Information Retrieval: Text Processing

Aditya Deshmukh (2018A7PS0246P)
Arshit Modi (2018A7PS0191P)
Guntaas Singh (2018A7PS0269P)
Saptarshi Das (2018A4PS0535P)
Siddarth Agrawal (2018A7PS0359P)
GROUP 11

Birla Institute of Technology and Science, Pilani

1 Introduction

The objective of this project is to build an information retrieval system for ranked retrieval of unstructured documents from a corpus consisting of articles from English Wikipedia in partially pre-processed form. Particularly, we use a subset of 89,095 documents from the corpus for the project. The text content of these documents is read and processed to build an inverted index to support fast and efficient retrieval. This index is then used to retrieve documents from the corpus, as per the user information need which he/she can give through single-term or multi-term free-text queries. The vector-space model is used for evaluating these queries, wherein the query as well as documents are mapped to a common vector space. Similarity metrics are used to judge the relevance of each document to the query, on the basis of which the documents are then ranked. The ranking performance of the model is evaluated by manual relevance judgement of the top 10 results for a set of queries. Subsequently, we propose various techniques and heuristics which can be used to improve the results.

2 Part I: Ranked Retrieval

2.1 Index Construction

A necessary prerequisite for index construction is processing the documents into a suitable form. Since, the corpus provided consists of files containing multiple documents each, we first used *Beautiful Soup* (a Python library for pulling data out of HTML and XML files) to extract the document text. Punctuation is removed and all terms are converted to lower-case. Other metadata, such as the document titles and IDs are also collected and stored.

While it may be possible to retrieve documents by linearly scanning the documents of the corpus, such a naive scheme can be prohibitively slow, even for a modestly sized corpus such as ours. An intermediate data structure can significantly improve query processing performance. We, thus, use an **inverted index** in the form of a dictionary mapping each term to a postings list - a list recording the documents containing that term. Since, Python implements dictionaries as hash tables with open-addressing, the index supports constant

time lookup, allowing efficient retrieval. Additionally, we also store the term frequency with every document in the postings list.

Different algorithms can be used for building the inverted index, depending on hardware constraints and corpus size. For our project, the **simple in-memory sort-based indexing** algorithm was found to be sufficiently efficient for building the index in a reasonable amount of time. We iteratively scan every document and record the frequency for every term, appending the entries in the form of a (document ID, term frequency) tuple to the postings lists for each term. Additionally, we also calculate the normalization factors using the *lnc* scheme for document vectors at the time of index construction to optimize query processing.

Finally, we sort the postings list for every term, before writing the inverted index to disk using the Pickle module for Python.

2.2 Query Processing

Query processing involves translation of the query into a form that can effectively and efficiently convey the query to the model. It generally involves applying similar processing techniques that are applied while constructing an inverted index for the model to ensure uniformity between document and query terms.

Our model first converts the query to lowercase and then removes the various punctuations present in the query. Next we tokenize the query so as to form a query vector that will be used to compute the score. We use the *lnc.ltc* scoring scheme while calculating the cosine similarity between the query and a document. Based on the score, the top-K documents (K=10) are returned to the user.

2.3 Evaluation

We proceed to test the performance of our model using a suite of 10 single-term and multi-term queries. An effort was made to choose queries spanning different domains, levels of specificity, and ambiguity. Relevance judgements were made manually by going through the content of the documents and, hence, are subjective. We also report the cosine similarity score for each retrieved document among the top-10 results.

Query	Document Title	Score	Relevance
formula one racing	Motor racing (disambiguation)	0.348	Y
	Grand Prix Legends	0.189	Y
	Manfred Winkelhock	0.174	Y
	Production car racing	0.135	N
	Production Bike Racing	0.132	N
	Matra	0.132	N
	Paul Stoddart	0.125	Y
	Road bicycle	0.12	N
	Ligier	0.118	Y
	Sports Car Club of America	0.112	N

Table 1. Query 1: **formula one racing**

Query	Document Title	Score	Relevance
computer	Krypton	0.308	Y
	Scan	0.270	N
	Graph reduction machine	0.252	Y
	Computer expo	0.239	Y
	Outline of computer science	0.229	Y
	Eric Raymond (disambiguation)	0.227	N
	Event cascade	0.226	N
	KL0	0.216	Y
	List of computer scientists	0.208	Y
	KISS (system)	0.197	Y

