Queen Mary
University of London

Search Engine Design – Group 27

Karen Kiernan, Dingkai Guo, Mengyu Li

8 minute presentation

Summary

What we did

- Labelled sample of 3k records from Reuters21578 dataset with a 1-5 relevance score using ChatGPT
- Compared BM25 vs BM25F ranking performance to ChatGPT gold standard for 10 queries

Tools used

- NLTK to preprocess (remove stopwords, tokenise, lowercase)
- Pyterrier for creating indexer and batch retrieval

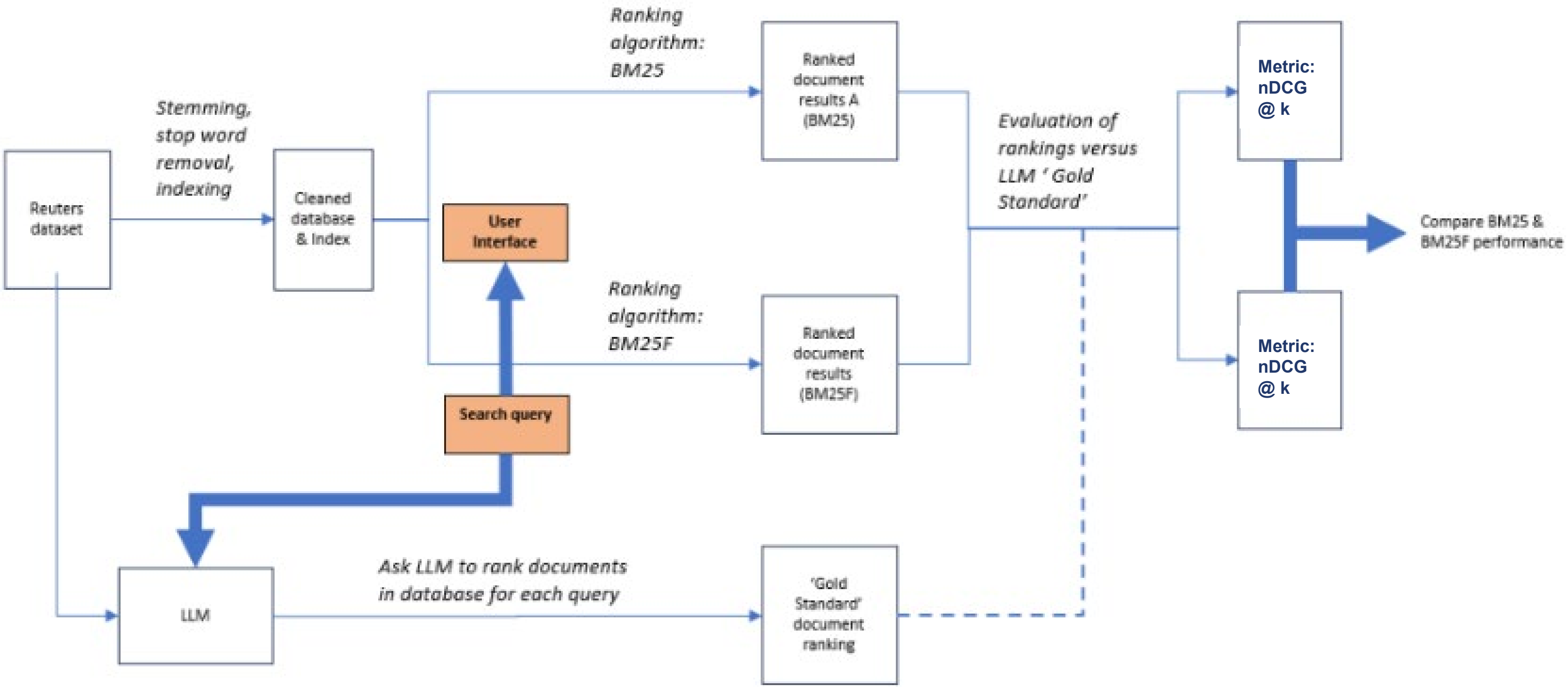
Results

- BM25F outperformed BM25 in relatively few cases
- LLM relevance labelling efficient method to repurpose text classification datasets to IR

