

Assignment 1 (Due: March 3) (30 pts)

First generation search engines retrieved documents that matched keyword-based queries. Second generation search engines incorporated content-specific relevance ranking based on vector space model (TF-IDF) to deal with high recall. To overcome spamming, and to exploit collective web wisdom, third generation search engines incorporated content-independent information using Google's PageRank algorithm, and glean relative semantic emphasis of various words based on features such as fonts, distance between query term hits, etc. Future search engines will incorporate context, profile, and past history associated with a user to personalize search, and apply additional reasoning to improve satisfaction of information need.

In this project, you will design, implement, and run Google's PageRank algorithm and its variant on different datasets. In general, PageRank is a content-independent approach to ranking entities (e.g., documents, people, etc.).

This project has two phases. In Phase I, you will develop a framework for analyzing a dataset, build appropriate link graph, determine ranking, and produce a suitable persistent data structure. In Phase II, you will experiment with (i) HTML documents to derive content-independent ranking of documents implicit in the *hyperlink structure* and (ii) Social media data, such as, Twitter, Facebook, etc. to derive content-independent influence measure / reputation of people implicit in the entity associations, such as Twitter *follower relationship*, Facebook's friends relationship, etc. Phase III is open-ended, and you are encouraged to discuss various other features of social media (posts, tweets, etc) that can be considered to improve determination of influence / reputation.

[For completeness, in the context of developing a search engine, one can either (1) implement an inverted index for Boolean Retrieval Model, and then rank query result sets using PageRank ranking, or (2) implement an inverted index for Vector Space Model and then rank query result sets by weighted aggregation of VSM ranking and PageRank ranking.]

The project may be done individually, or in a group of two. Both members of a group are expected to contribute to all aspects of the project: design, implementation, documentation, and testing.

Milestones

1. Extract individual documents / entity information and build link graph.
2. Compute PageRank, and store associated data structures to disk.

Benchmarks

1. Which collections are you using for your project?
2. Corpus statistics:
 - How many documents/entities are there in the collection?
 - What is the size of the link graph?
3. PageRank computation:
 - How many iterations did it take for the PageRank computation to converge?
 - How much time and memory (main and disk) is required for the PageRank computation?

Visualization of results

Show the link graph and the associated PageRank results.

Potential Extensions

As mentioned earlier, a number of content-based features are used to improve relevance ranking for documents dataset. What content-based features can be gleaned and used to improve influence/reputation ranking for social media data?

Deliverables

You will turn in the following *three* items in the form of a zip-archive: *pr.zip*.

Design document Describe your application's architecture and major data structures. Describe how your system implements PageRank computation. List any external libraries or resources required by your application.

You are encouraged to write the design document prior to writing the code for planning purposes, although it will certainly change during the course of the project. Aim to make the design decisions explicit enough so that one of your classmates could implement your program from it (e.g., What is a link? What is the format of stored structures?).

Benchmark output The information sought under benchmark section. (In your case, you will have at least two sets: one for HTML documents and another for Twitter/Facebook data.)

Code and accompanying documentation Your code needs to be well documented with comments. The comments should not literally verbalize what the code is doing, instead, it should provide a high-level, abstract summary of what the code is supposed to accomplish, listing aspects such as tacit assumptions, invariants, etc.

You also need to include a README.txt file that describes how to compile your program and run it on datasets.

To turn in your solution in *.zip with the accompanying README*.txt, run the following command on unixapps1.wright.edu:

```
/common/public/tkprasad/cs707/turnin-pa1 *.zip README*.txt
```

You are also expected to demo your program to me.

Implementation

You may code your project in any programming language (such as C++, C#, Java or Python). You can choose any document collection to test your program but your code should be adaptable easily to be run on a new test collection for scalability testing.

References

- [1] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", <http://infolab.stanford.edu/~backrub/google.html>.
- [2] David Austin, "How Google Finds Your Needle in the Web's Haystack", <http://www.ams.org/featurecolumn/archive/pagerank.html>.
- [3] Damon Horowitz and Sepandar D. Kamvar, "The Anatomy of a Large-Scale Social Search Engine", <http://vark.com/aardvarkFinalWWW2010.pdf>.