Information retrieval – Final Project *

Extended Abstract†

Yoav Zioni
Hiafa Univ.

Israel
zioni.yoav@gmail.com

Omer Shimoni
Hifa Univ.

Israel
omer.shimoni64@gmail.com

ABSTRACT

The focus of this project is the “Non-factoid Question-Answering” task. Within such a setting, a user may post questions in natural language and answers should be informative enough to satisfy the user’s information need.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics • **Networks** → Network reliability

KEYWORDS

IR, Non-factoid Question-Answering

ACM Reference format:

G. Gubbiotti, P. Malagò, S. Fin, S. Tacchi, L. Giovannini, D. Bisero, M. Madami, and G. Carlotti. 1997. SIG Proceedings Paper in word Format. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK’97)*, 4 pages. https://doi.org/10.1145/123 4

1 INTRODUCTION

The "nfL6: Yahoo Non-Factoid Question Dataset" is derived from Yahoo's Webscope L6 collection using machine learning techniques such that the questions would contain non-factoid answers. The dataset contains 87,361 questions and their corresponding answers. Each question contains its best answer along with additional other answers submitted by users. Only the best answer was reviewed in determining the quality of the question-answer pair.
The data fields correspond to dictionaries containing the information. These dictionaries comprise the following:

Question: The string representing the question. Answer: A string representing the best answer to the question. Nbestanswers: One or more strings representing other submitted answers to the question. main_category: A string representing the Yahoo category for the submitted question. Id: The unique Yahoo ID string the question.

We used the Lucene library so when given a query we return the 5 "best answers" to the query. The evaluation was done by giving a set queries which are a part of the data set so we know what the "best answers" are and can check if the answers returned are the "best answers" or a part of them.

2 Data Processing And Analysis

2.1 Basic Processing

We used the English analyzer and add to the stop word list a few signs like "?" and "," etc... We didn’t do much for this part of the data processing.

2.2 Scoring Terms

We give a score to each word in the query for every category; we did this by putting all the answers of a category in a file for every category. We searched (basic search from the Lucene library) the word in every category file and the score for every category is the score that this word got from the search in that category file.

2.3 Classifying Queries

According to the score of each word in the query we classified the query to one of the categories. The category with the highest score (considering all the word in the query) was the one that will be the query's category. The main search for the answers for the query was done on all the answers from the dataset and not for just a group of them..

3 The Retrieval Approach That We Implemented

3.1 25BM Similarity

We used the 25BMSimilarity to help find more relevant answers to the queries. This type of retrieval function was not enough for us to get a good result. While 25BM represent state-of-the-art TF-IDF-like retrieval functions used in document retrieval for our case this was just the base of it.

3.2 Boosting Terms According To Query Category

Because the 25BMSimilarity wasn’t enough for us to get anywhere close to a good result we used boosting. During our data processing we classified the query to a category, in addition we gave a score for each query term for all categories. What we did was query time boosting where we boosted each query term according to its score in the query's category. By doing this terms that are important in the query's category (have high score in that category) are considered to be more important in the search than terms that are not so important (have low score in that category). For example if the query's category is related to computers (according to our classification) and in the query there are the words "computer" and "juice". The word "computer" will be way more important in the search than the word "juice" because the score of the word "computer" in this category is high and the score of the word "juice" is low. If we had classified the query to be in the category "food" then this would have been the other way around (meaning that the word "juice" would have been important and the word "computer" would have been not really important or with real weight to it).