Diversification of search results using entity linking [*]

[Project Report]

Aravind Sankar
CS11B033
aravindsankar28@gmail.com

Adit Krishnan
CS11B063
adit101293@gmail.com

ABSTRACT

Our work addresses a very well known problem in Information Retrieval, which is the problem of Query Result Diversification. We focus specifically on Web Search queries which are small text fragments. To enrich the set of documents returned for the query , we use the idea of Entity Linking, where different entity mentions appearing in a set of documents are linked with their corresponding entities present in a Knowledge Base Resource. This also presents a user with a wider perspective to choose from.

The motivation behind an entity based approach, is to inherently accomodate the ambiguity associated with queries, by covering a wider set of entities that could be linked to the query text. Each entity represents a different direction that could be explored, hence a greater number of distinct entities covered would mean a more diverse set of results. To the best of our knowledge, no other previous work has been done that uses entity coverage to incorporate diversity. We have used the AOL query logs available [1] which contains anonymized click data, for our experiments.

General Terms

[Information Storage and Retrieval]: Information Search and Retrieval

Keywords

Entity linking, Diversity, Ranking

1. INTRODUCTION

The main motivation for our problem comes because of two reasons, firstly natural language is ambiguous and secondly user's preferences can't be adequately captured from short web search queries. Consider a case where the user searches for Kentucky. If a system takes only relevance and popularity of web pages into consideration, it would very likely return most of the search results about the Restaurant Chain, which will misguide users who are looking for the city for instance. In such a case, the user has to manually try to make his query more specific which may be hard to do in absence of knowledge. This shows the need for diversification of top few search query results.

[*]As part of project, CS6720 - Data Mining, Jan - May 2015
[1]AOL Query logs - http://jeffhuang.com/search_query_logs.html

We are proposing a 3-phase solution here.

Phase-1: Devise a method to obtain the set of entities which the query could reference.

Phase-2: Extend the entities from the first phase to cover a greater number of related entities. This is done by exploiting the structure of the underlying entity linkage graph of our Knowledge Base.

Phase-3: Apply a conventional retrieval technique to obtain a large number of documents using the initial query text and then re-rank these to cover the set obtained from the second step as well as possible, thus posing a set cover problem.

2. RELATED WORK

In the past, there has been a vast body of work on linking entity mentions in a document. Information extraction systems typically face the problem of Entity Disambiguation. For an example of this problem, consider an entity mention Michael Jordan. This could refer to either the basketball player, or the professor (at UCB) or in rare cases some other person. This problem has been widely solved as done in [2],[3],[4]. In addition to linking entities in documents, recent works have focused on entity linking in web search queries, where knowledge bases (such as Wikipedia, DBpedia, etc.) are used to enrich the query entities such as in [5].

One aspect of our solution is to do the same for web search queries, which is quite challenging owing to their sparse content. This is where anchor texts come in handy. Anchor texts essentially include all text appearing on incoming edges to an entity. They essentially include the roles played by this entity in its appearences.

Another orthogonal direction of research that has been pursued in the Information Retrieval literature, is the diversification of search query results. The basic idea behind diversity is that the top results presented should be semantically different from each other (in addition to being relevant) in order to have a maximum coverage and minimum redundancy. One of the commonly used approaches to this problem is using the notion of maximal marginal relevance (MMR) [6]. Here, documents are first retrieved for a given query based on relevance. Then, documents are iteratively selected from this list that are most relevant to the query, while being dissimilar to the ones already selected.

A second approach to address diversity is by using click stream data as in [7]. Here, the core idea is that if clusters of users click on different URLs (to different pages) after giving a specific query, then we know that diversity is needed for this query. Another (not so relevant here) approach to diversity, which is used more in the context of information networks is based on the idea of Markov random walks, as done by [8].

As opposed to the above ideas, We are looking at diversity not in terms of semantic measures (such as wordnet based) or markov chains, But instead in terms of the Maximal Marginal Relevance of a new document with regard to the new entities being introduced by it, which do not have mentions in the existing set of results.

3. DATASET

Our knowledge base of choice is Wikipedia due to it's accessibility, popularity and extensiveness.

About 27 million unique entities are covered in the Wikipedia annotations of it's documents. The corresponding linkage graph among these entities is also available based on the occurence of entities in the documents corresponding to the source entities. A secondary piece of text arising from the redirections that occur in Wikipedia among related documents, called the anchor text is also available for the edges (when moving from source to destination). We have used these for phase 1 as later explained. In all, our entity graph consists of 27 million entities and 250 million edges (links between entities).

