Building a part of IBM Watson - Report

Students:

Răuță Alexandra-Elena (DSI), Popa Iulian-Alexandru (DB), Stoian Silviu (DB), Ardelean Daniel-George(DSI), Chirvasiu Mihai(DSI)

Overview

This document serves as the report for our group project on creating a part of IBM's Watson by creating a Query Engine capable of responding to jeopardy quiz questions.

Our actual implementation can be sectioned into 4 distinct parts:

1. Building the index
2. Building the query and searching the results
3. Evaluating the results
4. Measuring the performance

Building the index

When it comes to building the index we used the dataset :

https://www.dropbox.com/s/nzlb96ejt3lhd7g/wiki-subset-20140602.tar.gz?dl=0

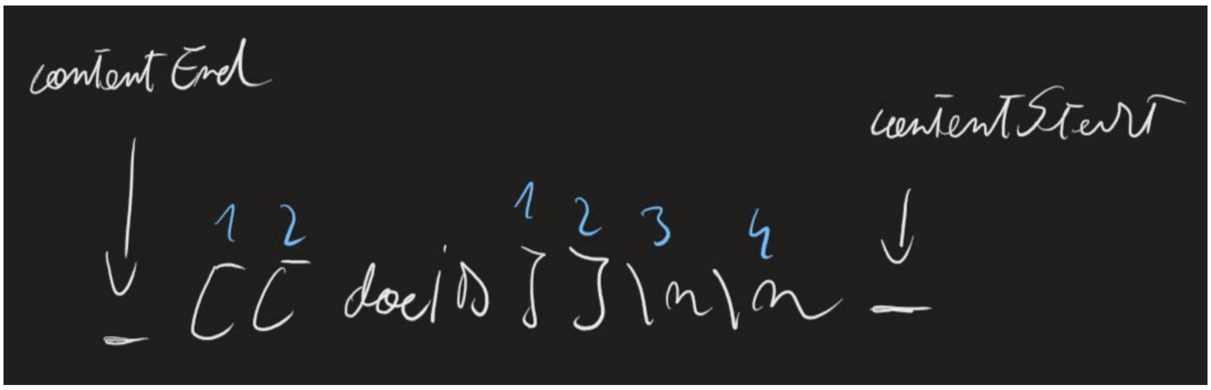
Consisting of .txt files with multiple Wikipedia pages in them. Considering that the index should be built by indexing the documents from the actual files we had to analyse the files and come up with a parsing strategy to build the index.

For the strategy we went with an approach that use 2 pointers: contentStart, contentEnd, pointers which mark the beginning and end of the content of a document in the text file. The content of the files follows a strict set of rules, so we follow these steps:

1. Get the documents title
2. Get the content of the document
3. Add the document to the index

In the documents each documents has it's title in this format:

[[Title]] followed by an empty line, then with the content of the document.



After correctly setting the pointer from where the content starts we increment the content of the document by a set offset (10 in our case), but we make sure to look after specific special cases.

- We should stop when we find beginning of titles "\n\n\n[["
- But we need to exclude 2 existing variations: "\n\n\n[[File:" and "\n\n\n[[Image:"

We increment the contentEnd pointer with the offset until we reach the title of the next document or the end of the actual file.

After reaching the next doc (or the end of the file) we substract the "substring" from contentStart to contentEnd (which will yield the actual content of the document) and we apply the following transformations.

We are using the CoreNLP API provided by Stanford, to apply the process of tokenization and lemmatization (process which would be hard to implement on our own). By lemmatizing the content of the document we reduce inflexions of terms and effectively normalize and reduce the overall size of the index, making the search more efficient and with higher chances of being relevant.

Particularly we used the: *tokenize*, *ssplit* , *pos* and *lemma* annotators which will tokenize the text, split it into sentences, annotate each token with it's part of speech (verb, substantive, adjective, etc..) and lemmatize the tokens to reduce inflexions (was, were, being, all become "be").

Finally we add the lemmatized content of the document along with the documents title into a Document base class, to be added to the index.