Table 2. Query 2: **computer**

Query	Document Title	Score	Relevance
stacks and queues	Priority queue	0.084	Y
	The Twelve Apostles (Victoria)	0.083	N
	Pike (programming language)	0.068	N
	Double-ended queue	0.065	Y
	Split (poker)	0.065	N
	Java Message Service	0.064	Y
	Queueing theory	0.063	N
	Overland Automobile	0.054	N
	Queue (abstract data type)	0.052	Y
	Glossary of patience terms	0.051	N

Table 3. Query 3: **stacks and queues**

Query	Document Title	Score	Relevance
information retrieval	Knowledge Aided Retrieval in Activity Context	0.173	Y
	Data bank	0.149	N
	Protocol-control information	0.123	Y
	User information	0.110	Y
	Telecommunications in Spain	0.108	N
	Identity map pattern	0.105	N
	Destination user	0.099	N
	Information retrieval	0.099	Y
	Document management system	0.096	Y
	Incremental reading	0.089	N

Table 4. Query 4: **information retrieval**

Query	Document Title	Score	Relevance
galaxies in universe	Timeline of knowledge about galaxies, clusters of galaxies, and large-scale structure	0.446	Y
	Galaxy cluster	0.180	Y
	Carl Wilhelm Wirtz	0.154	N
	Supercluster	0.149	Y
	IC 342/Maffei Group	0.138	N
	Galaxy groups and clusters	0.137	Y
	Sculptor Group	0.129	N
	Sandra Faber	0.116	N
	Local Group	0.114	N
	M81 Group	0.112	N

Table 5. Query 5: **galaxies in universe**

Query	Document Title	Score	Relevance
online courses	Online Books Page	0.119	Y
	MIT OpenCourseWare	0.118	Y
	ArtCyclopedia	0.116	N
	SANS Institute	0.104	Y
	Bad Bug Book	0.102	N
	Micro-mainframe link	0.097	N
	Free On-line Dictionary of Computing	0.091	N
	The Wiki Way	0.089	N
	Frank Schmalleger	0.088	N
	1995	0.088	N

Table 6. Query 6: **online courses**

Query	Document Title	Score	Relevance
renewable energy sources	Nuclear energy	0.227	Y
	Norsjö Municipality	0.141	N
	Geothermal power in Iceland	0.129	Y
	Osha Gray Davidson	0.125	N
	Electricity generation	0.108	Y
	Worldwatch Institute	0.104	Y
	Enrico Fermi Award	0.095	Y
	United States Secretary of Energy	0.091	Y
	List of lighthouses in Canada	0.090	N
	Solar furnace	0.089	Y

Table 7. Query 7: **renewable energy sources**

Query	Document Title	Score	Relevance
cash flow	Cash flow	0.198	Y
	Mean Flow	0.130	N
	Net Present Value	0.115	Y
	Discounted cash flow	0.113	Y
	Trade credit	0.100	Y
	Dual access	0.098	N
	Internal rate of return	0.098	N
	Atmospheric waveguide	0.097	N
	Neutral direct-current telegraph system	0.096	N
	Discounting	0.095	Y

Table 8. Query 8: **cash flow**

Query	Document Title	Score	Relevance
president	Rhododendron 'President Roosevelt'	0.232	N
	John Kennedy (disambiguation)	0.223	Y
	President of the Irish Republic	0.215	N
	Calhoun County	0.212	N
	List of facilities named after Lyndon B. Johnson	0.206	N
	President of Iceland	0.172	Y
	List of Presidents of the Swiss Diet	0.163	N
	Washington County	0.154	N
	Jonas Furrer	0.145	N
	Naphtali Daggett	0.143	N

Table 9. Query 9: **president**

Query	Document Title	Score	Relevance
tea coffee beverages	Coffee County	0.190	Y
	Bar (disambiguation)	0.127	Y
	Greasy spoon	0.114	N
	Bopis	0.111	N
	Brunch	0.096	Y
	Black Russian	0.091	Y
	Herbal Tea	0.090	Y
	Teaspoon	0.086	N
	Caffè	0.084	Y
	Sweet tea	0.081	Y

Table 10. Query 10: **tea coffee beverages**

Limitations

- There is no provision for giving suggestions for wrongly spelled queries entered in this system. The system simply gives irrelevant results, rather than suggesting the closest correct spellings or returning documents which contain correct spellings of the query term(or the closest word).
- In case the query contains words that are not present in the corpus, there is no provision of retrieving suggestions of similar words that are there in the corpus.
- There is no implementation of context specific spelling correction and the retrieval system returns absurd results.
- There do not exist any means of weighting the documents with relevance based on title or any zone in the document. The search results are based on a simple search score based on cosine similarity only. Clearly, in the case of Wikipedia documents, the ones with the same query words in the title are more important than other documents.