For our document corpus, we used the latest Wikimedia dump avaiable at [1] which contains the current version of all Wikipedia pages and articles. We're using the Wikipedia corpus because it gives us pre-annotated text (with entities), which saves us the task of annotating entities. The size of this corpus is over 30 GB of compressed text documents in xml format. A small excerpt from the our document corpus is shown below :

[[Anarchist schools of thought]] can differ fundamentally, supporting anything from extreme [[individualism]] to complete collectivism. Strains of anarchism have often been divided into the categories of [[social anarchism|social]] and [[individualist anarchism]] or similar dual classifications by [[Geoffrey Ostergaard|Ostergaard, Geoffrey]]

(The text in the square brackets denote annotations. Where there are 2 pieces of text, the first is the entity as named in wiki and the second is the raw text appearance linked to the first part)

For the query dataset, we used a small sample of AOL web search queries from the AOL query logs.

4. METHODOLOGY

4.1 Phase-1: Obtain an initial set of entities S from query text

A simple solution that was considered for this phase was to build an Apache Lucene Index on top of the Aggregated Anchor Texts corresponding to every entity. Note that the Aggregated Anchor Text of entity **E**, let us call it A_E, would include Anchor Texts appearing on every incoming edge to **E** in the entity graph of the Knowledge Base. Then using a Lucene Baseline Search with the query text on the above set of Agg. Anchor Texts, Identify the closest entries. We could either retrieve a contant number of entities here or use a relevance score threshold for choosing the starting set.

A second solution, which we have used here, is to use a simpler hashing based technique rather than indexing using Lucene. The idea is to build a reverse index from Anchor Text Phrase to entities, where an Anchor Text Phrase is the Anchor Text on a single edge between 2 entities.

Suppose an anchor text phrase P appears in some incoming edge to entities E_1 and E_2, then E_1 and E_2 are both hashed to the anchor text phrase P. Based on our experimentation, the number of unqiue anchor text phrases is around 24 million. Furthermore, the words present in the anchor text phrases were assigned an IDF score based on the number of phrases they appeared in. Below is a simple illustration.

Anchor Text Phrase	Word IDFS	Entity List
Holy Christ	x, y	E_i, E_j

Holy Christ was an Anchor Text phrase that appeared for some church related entities. E_i and E_j are illustrative of two such entities.

If the word Holy appeared in $count_{Holy}$ Anchor text phrases.

IDF x = $\text{Log}(1/(count_{Holy}))$.

The above hashmap is built over all the 24 million Anchor Phrases.

A little additional exploration reveals that the total number of unique words that appear in these 24 million Anchor Phrases is only a little over 9 million. So we also built an additional index on top of the above hashmap which indexes the Anchor Phrases based on whether the specific word appears in it or not.

Now when a query is processed, it is split into the constituent words. For each constituent word, we find all the corresponding Anchor Phrases using the above index, and hence the associated entities. The score then provided to each of these entities is the IDF of this word, which makes sense since a more discriminative word has a higher IDF and the corresponding entities are more important.

Scores are then added for all the entities appearing in any of the Anchor Phrases for any of the constituent words, and the best N entities are chosen as the starting set of entities.

4.2 Phase-2: Obtain the induced Subgraph using the starting set

The starting or seed set of entities S is obtained from the procedure described in Phase-1. For finding related entities we use the simple idea of Directed spread on the graph. In this project, we have used a simple two hop directed neighbourhood as the spread of a node i.e Every entity reachable from the starting entity in two directed hops.

When this is performed over all the entities in the starting set and their corresponding edges are added, we obtain a random forest graph that is illustrated by a smaller sample in Figure 1.

This is the general trend that we observed in the subgraphs. There are a set of strongly clustered neighbourhoods corresponding to the different starting entities, which have very inter cluster edges. This is exactly what we expect. We have obtained a fairly diverse set of starting entities, corresponding to the clusters, and their related sets of entities.

Once this is done, our goal will now be to obtain a set of top K documents from our collection that can cover all of these entities as best as possible.

