The HITS algorithm

Jussi Pakkanen

Tik-122.102 Analysis of binary data

jussi.pakkanen@hut.fi

Overview

- the problem of relevant documents
- difficulties of the world wide web
- the algorithm
- results
- conclusions

Finding relevant documents

- the problem is to find “relevant” documents in some collection
  - relevance is usually subjective
  - exact measuring is difficult
- mere keyword searching is insufficient
- theoretically best approach: a system that understands language and reasoning
  - infeasible, AI-complete

Query types

- three basic types
  - specific queries
  - broad topic queries
  - similar page queries
- quite similar, but all have their own set of problems

The world wide web

- a huge set of interconnected documents
- over 90% of documents are useless, how to find relevant ones
- any keyword query most likely returns thousands of pages
- a human can not browse this amount efficiently
- pages must be ranked efficiently somehow

The basic idea of HITS

- hyperlinks encode relationships between pages
- most links are placed intentionally by human beings
  - presumably they have very high information content
- pages that get more links are most likely important

Definitions

- web pages form a graph $G = (V, E)$
- every page has two features: in-degree and out-degree
  - they tell the amount of links to and from the page
- *hub pages* link together many other pages
- desired matches are called *authority pages*

Computing hubs and authorities

- an iterative algorithm
- assign to each node p a hub weight y and authority weight x
- compute new values of y and x for every node
- normalize y and x
- repeat until equilibrium is reached

The main iteration

Getting results

- it can be proven that the iteration converges
- the pages that have the largest x values are authorities
- correspondingly large y values indicate a hub
- sorting by x value should give the most relevant pages
- however this is infeasible for large collections

Narrowing search

- do a keyword search on e.g. *AltaVista*
- the 200 first matches form a *root set*
- add to *base set*
  - all pages the root set pages point to
  - random (max *d*) pages pointing to root set
- do further processing only on these pages

Basic results

Similar page queries

Refined search

- the basic system works fine, but some queries are problematic
- “jaguar” can mean a car, sports teams or a game console
- “abortion” divides pages very strictly to pro-choice and pro-life sections
- a good search system should be able to separate these

Subgraphs

- a basic idea: pages dealing with jaguars as animals are not that much linked to pages about the Jaguar game console
- different sets form densely connected subgraphs
- these can be found using eigenvectors
- unfortunately, we cannot know beforehand which eigenvector corresponds to which group

Refined search results

Conclusions

- a prototype for a web search engine was presented
- by analyzing binary data (link info) the quality of the results is increased
- not computationally heavy, but requires a lot of storage space/bandwidth
- the results seem fairly reasonable