3 Part II: Improvements and Innovation

3.1 Title Weighing

The structure of HTML documents is easily available through HTML tags. For example, in an HTML document, we can easily tell whether a term appears in the title or not. Intuitively terms that appear in title are more important for retrieval than other terms of document. So by storing the title information of HTML documents in the index and assigning an appropriate importance value to the appearance of the terms in the title, this information can be used to improve the rank assigned to retrieved documents. AltaVista [Alta96], HotBot [HotBot], and Yahoo [Yaho96] are some of such search engines who score a document higher if query words or phrases are found in the title of a web page. Here we calculate the **jaccard similarity** of document title and query and merge it with the cosine score using a suitable parameter.

Query	Document Title	Score	Relevance
radiohead	Paul Lansky	0.073	N
	Eight-bar blues	0.054	N
	Radiohead	0.053	Y
	Clintwood, Virginia	0.043	N
	Glockenspiel	0.042	N
	OK Computer	0.042	Y
	Britpop	0.042	Y
	Mercury Prize	0.040	Y
	Jeff Mills	0.039	N
	Parlophone	0.037	Y

Table 11. Query 1: **radiohead** without title weighing

Query	Document Title	Score	Relevance
radiohead	Radiohead	0.553	Y
	Paul Lansky	0.073	N
	Eight-bar blues	0.054	N
	Clintwood, Virginia	0.043	N
	Glockenspiel	0.042	N
	OK Computer	0.042	Y
	Britpop	0.042	Y
	Mercury Prize	0.040	Y
	Jeff Mills	0.039	N
	Parlophone	0.037	Y

Table 12. Query 1: **radiohead** with title weighing

Query	Document Title	Score	Relevance
teenage mutant ninja turtles	Belgian hip hop	0.145	N
	High Falls, New York	0.115	Y
	Teenage Mutant Ninja Turtles	0.099	Y
	Sean Astin	0.083	Y
	Mae Whitman	0.079	Y
	Corey Feldman	0.77	Y
	Parallax scrolling	0.077	N
	Nickelodeon	0.077	Y
	Eddie Campbell	0.073	N
	Red-eared slider	0.070	N

Table 13. Query 2: **teenage mutant ninja turtles** without title weighing

Query	Document Title	Score	Relevance
teenage mutant ninja turtles	Teenage Mutant Ninja Turtles	0.599	Y
	Ninja	0.186	Y
	Belgian hip hop	0.145	N
	Ninja Tune	0.128	Y
	High Falls, New York	0.115	Y
	Atari Teenage Riot	0.107	Y
	Sean Astin	0.083	Y
	Mae Whitman	0.079	Y
	Corey Feldman	0.77	Y
	Parallax scrolling	0.077	N

Table 14. Query 2: **teenage mutant ninja turtles** with title weighing

Query	Document Title	Score	Relevance
nintendo	List of Nintendo Entertainment System games	0.135	Y
	List of Game Boy Advance games	0.097	Y
	Mario Kart: Super Circuit	0.097	Y
	Mario Tennis	0.091	Y
	Doctor V64	0.091	Y
	Wart (character)	0.083	Y
	Mario Kart 64	0.080	Y
	Game Boy Advance	0.080	Y
	Dr. Mario	0.076	Y
	Wario	0.076	Y

Table 15. Query 3: **nintendo** without title weighing

Query	Document Title	Score	Relevance
Nintendo	Nintendo	0.559	Y
	Nintendo 64	0.312	Y
	List of Nintendo Entertainment System games	0.228	Y
	Super Nintendo Entertainment System	0.183	Y
	List of Game Boy Advance games	0.135	Y
	Mario Kart: Super Circuit	0.097	Y
	Mario Tennis	0.091	Y
	Doctor V64	0.091	Y
	Wart (character)	0.083	Y
	Mario Kart 64	0.080	Y

Table 16. Query 3: **nintendo** with title weighing

3.2 Automatic Query Expansion

Word mismatch is one of the well recognised fundamental problems in information retrieval. It refers to the phenomenon where the users of IR systems often use different words to describe a concept in their queries than those used in the documents. In such a scenario, the IR system ignores query terms not present in its vocabulary, which can negatively affect the ranking performance. This problem is more severe for short casual queries. There exist different techniques which can be used to tackle this problem, one of which is automatic query expansion. Particularly, when the query contains some terms not present in the model vocabulary, the query is automatically expanded using synonyms present in the vocabulary to support the retrieval of more relevant documents. Even in the case where a query term is present in the corpus, adding synonyms can be beneficial and yield improved results if the information need has been specified improperly or ambiguously.

