Cluster Rank Demo Harness

Philip Robinson

Oregon Health Sciences University

June 25, 2017

Abstract

It is often the case that initial query compositions result in frequent restarts as the user negotiates with their retrieval system. This is likely a product of unfortunate query formulations or choice of ranking algorithm. Our proposed retrieval system encourages diversity in displayed documents by introducing an unsupervised clustering step before displaying results. The clusters are then presented to the user with their documents ranked independent of each group. We do this by clearly seperating the retrieval process into the three steps `relevance`, `clustering`, and `ranking`, then allow the user to recurse this process on a cluster (rather than restarting their query). Additionally, we propose a simple method to compare results against varying quality `tfidf` queries. Our final product is a demo harness that abstracts these steps, so that others may easily produce and reproduce prototypes against their own corpora.

1 Introduction

Information retrieval systems have long suffered from non-informative query formulations by their users. Many systems employ techniques such as query expansion, domain ontologies, advanced search parameters, to address such difficulties. Unfortunately, most retrieval systems presume user provided queries to be informative, and select to return the most query-relevent documents. We can interpret a query-relevance ranking on documents retrieved by a non-informative query as being an overfit ranking (to the query). Unfortunately, this overfitting can happen regardless of query quality. In cases where all documents retrieved are nearly identical.

Either in the case of non-informated queries or of monolithic document listings, users are often tasked with query reformulation. Reformulation may be an inssuficent mechanism, for users not appropriately familure with their target domain. Additionally, as a side effect of this workflow, query reformulation can make it difficult to assess the general effectiveness of a retrieval system[1].

To prevent overfit rankings some retrieval systems have proposed document diversification strategies[2]. This can be accomplished by introducing similarity or taxomonic clustering of documents either prior to or after query submission. Additionally, to improve query formulation many search engines provide similar search terms along side retrieved documents[3]. These aid terms are usually assigned by experts to similar vocabulary or to retrieved documents, and can be expensive to curate.

In our system, we propose seperating the query processing pipeline into quaratneened steps. We first identify documents by relevence, then perform unsupervised clustering of relevent documents, finally rank each cluster by the query provided. We then allow the user to zoom in on a cluster. During this process, every cluster is also tagged with terms automatically, by providing the grouping's most impactful `tfidf` words. These terms don't impact the ranking, but act to provide the user with additionall information in selecting a term.

We hypothesis that a document clustering will allow users to eliminate poor document groupings. Additionally automatic tagging of clusters with relevance terms should allow users to navigate a listing retrieved from non-informative queries. To form queries, we sample words from randomly selected documents' `tfidf` distributions and capture the resulting ranking $R_{\texttt{tfidf}}$. For a system with C-way clustering to be considered performant, the same query should yield our target document within S steps.

$$S < log_C(R_{\texttt{tfidf}})$$

As this is a process heavily dependent on each component, we also seperate out these concerns to hopefully decrease testing time against multiple corpora, ranking, and clustering configurations.

2 Implementation Details

We provide a retrieval systems harness, developed in `python`[4].

3 Evaluation Approach

4 Expirimental Results

5 Limitations

6 References

[1] reference

[2] reference

[3] reference

[4] `https://github.com/probinso/IR-cluster-rank-demo`