Summary

- PageRank
  - Static score based on random walk
  - Long-term visit rates
  - Calculated based on link structure.
    - Not from actual processes wandering the web
- PageRank for topics
  - Know this
- HITS
  - PageRank for hub and authority model
  - Optional
Assignment 07

Calculate Cosine Similarity Score

• Input
  • Query
  • Posting List

• Output
  • List of 10 top ranked documents
Calculate Cosine Similarity Score

- Remember what this is about
  - A query as a vector
  - A corpus as a term-document matrix
    - Where each document is a column in the matrix

\[
sim(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{||\vec{V}(q)|| ||\vec{V}(d)||}
\]
Calculate Cosine Similarity Score

- We are **not** going to calculate the similarity score of a query with every document
- That would be inefficient.
- Many scores are zero.
- We are **not** going to actually create a term-document matrix
- The posting list has all the information that we need to calculate the similarity scores
We are going to calculate the cosine similarity score, but in a clever way.

Here are some constants we will need:

- The number of documents in the posting list (aka corpus).
  - Figure this out when creating the corpus (new thing)
- The document frequency of a term
  - This should be the number of items in a row of the posting list. (each term has its own row)
- The term frequency of a term in a document.
  - Different for every term document pair.
Calculate Cosine Similarity Score

• Steps
  • Get a query from the user
  • Convert it to TF-IDF scores

\[ tfidf(t, q) = WTF(t, q) \times \log \left( \frac{|corpus|}{df_{t, q}} \right) \]

\[
\begin{align*}
WTF(t, q) &= \begin{cases} 
1 & \text{if } tf_{t, q} = 0 \\
2 & \text{then } return(0) \\
3 & \text{else } return(1 + \log(tf_{t, q}))
\end{cases}
\end{align*}
\]
Calculate Cosine Similarity Score

- “UCI Informatics Professors”
- 3 terms {“UCI”, “Informatics”, “Professors”}
- 3 TF-IDF scores

- Size of the corpus comes from the posting list
- The document frequency of “UCI” comes from the number of entries in the posting list for “UCI”
- use 1 if your posting list is too small
- The term frequency is 1/3

\[
\text{tfidf} \left( \text{“UCI”}, \text{“UCI Informatics Professors”} \right) = 1 + \log(1/3) \times \log \left( \frac{|\text{corpus}|}{\text{df} \text{“UCI”}} \right)
\]
Calculate Cosine Similarity Score

• Steps
  • Get a query from the user
  • Convert it to TF-IDF scores
  • Create a data structure that is indexed by documents
    • Which will accumulate scores for the documents
    • so like, Scores = new HashMap<String, Double>()
Calculate Cosine Similarity Score

- Steps
  - Get a query from the user
  - Convert it to TF-IDF scores
  - Create a data structure that is indexed by documents
    - Which will accumulate scores for the documents
    - so like, Scores = new HashMap<String,Double>()
  - For each term in the query
    - Get the posting list for the term
    - For each document that has that term we are going to update the entry in Scores
Calculate Cosine Similarity Score

- **Steps**
  - For each term in the query
    - Get the posting list for the term
  - For each document that has that term we are going to update the entry in Scores
    - Scores[d] += TF-IDF(term,query) * TF-IDF(term, document)
Calculate Cosine Similarity Score

- At the end of this we will have the data structure Scores
- Which for “UCI Informatics Professors” required looking up 3 posting lists
- Finally the scores must be normalized so we can compare them against each other.
- Create a new data-structure like Scores called Magnitude
- For each term in the entire posting list
  - For each document represented in Scores
    - Magnitude[document] += TF-IDF(term, document)^2
Calculate Cosine Similarity Score

- Now we have Scores and Magnitude
- Now we calculate the highest rankings
- For each document in Scores
  - Double $x = \frac{\text{Scores}[\text{document}]}{\sqrt{\text{Magnitude}[\text{document}]}}$
Calculate Cosine Similarity Score

- Summary
  - Get query from user, transform to TF-IDF
  - Pull out a few postings to calculate scores
  - Look at every posting to calculate magnitudes
  - Calculate final scores
  - Output URLs and scores of highest documents
Calculate Cosine Similarity Score

**CosineScore**($q$)

1. **Initialize**($Scores[d ∈ D]$)
2. **Initialize**($Magnitude[d ∈ D]$)
3. **for each term**($t ∈ q$)
   4. **do** $p ←$ **FetchPostingsList**($t$)
   5. $df_t ←$ **GetCorpusWideStats**($p$)
   6. $α_{t,q} ←$ **WeightInQuery**($t, q, df_t$)
   7. **for each** $\{d, tf_t, d\} ∈ p$
      8. **do** $Scores[d] + = α_{t,q} · $ **WeightInDocument**($t, q, df_t$)
   9. **for** $d ∈ Scores$
   10. **do** **Normalize**($Scores[d], Magnitude[d]$)
11. **return** top $K ∈ Scores$