**Web Search Engine – Project Report**

**Search Overflow**

*Narasimman Sairam (ns3184), Manasa Kunaparaju (mk5376)*

**Objective of the project:**

A Query Engine to leverage the body of knowledge created by the socio-professional media, to recommend high quality, embeddable code.

*Category: Specialized search engine: Implement a search engine that gives high quality results for some particular class of queries*.

Q&A services like stack overflow are filling archives with millions of entries that contribute to the body of knowledge in software development” and they often become the substitute of the official product documentation.

When a developer is writing code and has to check online for a syntax or api related information, typically the developer would:

* Open the browser
* Google for the particular question (Ex: how to iterate over a map in Java)
* From the list of responses, (usually stack overflow) open couple of tabs of stack overflow pages that answers the query.
* Look for the best answer (Top voted in stack overflow website)
* Close the other tabs
* Copy paste the top rated answer to the development environment and continue working

Our search engine will do all the hard work of querying for the best answer and suggest the developer with the top rated response from the stack Overflow website. The answer to the query will be exactly one response (top rated).

Searching for code examples is possible using Stack Overflow directly. However, using designated code search tools on top of Stack Overflow may provide better results in terms of streamlining the various activities involved in example centric development (search, evaluation, and embedding).

**Sketch of the architecture**

The below is a broad overview of architecture of the system.

* Stackoverflow Data dump in XML format, converted into a database.
* An indexer, with a custom scoring function, which indexes each post in the data dump based on the title and body of the post.
* Retriever to retrieve the top 100 best matches based on the provided query.
* A Ranking algorithm to further fine tune the best match
* A web interface that accepts query and returns the result

Retriever

Run Query

Build

Query

Search User

Interface

Post retrieved

Users

Based on Query

Render Results

Best result sent to retriever

Page Rank Algo

Index

Document Analysis

Data Dump

XML format

Builds Index Results are ranked

Database

**Systematic Evaluation of the Project**

* The data dump is available at the Stackexchange Archive.
  + The information available were: Posts, Post Links, Post History, Tags, Users, Votes, Badges, Comments
  + We require only Posts as it has all the required information including the user information and all the answers.
  + All the posts are separated into 2 types: Questions and Answers. We use only the question to build the index.
* The index is built using a custom scoring function. The scoring function is a weighted average lucene score and the normalized vote score of the questions.
* Based on the input query the top 100 results are retrieved.
* These 100 results are further sorted in a priority queue based on the answer votes and the user ranking and the initial weighted score.
* The web interface is used to pass the queries to the retriever and get the result to provide to the user.

**List of external software used:**

Apache Lucene - for indexing the pages

Other tools and software to use - Yet to be decided

**Examples of Project Success and Failure:**

1. The search engine provides good results for cases with accepted answers. Ex: Query: ”Hashmap Iterate”. The top results in Google are present in the top 100 results that are returned.
2. Cases with no accepted answers, even ones with good votes are not being considered for preview.

**Next Steps:**

1. The customized scoring can be experimented on more. With experimentation, we could come up with a better normalization function and an efficient way to assign the weights to different factors.