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

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

• 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

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

\[
\text{WTF}(t, q)
\]

1. **if** \( tf_{t,q} = 0 \)
2. **then** \( \text{return}(0) \)
3. **else** \( \text{return}(1 + \log(tf_{t,q})) \)
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

\[
tf idf(“UCI”, “UCI Informatics Professors”) = 1 + \log(1) \times \log\left(\frac{|\text{corpus}|}{df_{“UCI”} + 1}\right)
\]
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
    - \( \text{Magnitude}[\text{document}] += \text{TF-IDF}(\text{term, document})^2 \)
Calculate Cosine Similarity Score

• Now we have Scores and Magnitude
• Now we calculate the highest rankings
• For each document in Scores
  • Double $x = \text{Scores[document]} / \sqrt{\text{Magnitude[document]}}$
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

\texttt{CosineScore}(q)

1. \texttt{Initialize}(Scores[d \in D])
2. \texttt{Initialize}(Magnitude[d \in D])
3. \texttt{for each term}(t \in q)
   \hspace{1em} do \hspace{1em} p \leftarrow \texttt{FetchPostingsList}(t)
   \hspace{1em} df_t \leftarrow \texttt{GetCorpusWideStats}(p)
   \hspace{1em} \alpha_{t,q} \leftarrow \texttt{WeightInQuery}(t, q, df_t)
   \hspace{1em} \texttt{for each } \{d, tf_{t,d}\} \in p
   \hspace{2em} do \hspace{1em} \texttt{Scores}[d] + = \alpha_{t,q} \cdot \texttt{WeightInDocument}(t, q, df_t)
4. \texttt{for } d \in \texttt{Scores}
5. \hspace{1em} do \hspace{1em} \texttt{Normalize}(\texttt{Scores}[d], \texttt{Magnitude}[d])
6. \texttt{return } top \ K \in \texttt{Scores}
Evaluation in IR
Introduction to Information Retrieval
CS 221
Donald J. Patterson

Content adapted from Hinrich Schütze
http://www.informationretrieval.org
~Sage~
http://www.flickr.com/photos/vickispix/2089649326/
Evaluation in IR
Introduction to Information Retrieval
CS 221
Donald J. Patterson

Content adapted from Hinrich Schütze
http://www.informationretrieval.org
~Sage~
http://www.flickr.com/photos/vickispix/2089649326/
Evaluation in IR

Outline

- Intro to Evaluation
- Standard Test Collections
- Evaluation of Unranked Retrieval
- Evaluation of Ranked Retrieval
- Assessing relevance
- Broader perspectives
- Result Snippets
Intro to Evaluation

- There are many implementation decisions to be made in an IR system
  - Crawler
    - Depth-first or breadth-first?
  - Indexer
    - Use zones?
    - Which zones?
  - Use stemming?
  - Use multi-word phrases? Which ones?
There are many implementation decisions to be made in an IR system:

- Query
  - Ranked Results?
  - PageRank?
- Which formula do we use in the TF-IDF Matrix?
- Should we use Latent Semantic Indexing?
  - How many dimensions should we reduce?
Intro to Evaluation

- There are many implementation decisions to be made in an IR system
- Results
  - How many do we show?
  - Do we show summaries?
  - Do we group them into categories?
  - Do we personalize the rankings?
  - Do we display graphically?
Intro to Evaluation
Intro to Evaluation

- How can we evaluate whether we made good decisions or not?
Intro to Evaluation

- How can we evaluate whether we made good decisions or not?
- Measure them
Measures for a search engine

- How fast does it index?
  - Number of documents per hour
  - Average document size

- How fast does it search
  - Latency as a function of index size

- Expressiveness of query language
  - Ability to express complex information needs
  - Speed on complex queries
Measures for a search engine

- We can measure all of these things:
  - We can quantify size and speed
  - We can make this precise
- What about user happiness?
  - What is this?
  - Speed of response/size of index are factors
  - But fast, useless answers won’t make a user happy
- Need to quantify user happiness also.
Measuring user happiness

• Issue: Who is the user we are trying to make happy?
• It depends.
Measuring **stakeholder** happiness

- Issue: Who is the user we are trying to make happy?
  - Web engine:
    - The user finds what they want.
    - Measure whether or not they come back.