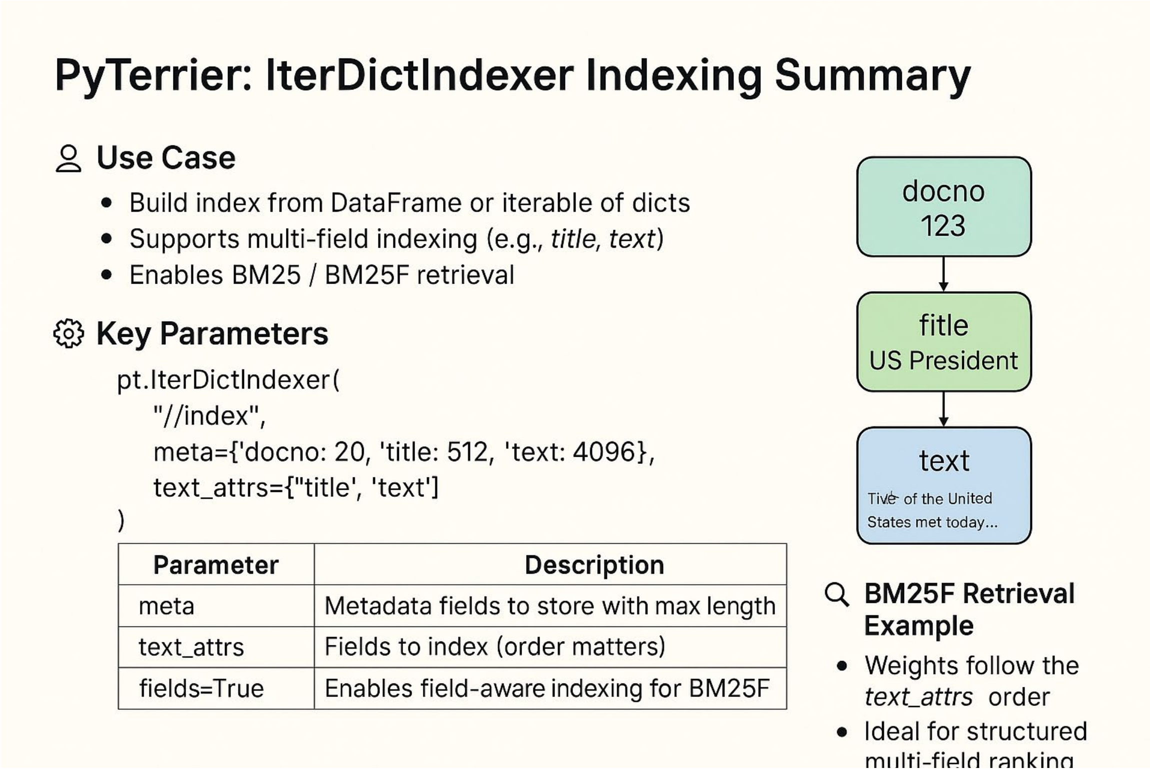
Further work

- Experiment with word embeddings / query reformulations to track language drift in historic datasets (e.g. can ‘US president’ query return Reuters articles from 1987 featuring Donald Trump?)

Pipeline


Reuters dataset
Stemming, stop word removal, indexing
Cleaned database & Index
User Interface
Search query
Ranking algorithm: BM25
Ranking algorithm: BM25F
Ranked document results A (BM25)
Ranked document results (BM25F)
Evaluation of rankings versus LLM ' Gold Standard'
Metric: nDCG @ k
Metric: nDCG @ k
Compare BM25 & BM25F performance
LLM
Ask LLM to rank documents in database for each query
'Gold Standard' document ranking

Indexing and retrieval



PyTerrier: IterDictIndexer Indexing Summary

Use Case

- Build index from DataFrame or iterable of dicts
- Supports multi-field indexing (e.g., *title, text*)
- Enables BM25 / BM25F retrieval

Key Parameters

```
pt.IterDictIndexer(
    "//index",
    meta={'docno: 20, 'title: 512, 'text: 4096},
    text_attrs={"title', 'text']
)
```

Parameter	Description
meta	Metadata fields to store with max length
text_attrs	Fields to index (order matters)
fields=True	Enables field-aware indexing for BM25F

