data to the system.

Also a server was developed that listens to this app using Socket Programming. At the present instance, the app and the ports are pre-configured to run on our system only which is clearly a limitation. Upon receiving data from the app, the server writes the data again to a text file in the system. Queries for further search will be taken as input from this file.

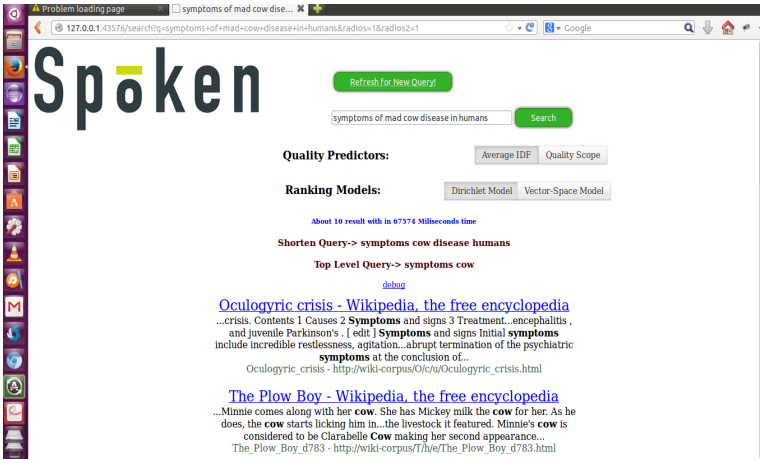
Web User Interface: Existing Galago user interface was changed to a great deal to make the project much more easier to use. Plus changes were incorporated in the user interface to accommodate not just conventional text-field type search box, but rather include options to read data that had been sent from the Android app. Also options to change search parameters in the form of Quality Predictors and Ranking Models were also incorporated in the user interface and subsequently these parameters forced different changes to the existing classes of Galago search engine.

The various parameters for speech search only are now accepted through command line and for two different spoken queries with different parameters, the search process has to be re-initiated. Changes to include such features in the web user interface is yet to be implemented. The following snapshot should give you a feel of the user interface.

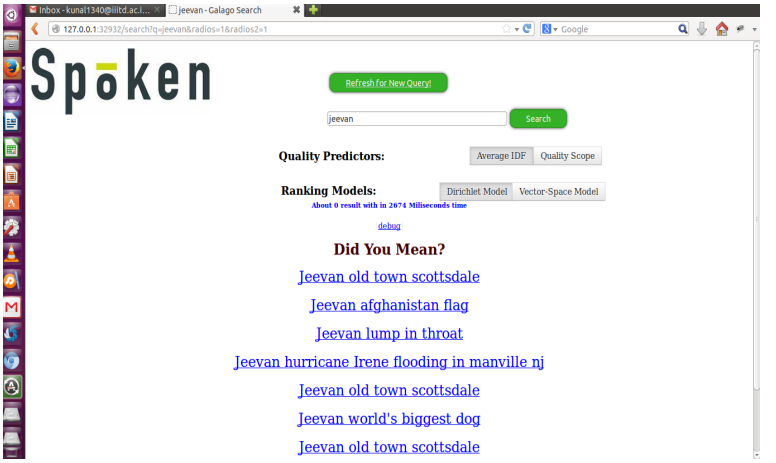


0:4 •

Working: Based on the parameters selected at the command line(for speech search) or user interface(for traditional text search), the query first goes through noun phrase detection to identify the key concepts(nouns and noun-phrases in this case). Thereupon the set of key concepts goes through the quality prediction techniques to identify the best possible query or sub-query. The best sub query then goes through one of the two methods viz. Dirichlet or Vector-Space to determine possible document(s) output and their respective scores.



Also if no results are found for the query, a small set of possible search terms are then presented to the user. But if a user intentionally or accidentally fires an empty query and initiates the search process, the browsing is redirected to the home page.



5. CHANGES TO EXISTING GALAGO

Noun Detection: Used it for detecting noun and noun phrases from the input query. Implemented in the java file Search.

Sub Query: Implemented it for figuring out the best possible sub-query from the various combina-

tions of keywords detected in noun phrase detection step. Implemented a class Combination that pipes input the output of noun detection phase, and give various possible sub-queries keeping their relative ordering same. These sub-queries are then passed through a function runSubQuery() in the Search class that finds the best sub-query out of all possible sub-queries returned from Combination class.

Vector-Based Model: Incorporated many changes to SearchWebHandler class. Also option to include quality predictors and choose between Dirichlet Scoring and Vector Based Model is incorporated in this class.

Batch Search: Changes were made in the App class and BatchSearch class to obtain output with the score and rank of the documents. For each query, the number of documents displayed are ten in number (if available).

6. EVALUATION

The project was evaluated using batch-search for a set of queries. The process was repeated for the two different ranking models. For each query, the ten results were looked into(if available) for their scores and their ranking according to that model.

The results were evaluated using Normalized Discounted Cumulated Gain(*nDCG*) against an ideal vector chosen as <**3,3,3,2,2,2,1,0,0,0**> and the normalized ideal vector being <**1,1,1,1,1,1,1,1,1,1**>. For quality prediction Average IDF features was used throughout.Measures starting from Cumulated Gain(*CG*) to *nDCG* were studied and a set of results for a query is displayed below.