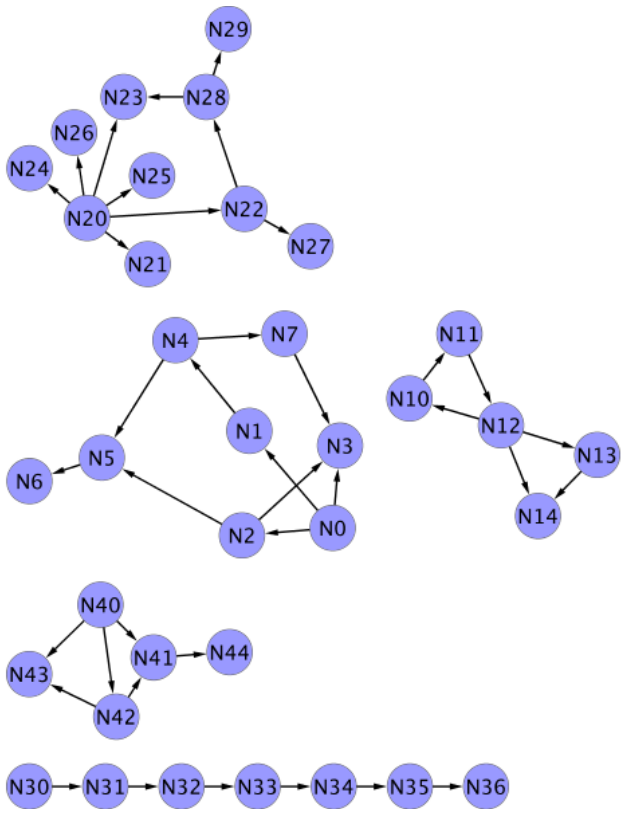

Figure 1: Starting Entity Set with spread

4.3 Phase-3: Rerank relevant documents to get the top-K most diverse set among them

4.3.1 *Lucene Document Index (Conventional Retrieval)*

The entire document corpus was indexed using Apache lucene [2] which is an existing scalable indexing and retrieval framework available in Java.

It uses a fairly involved retrieval mechanism based on the field used to index the document. We used 2 fields to index the documents in our corpus, entities present and the content of the document. For retrieving documents on giving a search query, we perform a content based search.

Lucene has its own measure of document relevance as shown in the next page, for a query q and document d. It depends on the relative frequency of the keywords of the query in the document, the length of the document, their cosine similarity, factors based on external boosting, tf-idf measures over the collection of documents etc.

[2]https://lucene.apache.org/

$$\text{score(q,d)} = \text{coord-factor(q,d)} \cdot \text{query-boost(q)} \cdot \frac{\text{V(q)} \cdot \text{V(d)}}{|\text{V(q)}|} \cdot \text{doc-len-norm(d)} \cdot \text{doc-boost(d)}$$

Lucene Conceptual Scoring Formula

Note that our diversity algorithm starts with a large set of relevant documents and then reranks them based on the diversity measure. For this purpose, we use the top 1000 documents returned here and rerank them to get the top K diverse documents. These are then evaluated for diversity against the top K of Lucene itself.

4.3.2 Greedy Set Cover Solution for Re-Ranking

We now have the top 1000 relevant documents from the Lucene retrieval for our query, ranked based on their content. We need to reduce this to the top K most diverse set of documents. We will pose this as a greedy set cover problem.

Let us say that document D_i has entity set e_i corresponding to it, i.e. the set of entities occuring in it. Let us call the vertex set of our induced subgraph in Phase 2 as $|V|$.

Then we want to choose the set of K documents, D_1,D_2...D_K which will try to maximize the fraction of $|V|$ covered. Since every entity in $|V|$ is considered equally important, for each v $\in$ V, score(v) = $1/|V|$.

Thus the objective is to maximize Σscore(v) where v $\in$ (D_1 U D_2 ... U D_K).

We then apply the greedy algorithm shown on the right.

The algorithm starts giving every entity in V an equal score. Then the topmost document is the one which covers the largest number of these. Once this has covered these entities, their scores get set to 0, since the remaining documents should aim to cover the entities that have not been covered by this document.