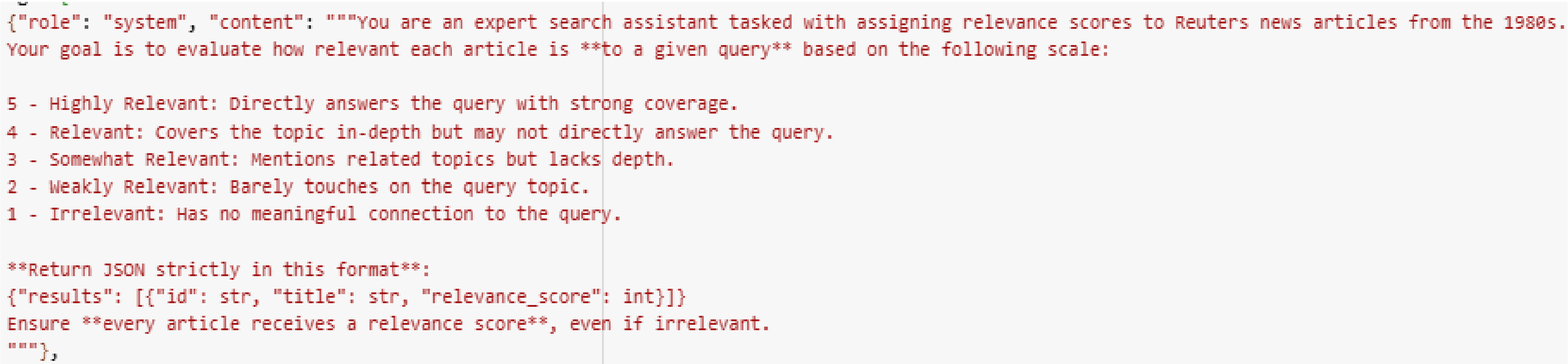
BM25F Retrieval Example

- Weights follow the *text_attrs* order
- Ideal for structured multi-field ranking

Demonstration

LLM relevance labelling

ChatGPT labelled database documents using following prompt:

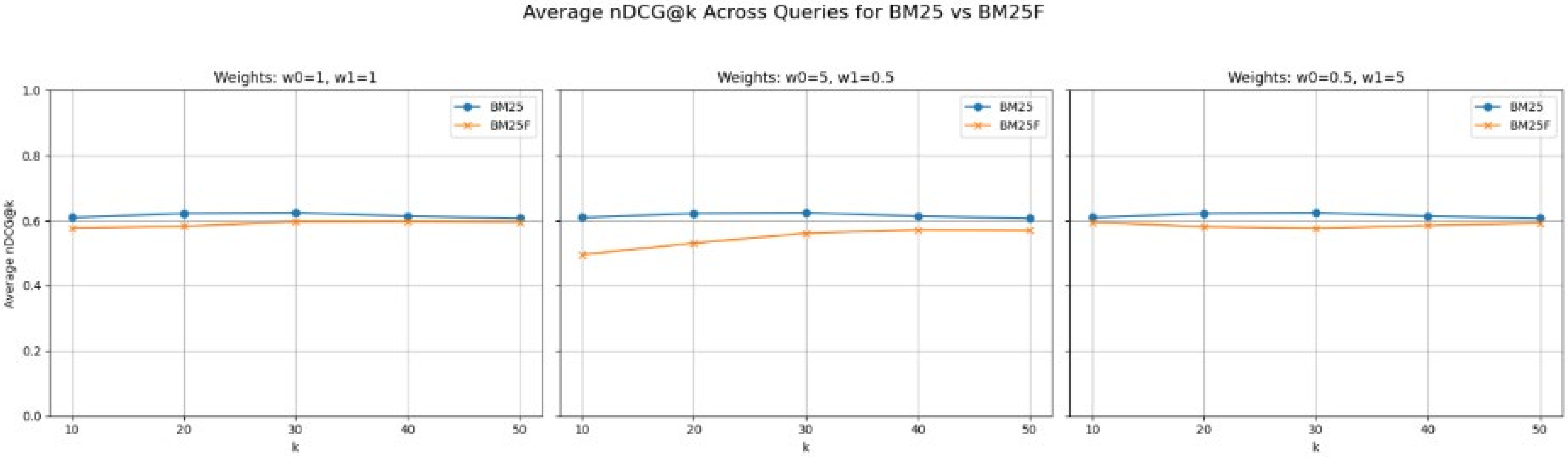


```
{"role": "system", "content": """You are an expert search assistant tasked with assigning relevance scores to Reuters news articles from the 1980s.
Your goal is to evaluate how relevant each article is **to a given query** based on the following scale:

5 - Highly Relevant: Directly answers the query with strong coverage.
4 - Relevant: Covers the topic in-depth but may not directly answer the query.
3 - Somewhat Relevant: Mentions related topics but lacks depth.
2 - Weakly Relevant: Barely touches on the query topic.
1 - Irrelevant: Has no meaningful connection to the query.

**Return JSON strictly in this format**:
{"results": [{"id": str, "title": str, "relevance_score": int}]}
Ensure **every article receives a relevance score**, even if irrelevant.
"""},
```