Building the query and searching the results

To actually interact with the index we ask it questions by providing queries. These queries also need to be transformed in order to properly communicate with the index. Firstly we transform the query into a Sentence class, to apply the lemmatization process. This will tokenize the query and in the process we also remove all punctuation signs. We create a Query class and tell it to look for the provided query in a certain field of the documents of the index (in our case the "content" field)

To communicate with the created index, Lucene provides us with a class called IndexSearcher. This searcher uses certain similarity strategies to rank the index pages and retrieve the most relevant ones.

The default similarity used is TF-IDF, similarity which yielded some good responses, but we went with an algorithm that extends TF-IDF, called BM25 which is more used in search engines where documents vary in length and importance. This algorithm uses a modified term frequency(TF) **k1** that saturates as the term frequency increases, and a **b** (IDF) weight which prioritizes terms that are more discriminative.

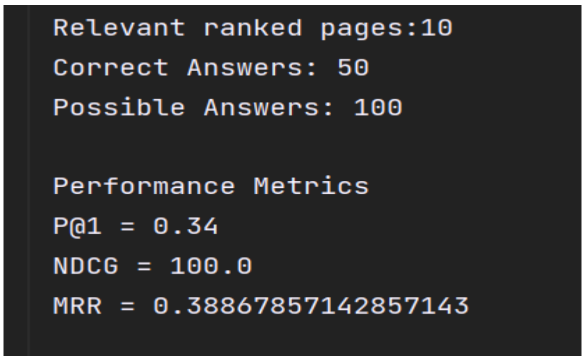
We did a hyperanalysis of these parameters by an exhaustive, incremental simulation. And found a good sweet spot for k1 = 0.5 and b = 0.8 , which yielded the most correct answers in a row of multiple close values.

This searcher will return a list of top n documents, which we will return as the final results.

Evaluating the results

Considering the fact that there exist questions with multiple variations of the correct answer, we need to check if the response we know it's true from the dataset, contains the same document title as the one found in the index (this ensures that an answer is correct even if it's a variation of the correct one).

By using the above strategies we get the following results for retrieving the top 10 pages for each query:



```
Relevant ranked pages:10
Correct Answers: 50
Possible Answers: 100

Performance Metrics
P@1 = 0.34
NDCG = 100.0
MRR = 0.38867857142857143
```

Answering question 1

Now let's answer the first question more succinctly.

- We made sure to create the index with all the documents present in all the files from the dataset

- We prepared the terms for indexing by tokenizing → split in sentences → get part the part of sentence → and finally lemmatizing the content of the document (using annotators from the CoreNLP API).

- Apart from the actual parsing of the documents (which is more detailed above) the only issue we came across was the existence of: [[File: and [[Image: which are not title of documents, and needed to be treated in a separate case, to not end the content of the document to early.

- The Jeopardy clue is implemented in the main function of the QueryEngine class, which takes the clue text and the category (we concatenate them for the final query).

- The query was built from by combining the category of the question and the clue (category + clue). We used all the words from this combination, and we used lemmatization on them before searching in the index.

Performance analysis

To know how performant is the system that is being developed, we need to introduce performance metrics, that will tell us how performant our system really is. Next, we will give a detailed description on this metrics, also stating why we ended up on choosing them in the end to evaluate our system.

These metrics are the following:

