Bronco ID: 0 1 5 2 6 2 6 2 4

Last Name: Nguyen

First Name: Loc

1a. The first factor is structure in database versus unstructured text in documents makes querying easier with database. The second factor is semantics, given that there are many ways to describe the same idea in natural language like English or Korean. Comparing text means that some documents will be better matches than other even though they may contain the "same" information for documents. In a database, information are categorized such as ID number or account number that doesn't have alternative semantics, making DB easier to query than documents.

1b. Before, researchers would use statistical models instead of linguistic models to measure relevance. This may result in situation where a sentence such as "A computer can computer computer the most computer" being marked with "High Relevance" even though it doesn't make sense. Currently, researchers are using ranking model implemented in search engines such as Google, Bing, etc. that crawls the web, index the results, and evaluate relevance based on user interaction, and effectively rank results, trying to maximize precision and recall. Search engines have to be effective and efficient, scalable with a massive user base, and adaptable to changes & specific problems.

2a. Web-search engines use text to conduct ad-hoc search, finding documents for arbitrary text query such as finding a research document for cutting-edge research on Google Scholar.

2b. Vertical search engine use images to filter information, identify relevant information for a task such as a movie recommendation on Netflix or a video recommendation on YouTube

2c. Enterprise search engine use video and classify it based on specific criteria, finding relevant labels for document and are used in tasks such as find genre of books or movies or sports.

2d. Desktop search engine use scanned documents to answer questions, such as finding information about what's the distance from Earth to the Moon?

2e. P2P search is different from the rest because it is decentralized, and information is not located on a single computer or server but is distributed throughout a network of devices, whose each holds a chunk of information. Services such as Napster where users can share & download music from other users.

3a. Classification of pictures using tags into one of the three folders

3b.Ad-hoc search to find Python books exercises

3c. Question answering to answer a question of “What is the largest river in the world?”

3d. Filtering to recommend movies on Netflix

4a. Topic relevance is dealing with how relevant search results are based on topic as weather, laws, schools. User relevance is dealing with how relevant search results are to users, incorporating a lot of features. User relevance deals with a lot of topics but the content is catered to the user. It's important to consider both because the users needs information that is both relevant to the topic and user in order for the information to be useful.

4b. A search query of "police laws" for a user in the US returns the result of police laws in Europe and police laws from 7 years ago. This is topic relevant but not user relevant because the information isn't relevant or useful to the user living in the US.

4c. Inversely, a query for "Taylor Swift" on Google might yield results such as her height, hometown, background but doesn't yield results of her music, which is the reason why someone is searching her up.

4d. A query of "Local news" for a person who lives around the San Bernardino area, is a college student, and likes going to concert, return the results of

> "Ed Sheeran is hosting a concert in downtown Fontana"
>
> "New college scholarships for 2023-2024 school year for students in SB county area"
>
> "DMV requires all college students to carry campus ID at all time when on campus"
>
> "Disneyland is having a promotional coupon for residents of SB county, for fall 2023"

These search results are relevant to the topic of "local news" and to the user whose profile indicates that they would be interested in these results.

5a. Precision = 2 green hits / 3 object recalled = 0.667 or 66.7%

Recall = 2 green hits / 3 green existed = 0.667 or 66.7%

5b. Precision = 3 green hits / 5 object recalled = 0.6 or 60%

Recall = 3 green hits / 3 green existed = 1 or 100%

5c. Precision = 2 green hits / 2 object recalled = 1 or 100%

Recall = 2 green hits / 3 green existed = 0.667 or 66.7%

5d. Precision = 0 green hits / 2 object recalled = 0 or 0%

Recall = 0 green hits / 3 green existed = 0 or 0%

6.

Indexing process starts with web crawlers following links to identify and stores document & metadata for indexing during the text acquisition process. Next, during text transformation, the parser tokenizes words through syntax of markup language & other formatting to identify structure, transforming words into index terms & features. Also, it removes stop words and stemming to increase efficiency. Then, during index creation, index terms are counted, and weights are computed by the ranking algorithm to determine relevance. Data structures such as inverted index are used to house all the index terms.

Query process starts with user interaction typing in a query to the search engines, supported by spell checking and query suggestions. Results are served to the user and the ranking algorithm scores how relevant each document is based on computed weights and adjust according to user interaction. Then, the query data and clickthrough data is logged for evaluation in a log database. Ranking analysis then measure the precision and recall using log data to analyze and improve performance.

7.

Step 1: Remove stop words {I, and, she, her, they, their} and consolidate stemming to generate {love, cat, dog}

Step 2: Index terms matrix – {love, cat, dog}

Step 3: Term frequency chart for each document

Document 1

Term	Term Count
Love	1
Cat	2

Document 2

Term	Term Count
Love	1
Dog	1

Document 3

Term	Term Count
Love	1
Dog	1
Cat	1

Tf-idf Calculation:

Tf-idf(“love”,d1, D) = 1/3 * log(3 / 3) = 1/3 * 0 = 0

Tf-idf(“love”,d2, D) = 1/2 * log(3 / 3) = 1/2 * 0 = 0

Tf-idf(“love”,d3, D) = 1/3 * log(3 / 3) = 1/3 * 0 = 0

Tf-idf(“cat”,d1, D) = 2/3 * log(3 / 2) = 2/3 * 0.176 = 0.12

Tf-idf(“cat”,d2, D) = 0/2 * log(3 / 2) = 0 * 0.176 = 0

Tf-idf(“cat”,d3, D) = 1/3 * log(3 / 2) = 1/3 * 0.176 = 0.06

Tf-idf(“dog”,d1, D) = 0/3 * log(3 / 2) = 0/3 * 0.176 = 0

Tf-idf(“dog”,d2, D) = 1/2 * log(3 / 2) = 1/2 * 0.176 = 0.09

Tf-idf(“dog”,d3, D) = 1/3 * log(3 / 2) = 1/3 * 0.176 = 0.06

Tf-idf representation

	Love	Cat	Dog
Document 1	0	0.12	0
Document 2	0	0	0.09
Document 3	0	0.06	0.06

8. https://github.com/Skyhorizon2021/CS4250/blob/master/search_engine.py