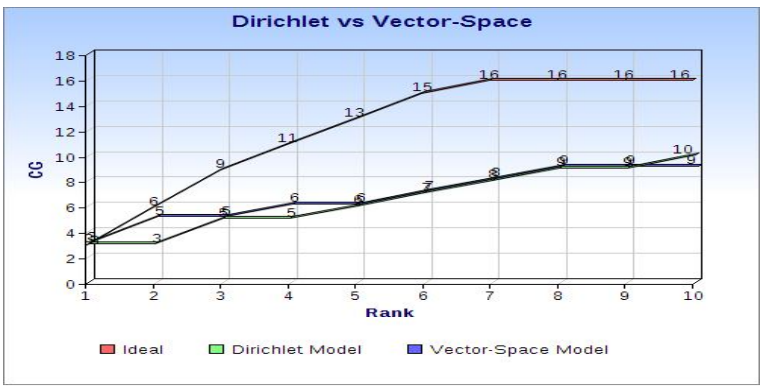


Figure 1: Comparison between CG values for Dirichlet Model and Vector-Space Model.

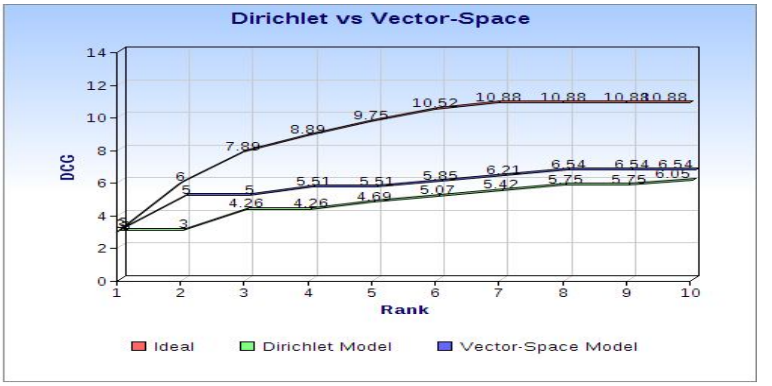


Figure 2: Comparison between DCG values for Dirichlet Model and Vector-Space Model.

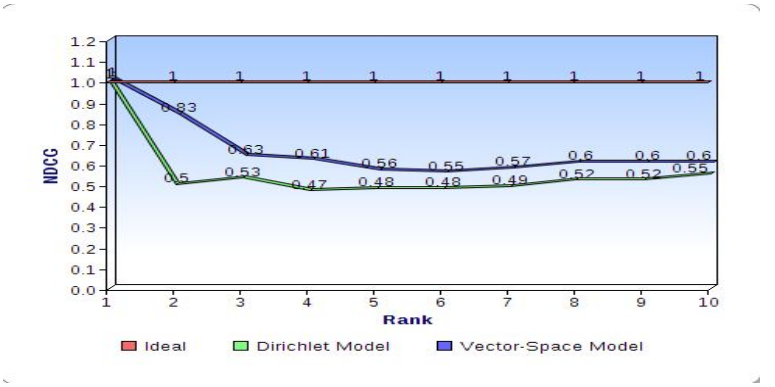


Figure 3: Comparison between nDCG values for Dirichlet Model and Vector-Space Model.

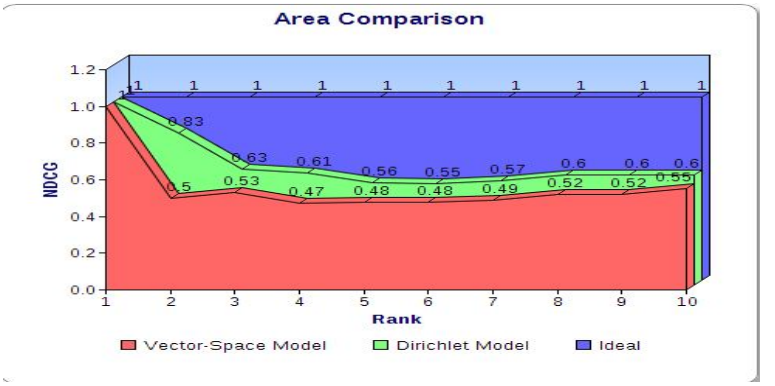


Figure 4: Area Comparison between nDCG values for Dirichlet Model and Vector-Space Model.

7. RESULTS

The figures in the previous section tells a lot about the performance and quality of the IR techniques. *nDCG* area comparison says that the area between ideal curve and *nDCG* curve represents the quality of the IR technique. For the query evaluated an shown in the figure, Vector-Space Model(*area in blue+green*) performs better than Dirichlet Model(*area in blue*). But the same cannot be said for every

queries.

Also judging a document being relevant or non-relevant was entirely the decision of the group members involved in the project. And as such, the inference from this system/report may be not be taken as a hard and fast rule for any other experiments or evaluations. This project aimed at studying various ways to handle verbose spoken queries and thereupon using IR techniques to evaluate results based on various parameters, which we believe has been successfully taken into account.

8. REFERENCES

[1]www.search-engines-book.com/collections
[2]http://dumps.wikimedia.orgother/static_html_dumps/current/
[3] Reducing Long Queries Using Query Quality Predictor-Giridhar Kumaran and Vitor R.Carvalho.