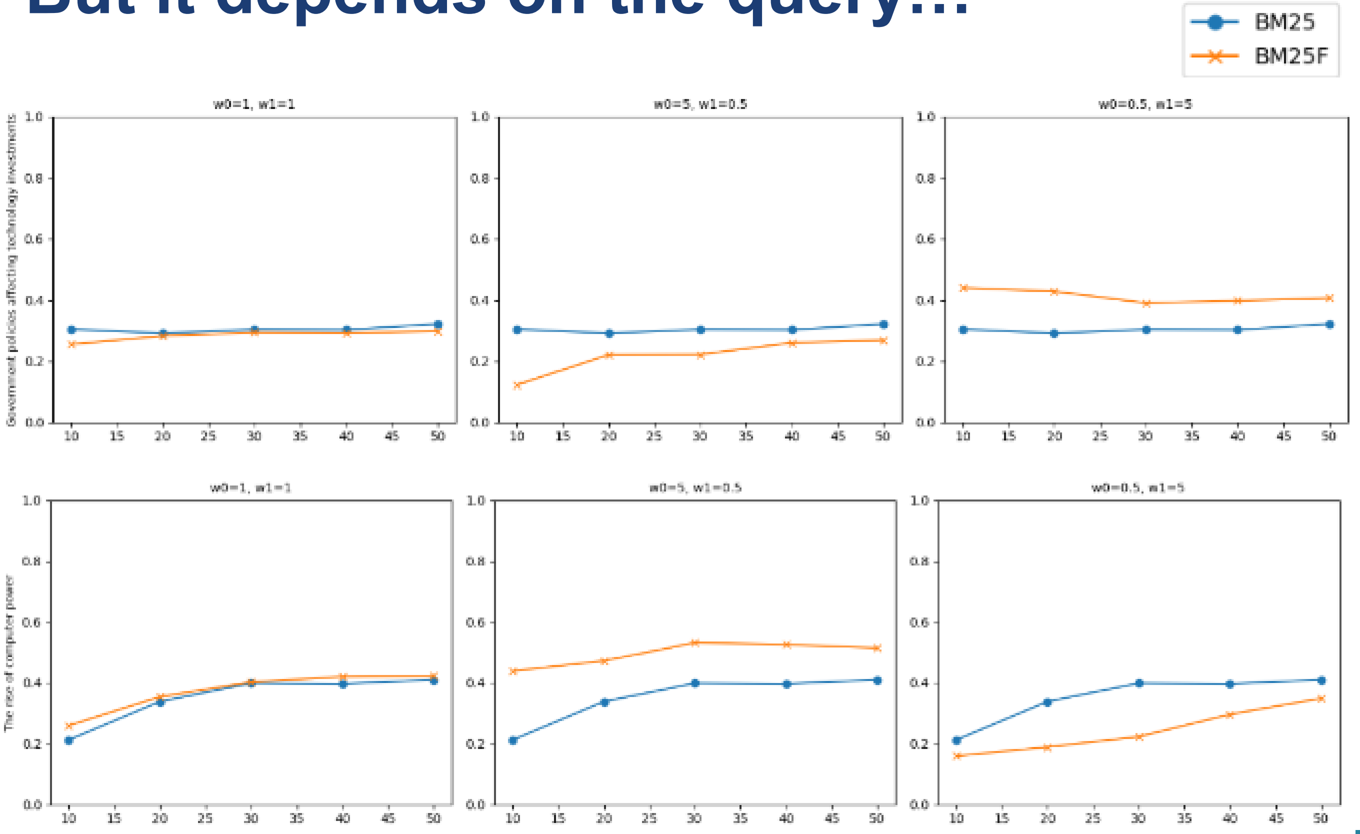
- Initial attempt at 1-100 score caused ChatGPT to provide incomplete results and buffering challenges
- Range of 1-5 informed by *Beyond Yes and No: Improving Zero-Shot LLM Rankers via Scoring Fine-Grained Relevance Labels*, https://arxiv.org/abs/2310.14122

