INFORMATION RETRIEVAL

Luca Manzoni
lmanzoni@units.it

Projects

GENERAL INFORMATION (1)

- You can use any programming language for the implementation (C, C++, Python, Java, Common Lisp, …). Ask if you want to use something unusual.
- Your code should be able to compile/run on Linux/macOS or any other system with a unix-like environment.
- You should provide a way to save and load the entire index from disk, to avoid re-indexing when the program starts.
- Obviously, if you find a library with “make_ir_system()” or something that removes the large majority of the work, ask me if it is OK to use it.

GENERAL INFORMATION (2)

- Comment the code in a reasonable way.
- Keep an eye on performances: if it is faster to read a large collection from disk than to answer a query then there is a problem.
- You will need to prepare a presentation (approx. 25-30 minutes + time for questions) explaining your work.
- Be ready to answer *general* questions about the topics covered in the course (i.e., don't burn your notes after selecting the project).
- Projects should be made by a single person.

BOOLEAN MODEL

PROJECT #1

- Write an IR system able to answer:
- Boolean queries with AND, OR, and NOT.
- Wildcard and phrase queries.
- Some normalisation or stemming can be performed (for stemming you can use a library)
- You can implement spelling correction.
- Evaluate the system on a set of test queries.

VECTOR SPACE MODEL

PROJECT #2

- Write an IR system able to answer, using the vector space model:
- Free-form text queries.
- The system must allow relevance feedback.
- Also the use of pseudo-relevance feedback is possible.
- Evaluate the system on a set of test queries.

PROBABILISTIC IR

PROJECT #3

- Write an IR system able to answer, using a probabilistic model (BIM or BM25):
- Free-form text queries.
- The system must allow relevance feedback.
- Also the use of pseudo-relevance feedback is possible.
- Evaluate the system on a set of test queries.

LOW-LEVEL DETAILS

PROJECT #4

- This project is better written using a lower-level language like C, C++, Rust, etc. since some low level manipulations might be necessary.
- Write a standard Boolean IR system (queries with AND/OR/NOT).
- Implement a compressed representations for the dictionary.
- Compress the posting file. For the size of the gaps use a variable length encoding.
- Compare the size occupied (and, possibly, speed) by a non-compressed representation vs a compressed one.

UPDATE THE DICTIONARY

PROJECT #5

- Write a standard Boolean IR system (queries with AND/OR/NOT) plus the ability to answer phrase queries.
- Allow the user to add and delete documents.
- Addition and deletion should be efficient, so one or mode additional indexes must be used. Recall that it should also be possible to merge the indexes.
- To test it, split the dataset in 3 parts, A, B, and C. Initially the index should contain A and B, then C must be added and B removed (not all at the same time).

WEB CRAWLING

PROJECT #6

- Write a simple web crawler that has that is both fair and robust (i.e., the essential properties that any web crawler should have).
- For robustness only the minimum amount is needed and can be a simple heuristic.
- The crawler should either be multithread (or use async to avoid blocking on requests)…
- …or implement some kind of check for freshness.
- This project can potentially be “split” into two distinct projects.

PAGERANK

PROJECT #7

- On a dataset of webpages to be decided (main problem is of finding a small and significant dataset)
- Implement the PageRank algorithm
- Also implement the ability to get a list of pages and use them to define the jump vector, thus performing topic-specific PageRank.

LATENT SEMANTIC INDEXING

PROJECT #8

- Write an IR system that uses latent semantic indexing to answer queries.
- The system must accept free-form text queries
- Evaluate the system on a set of queries…
- …and try to use different dimensions for the dimensionality reduction.

RECOMMENDER SYSTEM

PROJECT #9

- Given a dataset, containing items and users (what items the user has liked)
- Build a Weighted MF to find the user and item embedding.
- The system must accept as input “a user” (a set of liked documents).
- Return a ranking of documents
  (you can use the fact that you have a vector representation for document to provide the ranking).

SPLITTABLE PROJECTS

SOME PROJECT CAN BE MADE DIFFERENT ENOUGH

- Project #2 (vector space model) can be split into two parts if the method used to perform the similarity computation efficiently is different. (#2.a and #2.b)

- Project #3 (probabilistic IR) can be split into two parts if the probabilistic model employed are different. (#3.a and #3.b)

- Project #4 (low level details) can be split into two parts if the data structures employed by the dictionary are different. (#4.a and #4.b)

- Project #6 (web crawler) can be split into two parts (#5.a and #5.b) if one implementation is multithreaded and one check for freshness.

"THEORY PROJECTS"

WITH LESS CODE

All project without code will require to write a report of reasonable length. Some possible projects include:

- Perform a review of the different similarity measures used in different models (the vector space model in particular), comparing the differences among them and how to compute them efficiently.
- Perform a review of the operations of stemming and spelling correction for languages different from English.
- Provide an introduction to fuzzy logic and fuzzy information retrieval.
- Information retrieval for bioinformatics: what are the main differences from classical IR? Which data structures are used?

ADDITIONAL PROJECTS

MIXED THEORY/PRACTICAL

Additional possible theory (or practical) projects are:

- A review of the different applications of neural networks in information retrieval. This project can also be in the form of a notebook illustrating the application to a toy collection
- A review of different multimedia IR techniques, in particular:
  - Retrieval of images
  - Retrieval of music
  - Retrieval of video