Watson

Documentation

Team: Buboi Oana-Andreea, Butcă Andreea-Cristina, Ifrim Gianina-Cosmina, Pitac Luminița-Gabriela, Stoica Andreea

Project outline:

- The main purpose of this project is to build a part of IBM's Watson
- Watson is a project that aims to create a simplified search engine capable of retrieving relevant Wikipedia articles based on user queries. The project utilizes a subset of questions from Jeopardy games, with answers corresponding to Wikipedia pages. The system performs data preprocessing, indexing, and search operations using Lucene.

How it works?

There are five functionalities:

- Create index
  - It deletes the existing indexes from the wiki-index folder
  - Creates a document for every article with a stored index on title and category
- Run question
  - The user inputs a category and a clue
  - The system converts them into a query
  - The query is used to search for the most relevant article
  - The output is a list of the most relevant articles with its' size equal to the max number of retrieved documents
  - If the question is from questions.txt, the output will also contain if the result is a hit or a perfect hit
- Run question with AI
  - The user inputs a category and a clue
  - The system converts them into a query

    - The query is used to search for the most relevant article
    - The output is a list of the most relevant articles with its' size equal to the max number of retrieved documents and the same list reranked by the AI
    - If the question is from questions.txt, the output will also contain if the result is a hit or a perfect hit
- Run default questions and measure
    - The system searches the most relevant article using the categories and clues from questions.txt
    - The output is a list of the most relevant articles with its' size equal to the max number of retrieved documents and if the result is a hit or a perfect hit
    - The output also contains the NDCG and MRR metrics
- Set max number of retrieved documents (> 0 – default 10)
    - The user inputs the desired max number of retrieved documents

Data used (Dataset):

- 100 questions from previous Jeopardy games, whose answers appear as Wikipedia pages. The questions are listed in a single file, with 4 lines per question, in the following format: CATEGORY CLUE ANSWER NEWLINE.
- A collection of approximately 280,000 Wikipedia pages, which include the correct answers for the above 100 questions. The pages are stored in 80 files (thus each file contains several thousand pages). Each page starts with its title, encased in double square brackets. For example, BBC's page starts with "[[BBC]]".

Notes:

- Program comes pre-loaded and ready to run
- Indexing and searching techniques are done with Java Lucene API
- The program only runs from IDE Console

How it is done:

- There is a TextToXml class which reads data from the "wiki_labels.txt," "wiki_links.txt," and "wiki_abstracts.txt" files. It decodes URLs, compares titles, and constructs an XML file ("wiki_data.xml") containing structured article information.
- There is also the Article class, the ArticleIndexer class for the index, the ArticleParser class for parsing the articles, and the ArticleParseResult class for storing the parsing results
- For indexing there is an IndexWriter which uses „stop-words” and „stemming”. It uses an EnglishAnalyzer for text analysis, and the index includes fields for title, category, and body of the articles. The ArticleParser is expected to provide articles, and the indexArticle method from ArticleIndexer adds each article to the Lucene index. The index is created in the specified directory (Wiki-Index)
- For ranking the articles, the Lucene score is used so that the most relevant ones appear at the top of the search results. The scoring algorithm in Lucene is based on the Vector Space Model, which represents documents and queries as vectors in a multi-

dimensional space. The relevance score is calculated by measuring the similarity between the query vector and the document vector

Issues specific to Wikipedia:

- Some pages do not contain text but they redirect the user to the „parent" page –> these pages will be linked to the „parent" page

How the query is built:

- If the category is not mentioned, the query contains only the clue without punctuation marks
- If the category is mentioned, the query is built with a disjunction between the category and the clue without punctuation marks

Measuring the performance:

- The normalized discounted cumulative gain (NDCG) and the mean reciprocal rank (MRR) metrics are used
- NDCG is a metric that evaluates the quality of recommendation and information retrieval systems. It helps measure the algorithm's ability to sort items based on relevance. NDCG considers the entire ranking order and assigns higher weights to relevant items at higher positions. This is useful in many scenarios where you expect the items to be sorted by relevance. This differentiates NDCG from purely predictive quality metrics like Precision and Recall

- MRR is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness
- NDCG performance ranges from 0.000 for non-hit to 0.220 for a perfect hit
- MRR performance: 0.353

Error analyzing:

- The system shows hits and perfect hits; perfect hits represent the number of correct articles returned on the first position in the list, whereas hits is the number of correct articles in the list returned
- In the best system, there are 51 hits and 29 perfect hits out of a total of 100 questions
- The correct questions can be answered by such a simple system due to the presence of all/almost all the words from the query in the content of the page. Lucene is known for its powerful full-text search capabilities and can efficiently index and retrieve relevant documents
- The number of correct answers can be due to the fact that both the documents and the questions with the categories have been tokenized while also stemming and the elimination of stop words was applied to them. The operator „OR” does not force the system to match the content of the document 100% with the words from the question. It is enough if only some of the words can be found within the document

- Wrong answers occur due to the fact that the system cannot interpret the meaning of the words, it only searches for their occurrences
- Another case could be the search for questions in certain categories other than the categories that the system processed
- Observed Error Categories**:**
    - Uniqueness: Lack of distinctive keywords leads to missed matches;
    - Ambiguous Queries: Clues too wide-ranging or vague;
    - Insufficient contextual comprehension: The system encounters difficulties when dealing with synonyms and variations in meaning;

Improving retrieval:

- ChatGPT is used to rerank the top K pages produced
- If the list of pages sent to ChatGPT contains the correct article, it may change the hit to a perfect hit
- If the list of pages sent to ChatGPT does not contain the correct article, nothing changes about the presence of the correct article in the list

```
Enter the question category (press enter for no category):
THE RESIDENTS
Enter a search query:
Don Knotts took over from Norman Fell as the resident landlord on this sitcom


* Performing search...
* Results Found! (Elapsed Time: 0.0109941 Seconds)

----------------------------
| Search Result Top 10 found |
----------------------------
1.  Don Knotts  (score: 21.049486 )
2.  Steward's Lodge (score: 17.660767 )
3.  Burwash Hall    (score: 16.771036 )
4.  Crinkle Crags   (score: 15.567525 )
5.  Three's Company (score: 15.1590395 )
6.  Quirinal Palace (score: 14.835606 )
7.  Walter Knott    (score: 14.508495 )
8.  Dublin Castle   (score: 14.316525 )
9.  Gatchina    (score: 14.295081 )
10.     Buen Retiro Palace  (score: 14.111838 )
** Hit found: true
** Perfect hit found: false
--------------------------
| AI Result Top 10 found |
--------------------------
1.  Three's Company
2.   Norman Fell
3.   Don Knotts
** Hit found: true
** Perfect hit found: true
```

For implementing this project, the team members had the following tasks:

- Buboi Oana-Andreea – documentation & power point, research, error analysis
- Butcă Andreea-Cristina – documentation & power point, research, error analysis
- Ifrim Gianina-Cosmina – indexing & retrieval, video presentation, research, error analysis
- Pitac Luminița-Gabriela – measuring performance (NDCG&MRR), research, error analysis
- Stoica Andreea – improving retrieval & chatGPT, research, error analysis