CS 221 PROJECT 3 – Complete Search Engine

KARTHIK PRASAD 42686317
PHANI SHEKHAR MANTRIPRAGADA 85686586
RISHABH SHAH 79403075

NDCG Score computation:

Queries	Old NDCG Score	NDCG Score after improvements
mondego	0.777	0.791
machine learning	0.170	0.170
software engineering	0.170	0.588
security	0.170	0.503
student affairs	0.342	0.926
graduate courses	0.169	0.515
Crista Lopes	0.169	0.732
REST	0.797	0.829
computer games	0.169	0.733
information retrieval	0.452	0.694

List of Improvements made for Final search engine:

- Introduced new database of 'Anchor Text' to 'Target Website' for **1-gram** and **2-gram**.
- Introduced new database of 'Title Text' to 'Target Website' for **1-gram** and **2-gram**.
- Computed **PageRank** for each website.
- Introduced a new score that determines **proximity** (based on **positions**) for 1+ word queries cited from a popular paper - "http://sifaka.cs.uiuc.edu/czhai/pub/sigir07-prox.pdf"
- **Stemming** for words based on Snowball Stemmer (improvised Porter Stemmer).

Search Engine Algorithm:

1. Each feature (TF-IDF, Anchor Text, Title Text, Proximity measure) return a list of documents/URLs (Referred as DOC_ID) along with their score (briefly explained in next section).
2. We divide our list of documents in to three buckets:
    a. Bucket 1: This bucket contains list of documents that are seen in Anchor Text and Title Text Database.

   b. Bucket 2: This bucket contains list of documents that are seen either in Anchor Text or Title Text Database, but not in both.
   c. Bucket 3: This bucket contains list of all other documents.
3. Intuitively, results from Bucket-1 are better than Bucket-2 and Bucket-3. So we return the top 10 results or top N (N- is number of DOC_IDs in Bucket-1) results whichever is minimum. These results are sorted based on harmonic mean of TF-IDF and proximity score.
4. If bucket-1 is exhausted, we go to Bucket-2 and retrieve the remaining results based on other parameters.
5. Similarly, after exhaustion of Bucket-2, we move to Bucket-3 and retrieve the missing results based on a cumulative score computed using PageRank, TF-IDF and Proximity Score.

Algorithm for Computing NDCG (top 5 results):

DCG scores are computed using the following formula and normalized using Google's DCG scores.

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where the relevance(rel_i) is computed as follows:

- First 3 results from google have a value of 3,
- next 2 results have a value of 2,
- All the next results have a value of 1.

Individual Score Details:

1. Anchor Text and Title Text Database:
   - For 1+ word queries, we assign higher weight if there is a match in 2-gram database.
   - Score also depends on number of documents returned for the query. Score for multiple documents returned for same query would be less, as compared to few documents retrieved for the same query.
   - Fine Tuning based on number of results in anchor text. For e.g: more weight is given to DOC_IDS when less number of DOC_IDS are retrieved and vice versa.

2. Proximity Score:
   - It is calculated based on negative exponential of the minimum distance between pair of words in a query. Here is the formula that we used to calculate the score:

     $\text{Proximity Score} = \log[1 + (\alpha + \exp(-\delta(Q, D)))]$.

   [$\alpha = 0.3$, δ = minimum distance between any pair of words of a query and document]

3. Page Rank:
    - We built a graph based on outlinks from each document and source document itself; nodes are documents and edges are outlinks coming from each document.
    - We have used Python Network X library to generate the PageRank from the input graph. Damping parameter was set to 0.85.

4. Term Frequency – Inverse Document Frequency Matrix:
    - Our TF-IDF score was identical from the class-notes.

Snapshot of our Search Engine for “Student affairs” query: