Building an Information Retrieval System using University Data – M3

PRI 2023/24

GONÇALO ALMEIDA

ŽAN ŽLENDER

ILINA KIROVSKA

Milestone 1 and 2 recap

- Main idea is to build a search engine where students can get as much information about universities and various aspects related to them.
- The final system configuration from Milestone 2 uses the eDismax query parser, fields: country, Wikipedia_text, city_Wikipedia_text and size with weights 4,3,2 and 1 respectively.
- Final results from Milestone 2 for the boosted system configuration:

Query	P@10	AvP	R@10	F@10
Query 1	0.7	0.8	0.33	0.45
Query 2	1.0	0.85	0.56	0.71
Query 3	0.4	0.49	0.36	0.38
Query 4	0.0	0.18	0.0	0.0
Query 5	0.7	0.6	0.47	0.56

Milestone 3

1. Explored improvements
2. Final approach
3. Application
4. Conclusion

Semantic search

- Improvement that was explored in-depth.
- Goal is to retrieve documents based on the meaning and context of the words and not just do an exact match.
- The input's query and all the documents are encoded into a vector embedding using a sentence transformer.
- Documents are retrieved based on the similarity between their vector embedding and the input's embedding.
- To retrieve the k-nearest documents we used the KNN Query Parser.
- Evaluated using two input types:
  1. Information need
  2. Query

From information need to query

- **Information need 1:** Looking for universities that are top-ranked in computer science and are located in cities that have rich cultural heritage
  - **Query:** top universities in computer science in city with rich cultural heritage
- **Information need 2:** Looking for universities located in the United Kingdom that have courses in biology and are ranked in the top 150
  - **Query:** top universities in united kingdom biology courses
- **Information need 3:** Looking for universities in Germany that have a dental medicine faculty/dentistry and a large number of students
  - **Query:** large universities in Germany dental medicine dentistry
- **Information need 4:** Looking for universities that have a faculty of engineering or a faculty of science and are located in a city with a Mediterranean climate
  - **Query:** faculty of engineering faculty of science in city with a mediterranean climate
- **Information need 5:**Looking for top-ranked universities in the north of Europe with a focus on the Computer Science field
  - **Query:** top universities in North of Europe computer science

Evaluation of the Semantic Search

- **Semantic search using information needs**

IN	P@10	AvP	R@10	F@10
IN 1	0.6	0.66	0.35	0.44
IN 2	0.6	0.67	0.3	0.4
IN 3	0.4	0.39	0.33	0.36
IN 4	0.6	0.67	0.29	0.39
IN 5	0.7	0.73	0.41	0.52

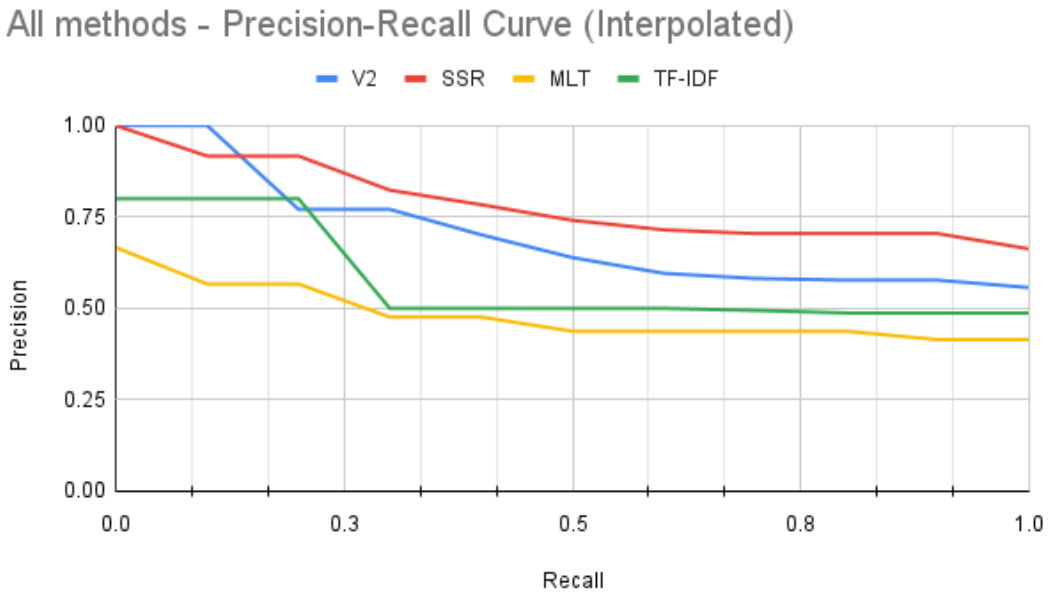
- **Semantic search using queries**

Query	P@10	AvP	R@10	F@10
Query 1	0.1	0.27	0.09	0.09
Query 2	0.6	0.84	0.75	0.67
Query 3	0.4	0.58	0.67	0.5
Query 4	0.6	0.68	0.32	0.41
Query 5	0.6	0.62	0.46	0.52

Metric	SS – IN	SS - Q
MAP	0.62	0.6

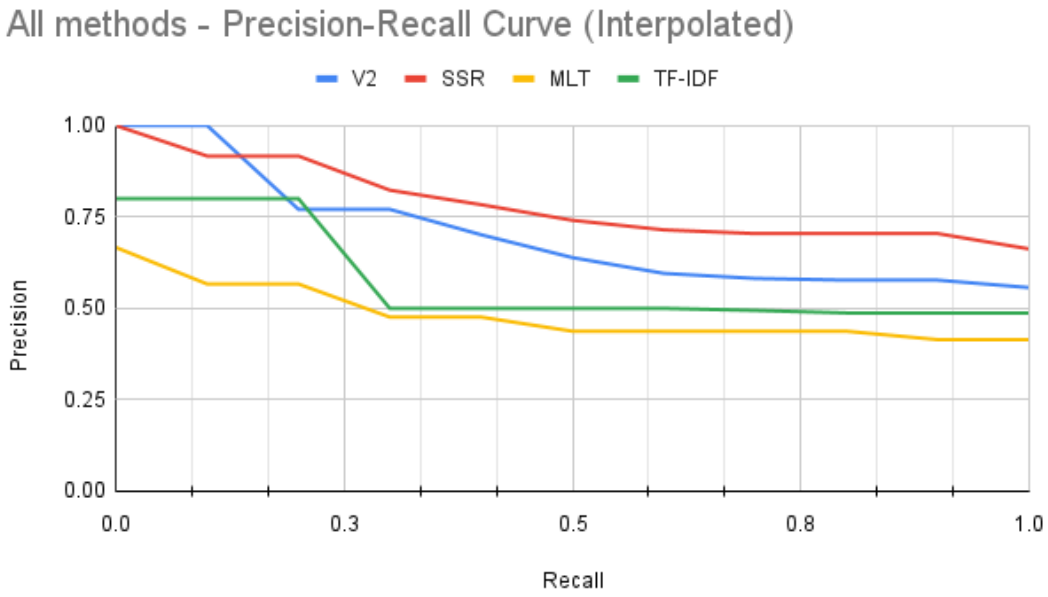
Re-ranking using Semantic search

- Semantic search can also be used fro re-ranking.
- The re-ranking is executed by calculating a score for the document using the KNN algorithm, multiplying it by a multiplicative re-rank weight, and applying it on top of the initial query results.
- Applied to the boosted system configuration from Milestone 2



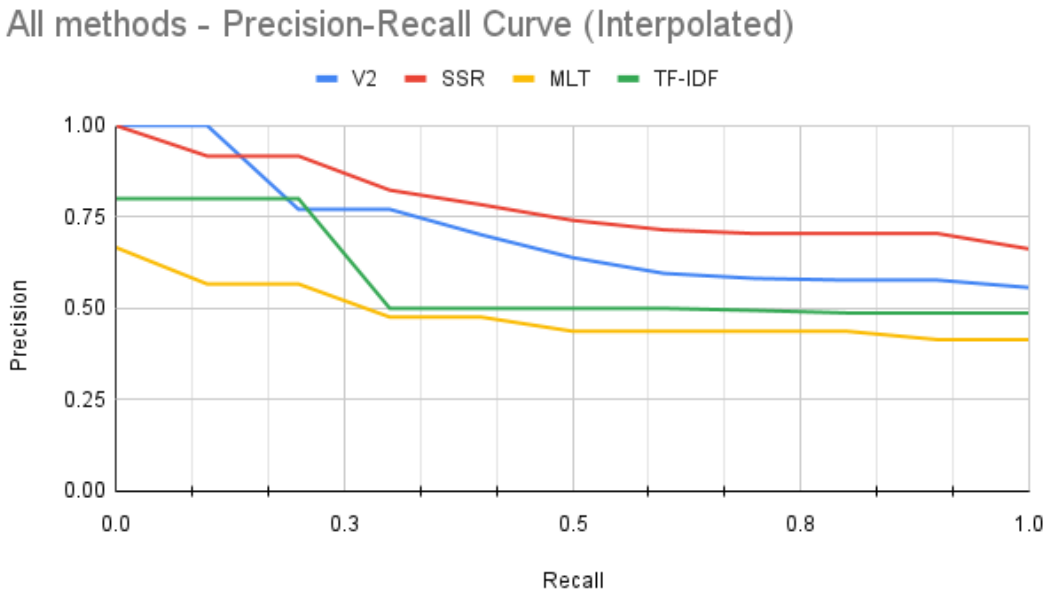
MoreLikeThis - MLT

- A method of enhancing queries in Solr that enables retrieval of documents similar to the referenced documents.
- Built-in function in Solr
- Applied to the boosted system configuration from Milestone 2
- Term frequencies are calculated by the number of times a specific term appears in the text.



Term Frequency-Inverse Document Frequency – TF-IDF

- A statistical measure that evaluates the importance of a term within a collection of documents.
- TF measures how often a term appears in a document.
- IDF measures how common/rare is a term within a collection of documents
- If a term with a high TF-IDF matches a term from our query, then this occurrence increases the ranking score of the document.
- Applied to the boosted system configuration from Milestone 2



(Pseudo) Relevance feedback algorithm

- Relevance feedback algorithm is triggered when the user selects the relevant documents and submits them by pressing a button (Submit Relevance)
  - After that, we utilize the Rocchio Algorithm to incorporate user feedback into the vector space model.
  - In this model, it assigns weights to each term in the inverted matrix (an inverted index) based on its occurrence in relevant or non-relevant documents.
  - Greater weight indicates presumed relevance to be added to the query.

- Pseudo relevance feedback algorithm is triggered automatically when the user search using the search bar and assumes the first 3 results as relevant instead of using the user's feedback.

(Pseudo) Relevance feedback algorithm Step-by-Step Example (Part 1/2)

1. Original Query: The user initiates a search with the original query [universities in Portugal];
2. Result Retrieval: The system retrieves the top 10 results for the query;
3. As part of Relevance Feedback, each result is presented to the user for classification
    1. Doc 1: Relevant
    2. Doc 2: Relevant
    3. Doc 3: Non-Relevant
    4. ...
    5. Doc 10: Non-Relevant
4. Inverted Matrix Construction: The system builds an inverted matrix, featuring terms associated with document IDs and positions of occurrence. Its fed into the Rocchio Algorithm and it computes the Query Vector, associating each term with a weight

(Pseudo) Relevance feedback algorithm Step-by-Step Example (Part 2/2)

i. [university] -> [0.9]

ii. [europe] -> [0.75]

iii. [capital] -> [0.46]

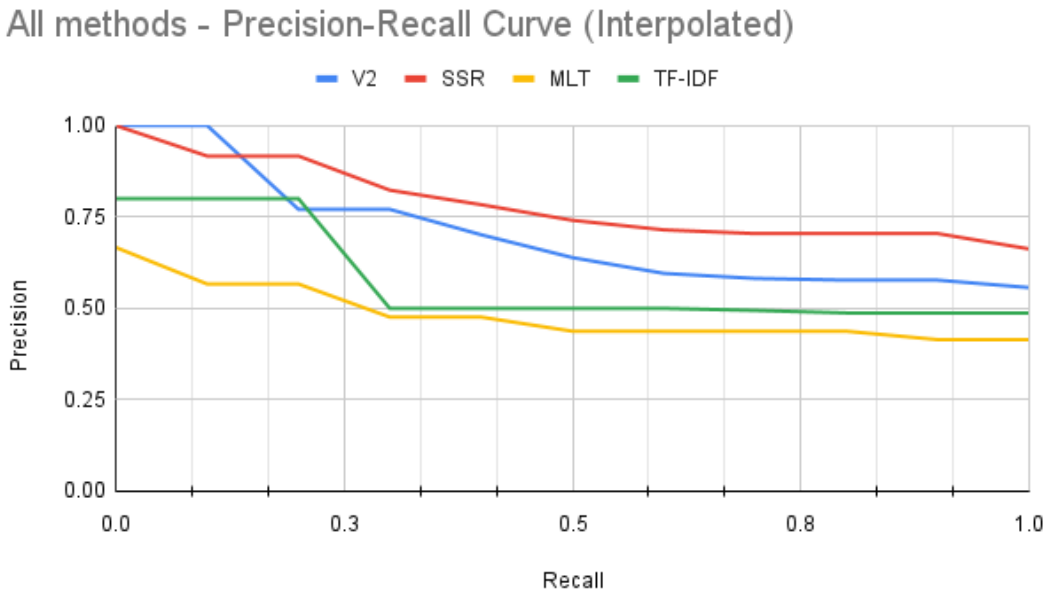
iv. ...

v. [west] -> [0.31]

5. Modified Query: The system selects the top three highest-weighted terms to enhance the original query: [university in portugal europe capital west] is the modified query.

Final approach

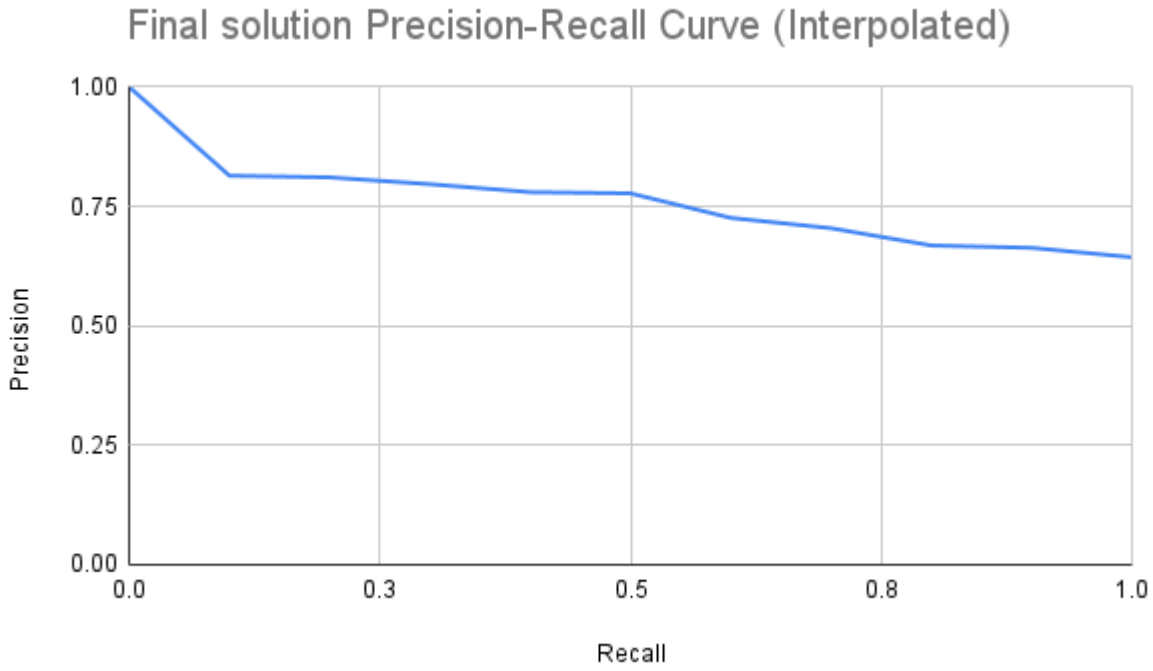
- Re-ranking using semantic search with a relevance feedback algorithm.
- We re-rank the results from the boosted system configuration from M2 using semantic search.
- Afterwards the pseudo/relevance feedback algorithm is used for one final optimization.



Evaluation

IN	P@10	AvP	R@10	F@10
IN 1	0.7	0.75	0.3	0.42
IN 2	0.9	0.89	0.43	0.58
IN 3	0.7	0.64	0.41	0.52
IN 4	0.9	0.82	0.47	0.62
IN 5	0.5	0.56	0.31	0.38

Mean average precision = 0.73



Conclusion

- The combination of using the boosted system configuration from Milestone 2, semantic search and the (pseudo) relevance feedback turned out to successfully improve our final solution for M3.
- MaP of the boosted system configuraton from M2 = 0.59
- Mean average precision of the final approach = 0.73
- We managed to obtain an improvement of 0.14 from one system to another.

Application

- The application consists of three main parts
  1. Solr engine which contains the document collection
  2. NextJS applicaton which serves both as a graphical user interface and a web server
  3. Flask API server that queries the Solr engine API