For our project, we used **WordNet** - a lexical database of English which groups words into sets of cognitive synonyms (synsets), each expressing a distinct concept. The Natural Language Toolkit (NLTK) provides a Python interface to conveniently query the WordNet database and find synonyms for a given term, which can then be used to expand the query. However, since the meaning of a word can be heavily context-dependent, we cannot depend on the synonyms as much as the terms originally present in the query, which should naturally be assigned greater importance. Therefore, we assign a weight (adjustable parameter, less than 1) to each synonym added to the query, using which the terms are weighed when calculating the cosine similarity.

Query	Document Title	Score	Relevance
competition	Imperfect competition	0.298	Y
	Snowboarding at the 2002 Winter Olympics	0.227	Y
	Gandalf (theorem prover)	0.180	N
	Luge at the 2002 Winter Olympics	0.175	Y
	Bobsleigh at the 2002 Winter Olympics	0.149	Y
	Wheeler–Lea Act	0.148	N
	Roosdaal	0.140	N
	Interactive Fiction Competition	0.134	Y
	Sockerdricka	0.125	N
	IWAS World Games	0.119	Y

Table 17. Query 1: **competition** without query expansion

Query	Document Title	Score	Relevance
competition	Imperfect competition	0.149	Y
	Snowboarding at the 2002 Winter Olympics	0.114	Y
	Interactive Fiction Competition	0.105	Y
	Gandalf (theorem prover)	0.090	N
	Luge at the 2002 Winter Olympics	0.088	Y
	International Obfuscated C Code Contest	0.082	Y
	Eurovision Song Contest 1956	0.078	Y
	Miss World	0.077	Y
	Bobsleigh at the 2002 Winter Olympics	0.075	Y
	List of Olympic medalists in athletics (men)	0.074	Y

Table 18. Query 1: **competition** with query expansion

Query	Document Title	Score	Relevance
world war	Japanese relocation	0.432	Y
	1918	0.425	Y
	1917	0.425	Y
	1939	0.425	Y
	1944	0.425	Y
	1945	0.425	Y
	1942	0.425	Y
	1943	0.425	Y
	1941	0.413	Y
	1916	0.413	Y

Table 19. Query 2: **world war** without query expansion

Query	Document Title	Score	Relevance
world war	Cold War (disambiguation)	0.128	Y
	Global Diplomacy	0.124	Y
	Japanese relocation	0.109	Y
	1918	0.108	Y
	1917	0.108	Y
	1939	0.108	Y
	1944	0.108	Y
	1945	0.108	Y
	1942	0.108	Y
	1943	0.108	Y

Table 20. Query 2: **world war** with query expansion

3.3 Limiting Cases

In automatic query expansion, we search for synonyms of the query word in the corpus, along with the word itself. As stated earlier, the meaning of a word can be heavily context-dependent and thus this heuristic can retrieve irrelevant documents. Although we tried to reduce this shortcoming by adjusting the weight of the synonyms, it still does not eliminate this irregularity completely. For example, when we query "Hello World", we expect results related to programming. However, synonyms of the query term "World", like "Global", cause some irrelevant documents (like "Global Diplomacy") to appear in the top results.

We have used the jaccard coefficient for the title weighing heuristic, which can be ineffective if the query contains very frequent terms appearing in the corpus such as 'the', 'of', 'in' which do not convey any meaning of their own. For example, when we query "Murder on the Orient Express", documents like "On the Waterfront" and "Watch on the Rhine" are returned.

4 Future Scope

For free text queries, we can also weigh in proximity between terms. This can improve the performance of the model significantly in case of queries involving frequent bi-words like "red wine". However, this will produce an additional overhead while building the index.

We can also improve the performance of the model by implementing something similar to zone weighing. Although in our case documents do not have explicitly defined zones. We can employ some heuristic considering the first 100 lines as introduction, the next 500 lines as body and so on.

We can also add a spelling correction feature (which could be based on the Levenshtein distance or some other heuristic) to improve model performance and provide more flexibility to the user.