BM25 beats BM25F at all weightings

W0 = article ‘title’
W1 = article ‘text’



But it depends on the query…



'Government policies affecting technology investments'

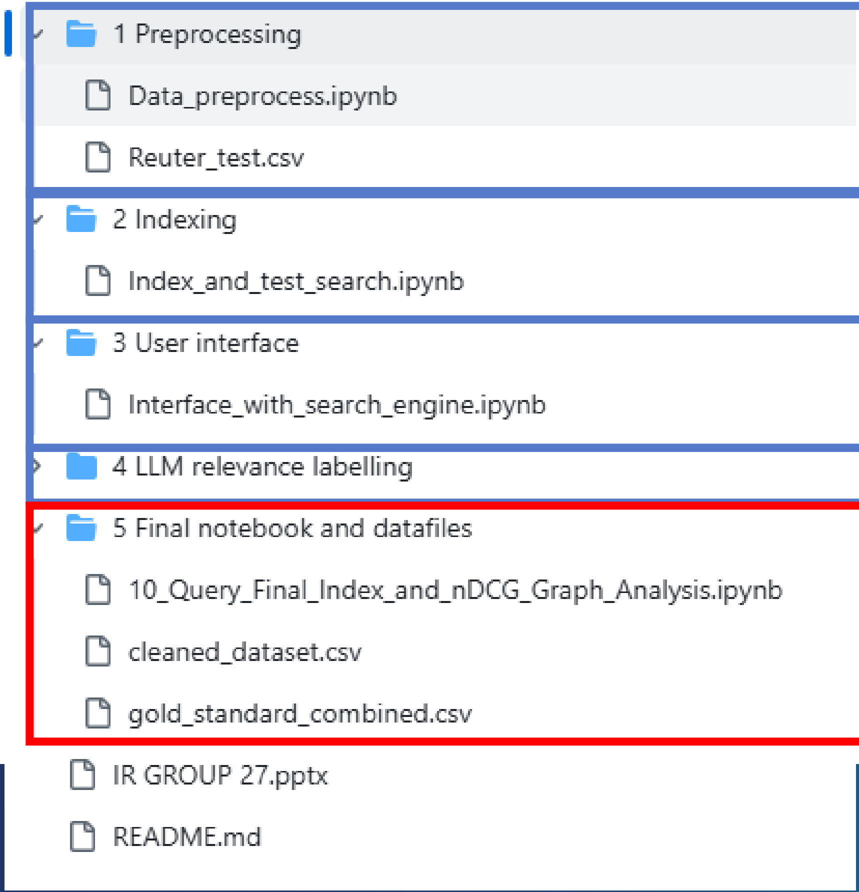
The general theme of this query suits a search weighted to article content. Therefore BM25F performs better when w0 < w1.

'The rise of computer power'

Although this query is also thematic, several relevant articles had 'computer' in the title. Therefore, BM25F outperformed where w0 > w1.

Github repository structure

https://github.com/Bob-623/Search-Engine/tree/main



- 1 Preprocessing
  - Data_preprocess.ipynb
  - Reuter_test.csv
- 2 Indexing
  - Index_and_test_search.ipynb
- 3 User interface
  - Interface_with_search_engine.ipynb
- 4 LLM relevance labelling
- 5 Final notebook and datafiles
  - 10_Query_Final_Index_and_nDCG_Graph_Analysis.ipynb
  - cleaned_dataset.csv
  - gold_standard_combined.csv
- IR GROUP 27.pptx
- README.md

Pre-processing code runs on Reuter_test.csv file

Indexer code (runs on cleaned_dataset.csv)

User interface code (runs on cleaned_dataset.csv)

LLM relevance labelled files per query

Consolidated notebook runs BM25 & BM25F across multiple queries and compares nDCG performance. Runs on:

- *BM25 & BM25F indexer runs on cleaned_dataset.csv*
- *Consolidated relevance scores for all query article pairs*

Thank you

Queen Mary
University of London