So the second document will be that which covers the largest number among the remaining entities, i.e. those not covered by the first document. Thus the i-th document will maximally the entities not covered by the first i-1 documents.

```
begin
    V ⟵ Set of reqd entities
    TopK_Docs ⟵ new Array()
    for $v \in V$ do
        score(v) ⟵ 1/|V|
    end
    score(v') ⟵ 0 , for all v' not in V

    for int $i \in$ 1 to k do
        best_doc ⟵ emptydoc()
        best_score ⟵ 0
        for doc $\in$ Lucene-top-1000 do
            score ⟵ 0
            for entity $e \in$ doc.entities() do
                score ⟵ score + score(e)
            end
            if score > best_score then
                best_score ⟵ score
                best_doc ⟵ doc
            end
        end
        TopK_Docs.append(best_doc)
        for entity $e \in$ best_doc.entities() do
            score(e) ⟵ 0
        end
    end
end
```

4.3.3 Scoring Entities using PageRank in Greedy Set Cover

We just introduce an additional idea here. Note that not all entities in vertex set V of the induced subgraph are equally important. A simple solution to weight them differentially is to run PageRank algorithm on the subgraph and assign the scores corresponding to the entities as the importance assigned by PageRank to their respective nodes.

The only difference from the previous algorithm is in the initial assignment of score(v) for v $\in$ V. Instead of $1/|V|$, we will need PageRankImportance(v).

5. RESULTS

Most diversity related papers in the IR domain use some external knowledge of the dataset to rank the diversity of their results for a specific set of queries. However we lack external knowledge here since we use a query log that contains unseen queries whose results cannot be predicted. Our only alternatives are manual inspection or some score based on entity coverage.

Here we just look at the coverage of the entities present in $|V|$, the set of entities in the induced subgraph, by our final top-K documents. We compare this to the simple Lucene based top-K retrieval.

We tested all the 3 solutions (simple Lucene, greedy Set Cover, Set Cover with PageRank) over 50 different hand-picked queries. We ran PageRank with a restart probability of 15% on the induced subgraphs. We did not notice a consistent appreciable difference in the performances of greedy Set Cover with or without PageRank. The most likely reason for this is that the graph has a large number of similar nodes of low degrees and another large fraction of the entities do not have any outgoing edges (about 70% of our 28 million entities have no outgoing edges). Most of them have a single incoming edge in the subgraph and hence get a similar Importance assigned to them.

This is probably the reason for not noticing a very considerable difference. The table on this page contains results for a few queries. The results are the coverage of the entities in V on a 0-1 scale by the top-10 documents for that query as obtained just based on lucene, and on applying PageRank based Diversification on the documents.

Query Text	Lucene	PageRank Diversified
Merit Release Appearence	0.042	0.56
Gall Stones	0.014	0.274
Vietnam	0.009	0.446
Budget Truck Rental	0.031	0.08
Back to the Future	0.001	0.024
Coats Tire Equipment	0.006	0.34
Select Business Services	0.012	0.124
National Real Estate Settlement Services	0.012	0.14
Integrated Loan Services	0.005	0.1855
Kentucky Fried Chicken	0.009	0.24

6. CONCLUSION

Clearly, the Diversification Re-Ranking based on PageRank vastly outperforms a simple text based top-K retrieval in terms of overall related entity coverage. We believe that the results can be improved even further by improving the quality of the starting set of entities. A lucene index could probably do much better than the simple word IDF based retrieval that we used here.

We are also yet to explore a better way to evaluate our results. The entity coverage is not a very good evaluation metric since that is what the algorithm is designed to maximize (it only serves as a comparision against the baseline). We need to inspect ways to incorporate external knowledge for an impartial standalone diversity evaluation.

7. REFERENCES

[1] Wikimedia dumps - http://dumps.wikimedia.org/enwiki/

[2] Cucerzan, Silviu. ”Large-Scale Named Entity Disambiguation Based on Wikipedia Data.” EMNLP-CoNLL. Vol. 7. 2007.

[3] Dredze, Mark, et al. ”Entity disambiguation for knowledge base population.” Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics, 2010.

[4] Li, Yang, et al. ”Mining evidences for named entity disambiguation.” Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013.

[5] Deepak, P., et al. ”Entity Linking for Web Search Queries.” Advances in Information Retrieval. Springer International Publishing, 2015. 394-399.

[6] Carbonell, Jaime, and Jade Goldstein. ”The use of MMR, diversity-based reranking for reordering documents and producing summaries.” Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1998.

[7] Agrawal, Rakesh, et al. ”Diversifying search results.” Proceedings of the Second ACM International Conference on Web Search and Data Mining. ACM, 2009.

[8] Mei, Qiaozhu, Jian Guo, and Dragomir Radev. âĂİDivrank: the interplay of prestige and diversity in information networks.âĂİ Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2010.