- P@1 or Precision at 1: is an evaluation metric used in information retrieval and machine learning. It is commonly employed in scenarios where the task involves ranking a list of items and determining the relevance of the top-ranked item. P@1 specifically measures the precision of the top-ranked result. In a binary classification context (relevant or not relevant), P@1 is the precision of the model at the first position in the ranked list. It is calculated as the number of relevant items at the top position divided by the total number of items at that position. Mathematically, it can be expressed as: $P@1 = \frac{noOfRelevantItemsAtPos1}{totalNoOfItemsAtPos1}$. In information retrieval, precision is a measure of how many of the retrieved items are relevant to the user's query. P@1 specifically focuses on the precision at the top of the ranked list, indicating how well the model performs when considering only the most highly ranked item. High precision at 1 indicates that the relevant items are frequently found at the top of the list, which is often desirable in applications where users are primarily interested in the top-ranked results.
- Normalized Discounted Cumulative Gain (NDCG) is an evaluation metric commonly used in information retrieval, machine learning, and recommendation systems to assess the quality of ranked lists. It takes into account both the relevance of retrieved items and their positions in the ranked list. NDCG is a normalized version of Discounted Cumulative Gain (DCG), and it addresses the issue of varying list lengths by normalizing the score. The computation done to evaluate the NDCG is the following:
    - $DCG = \sum_{i=1}^{N} \frac{2^{rel_i - 1}}{\log_2(i+1)}$, where rel_i is the relevance score of the item at position i in the ranked list and N is the total number of items in the list. (Discounted Cumulative Gain)
    - $IDCG = \sum_{i=1}^{N} \frac{2^{rel_{sorted_i} - 1}}{\log_2(i+1)}$, where rel_{sorted_i} is the relevance score of the item at position i in the ideal (perfectly ranked) list.
    - $NDCG = \frac{DCG}{IDCG}$ The NDCG score is the ratio of the actual DCG to the ideal DCG, which normalizes the result to a value between 0 and 1. A higher NDCG indicates a better-ranked list.

  NDCG is particularly useful in scenarios where the ranking of items is important, and it considers not only the relevance of items but also the position of those items in the ranked list. It helps in evaluating how well a model is performing in providing relevant items at the top of the list.

- Mean Reciprocal Rank (MRR) is an evaluation metric commonly used in information retrieval and recommendation systems to assess the effectiveness of a model in ranking relevant items. MRR is particularly relevant in scenarios where the task involves returning a ranked list of items based on their relevance to a user query. The MRR is calculated as the average of the reciprocals of the ranks of the first relevant item in each list. The reciprocal rank is the inverse of the position of the first relevant item. The formula for Mean Reciprocal Rank is as follows:

- $MRR = \frac{1}{|Q|} \cdot \sum_{i=1}^{|Q|} \frac{1}{rank_i}$, where $|Q|$ is the total number of queries and $rank_i$ is the position of the first relevant item in the ranked list for the i-th query.

In simpler terms, for each query, MRR calculates the reciprocal of the rank of the first relevant item and then takes the average across all queries. The result is a value between 0 and 1, with higher values indicating better performance.

MRR is particularly useful when the focus is on the position of the first relevant item rather than the entire ranking list. It provides a single, aggregated measure of the effectiveness of a ranking model in returning relevant items early in the list.

In our system the values for each performance metric, are P@1 = 0.34, NDCG = 100.0 and MRR = 0.38867857142857143. After the examination of these results, we decided to continue using and evaluating our system with the following: P@1 and MRR. Why did we choose to do this, is because as the list with the ranked documents for each query is already sorted by the document score(relevancy), then the NDCG (using the formula from above) will be equal to 1 for each query. As each definition states for each performance metric, P@1 showing how performant the system when looking only and the top ranked result and for MRR we calculate the reciprocal of the rank of the first relevant item, then average with across all the queries.

Knowing this we can say that our system is performing under average for ranking the most relevant item in the first position and for the average is almost average. Finally, we chose these two metrics because they express the best our system's ability to rank documents among all the documents provided.

Error analysis

Cateegory	Correct Answers	Incorrect Answers	Precision
NEWSPAPERS	4	1	80.0%
OLD YEAR'S RESOLUTIONS	2	2	50.0%
BROADWAY LYRICS	0	1	0.0%
POTPOURRI	3	0	100.0%
STATE OF THE ART MUSEUM	0	5	0.0%
"TIN" MEN	4	0	100.0%
UCLA CELEBRITY ALUMNI	4	1	80.0%
SERVICE ORGANIZATIONS	2	1	66.7%
AFRICAN CITIES	2	2	50.0%
HISTORICAL QUOTES	1	2	33.3%
CEMETERIES	1	1	50.0%
GOLDEN GLOBE WINNERS	1	2	33.3%
HISTORICAL HODGEPODGE	2	2	50.0%
CONSERVATION	1	2	33.3%
'80s NO.1 HITMAKERS	0	5	0.0%
AFRICAN-AMERICAN WOMEN	2	2	50.0%
1920s NEWS FLASH!	2	2	50.0%
CAMBODIAN HISTORY & CULTURE	1	2	33.3%
HE PLAYED A GUY NAMED JACK...	2	1	66.7%
I'M BURNIN' FOR YOU	2	0	100.0%
NAME THE PARENT COMPANY	2	3	40.0%
GOLDEN GLOBE WINNERS	1	1	50.0%
THE RESIDENTS	2	1	66.7%
NOTES FROM THE CAMPAIGN TRAIL	1	2	33.3%
POETS & POETRY	2	1	66.7%
CAPITAL CITY CHURCHES	0	3	0.0%
THAT 20-AUGHTS SHOW	0	1	0.0%
THE QUOTABLE KEATS	1	1	50.0%
GREEK FOOD & DRINK	2	1	66.7%
RANKS & TITLES	2	0	100.0%
COMPLETE DOM-INATION	0	3	0.0%

Overall, the accuracy across all categories is 50%, indicating an even distribution of correct and incorrect responses. Some categories, like "Potpourri", "Newspapers", "Ranks & Titles" and "I'm Burnin' for You," achieved high accuracy, while others, such as "Broadway Lyrics" and "State of the Art Museum," had low or no accuracy.

Let's break down some category and their answers and analyze possible reasons for the results obtained above, focusing on the reasons why certain questions received incorrect answers.

- NEWSPAPERS
- ✓ The dominant paper in our nation's capital, it's among the top 10 U.S. papers in circulation.
- ✓ Daniel Hertzberg & James B. Stewart of this paper shared a 1988 Pulitzer for their stories about insider trading
- × Early in their careers, Mark Twain & Bret Harte wrote pieces for this California city's Chronicle
- ✓ In 1840 Horace Greeley began publishing "The Log Cabin", a weekly campaign paper in support of this Whig candidate
- ✓ This Georgia paper is known as the AJC for short

A possible reason is that this question requires knowledge about the early careers of Mark Twain and Bret Harte, which may not be available in common sources of information or may be less known, as evident on Wikipedia, where such information is not found.

- BROADWAY LYRICS
- × Song that says, "you make me smile with my heart; your looks are laughable, unphotographable"

Regarding this particular song, neither the Wikipedia page for the song nor the page for its author includes the lyrics. Consequently, the absence of these lyrics renders them irrelevant in the process of identifying the song based on the available information, making it practically impossible to ascertain the correct answer.

- STATE OF THE ART MUSEUM (Alex: We'll give you the museum. You give us the state.)
- × The Naples Museum of Art
- × The Taft Museum of Art
- × The Georgia O'Keeffe Museum
- × The Kalamazoo Institute of Arts
- × The Sun Valley Center for the Arts

Nice! Here, all the answers to the questions were incorrect. There are several factors to consider. Firstly, in the search query, alongside the museum's name the category and additional information in parentheses is included. This causes the supplementary information in the query to be much more extensive compared to the museum's name itself. Additionally, the Naples Museum of Art is now better known by the name The Baker Museum and can only be found on Wikipedia under that title. It's interesting to note, supporting the idea that the search is not directed towards each museum for each question, that in the results obtained for each question, the following results always appear (Clyfford Still, For the Love of God, Red Jacket, John Mellencamp, Yad Vashem). This demonstrates that more emphasis is placed on the rest of the question's text rather than on the specific museum or art institute, leading the search in the wrong direction.

- "TIN" MEN
- ✓ This Italian painter depicted the "Adoration of the Golden Calf"
- ✓ He served in the KGB before becoming president & then prime minister of Russia
- ✓ He earned the "fifth Beatle" nickname by producing all of the Beatles' albums
- ✓ You can't mention this shortstop without mentioning his double-play associates Evers & Chance

The success of the system in the "TIN" MEN category can be attributed to several factors:

- Specificity of Information:

The questions in the "TIN" MEN category seem to be specific and focused on well-known individuals or figures. The specificity of the information makes it easier for the system to identify and retrieve accurate answers.

- Clarity in Question Phrasing:

The questions are formulated in a clear and unambiguous manner. Clarity in question phrasing reduces the chances of misinterpretation, allowing the system to better understand the intent of the question.

- Famous Personalities:

The individuals mentioned in the questions are well-known and extensive information, makes it proficient in answering questions related to them.

- AFRICAN CITIES
- ✓ Several bridges, including El Tahrir, cross the Nile in this capital
- ✓ The name of this largest Moroccan city combines 2 Spanish words
- ✗ This port is the southernmost of South Africa's 3 capitals
- ✗ Wooden 2-story verandas in this Liberian capital are an architectural link to the U.S. south

The incorrect answers result from the provided information being either too general (Wooden 2-story verandas, southernmost) or offering irrelevant additional details that, nonetheless, steer the search in the wrong direction (3 capitals, architectural link to the U.S. south).

- '80s NO.1 HITMAKERS
- ✗ 1988: "Father Figure"
- ✗ 1983: "Beat It"
- ✗ 1988: "Man In The Mirror"
- ✗ 1989: "Miss You Much"
- ✗ 1980: "Rock With You"

Once again, all incorrect. The inaccuracies may be due to the fact that the information in the query leading to the correct answer is very limited compared to the rest of the details. This results in the obtained results pointing to many other artists from that period instead of focusing on a specific one.

- NAME THE PARENT COMPANY
- ✗ Jell-O
- ✓ Milton Bradley games
- ✓ Fisher-Price toys
- ✗ Crest toothpaste
- ✗ Post-it notes

Here, the incorrect answers were generated by those that were more general, but those that included not only the name but also a category had the correct result.

- CAPITAL CITY CHURCHES (Alex: We'll give you the church. You tell us the capital city in which it is located.)
- ✗ In this Finnish city, the Lutheran Cathedral, also known as Tuomiokirkko
- ✗ The High Kirk of St. Giles, where John Knox was minister
- ✗ Matthias Church, or Matyas Templom, where Franz Joseph was crowned in 1867

Besides the fact that the query has ambiguous information, there is a lot of additional information alongside the indicator itself we have the ambiguity in Church Names:

Some church names may be ambiguous or shared by multiple churches in different cities. Might not have the capability to disambiguate between similar church names, leading to incorrect associations with capital cities.

- COMPLETE DOM-INATION(Alex: Not "domination.")
- ✗ This Wisconsin city claims to have built the USA's only granite dome
- ✗ This New Orleans venue reopened Sept. 25, 2006
- ✗ This sacred structure dates from the late 600's A.D.

The system may not correctly interpret the wordplay nature of the category, leading to incorrect answers. If it treats the questions as standard questions about cities, venues, or structures, it might not consider the specific theme related to "dom-ination."

Conclusions

The correct questions can be effectively addressed by a straightforward system due to several key factors. Specifically, questions that exhibit a high degree of specificity and focus on well-known individuals, established facts, or recognized topics stand a greater chance of receiving accurate answers. Additionally, inquiries pertaining to widely acknowledged figures or famous personalities are more likely to yield precise results, benefiting from the extensive information available about these individuals. The clarity and straightforwardness in the formulation of questions play a crucial role, as such attributes minimize the likelihood of misinterpretation. This clarity allows the system to better comprehend the intent of the question, contributing to its proficiency in delivering accurate responses. Notably, in the context of information retrieval and document analysis, where the system is tasked with making connections between documents rather than engaging in complex reasoning, the straightforward nature of the queries facilitates the system's ability to reach concrete answers efficiently.

When it comes to incorrect answers, there are several reasons. The information used in the query is too extensive, making the search challenging, especially when the useful information is minimal and the direction of the search is given by irrelevant informations. Additionally, issues arise when the query has too little information, making the search ambiguous. Problems are also encountered with questions that require an understanding of context to be answered or involve details that cannot be captured at the textual level by the system. Questions with ambiguous information or those containing additional details in parentheses alongside the main query can lead to incorrect results. The supplementary information may overshadow the crucial details needed for accurate answers. Categories with wordplay may pose challenges if the system does not correctly interpret the wordplay nature of the questions. It might treat them as standard queries, resulting in incorrect answers.