Information Retrieval (CS F469)

Assignment report

on

TEXT PROCESSING

BY

GROUP 13

Srinath Swaminathan	**2018A7PS0204P**
Peddi Vineel	**2018A7PS0241P**
Aman Kumar	**2018A8PS0764P**
Archit Golatkar	**2018A4PS0572P**
Aurindom Bhattacharyya	**2018A2PS0082P**

APRIL, 2021

Introduction

Information Retrieval is the process of finding material, that satisfies a user's need, from a large collection of data. This data is usually present as an unstructured sequence of documents, also known as a corpus.

The form of information retrieval used in this assignment is ranked retrieval, where the documents are not only retrieved, but ranked in the order of their relevance to the user. For implementing ranked retrieval, we use the vector space model as the building block, and add additional functionalities for better performance.

The information need is represented in the form of free-text queries. A free text query is simply one or more words, terms, numbers, and optionally operators. This query is given as input to the information retrieval model, which then outputs the relevant documents corresponding to the query.

Corpus

English Wikipedia in partially processed form and divided into approximately 1500 files was made available to us at

https://drive.google.com/drive/folders/1ZsnuEm7_N6aUwhjFpv-TZXFt4DiYex4t?usp=sharing

Each file consists of multiple documents. For this assignment, we have used 14 files consisting of a total of 6727 documents. We have indicated the names and ids of the documents used in the "document_list.txt" file.

Part 1: Vector Space Model

The vector space model considers each document and each query as a vector, whose dimensions equal the number of distinct terms present in the corpus. In other words, each document vector has a value representation for each term in the corpus. These values are determined by a specific SMART notation. The notation used throughout the project is **lnc.ltc**, which specifies that the document vectors are built using the log of term frequency (tf) values, while the query vectors additionally weight the inverse document frequency (idf) with the term weight. In essence, if $w_{t,d}$ is the term weight of term t in vector d, $tf_{t,d}$ is the term frequency of t in d, df_t is the document frequency of t in the corpus, and idf_t is the inverse document frequency of t, then

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

If d corresponds to a query,

$$\mathrm{w}_{t,d} = (1 + \log_{10}\mathrm{tf}_{t,d}) \times \mathrm{idf}_t$$

and if d corresponds to a document, we don't weight with idf

$$w_{t,d} = (1 + \log_{10} tf_{t,d})$$

Note: If tf is 0, $w_{t,d}$ is also 0.

The c's in **lnc.ltc** indicate the use of cosine similarities for scoring the documents. The cosine similarity between any two vectors A and B is computed as

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|}$$

The top K documents can be then retrieved in descending order based on their scores with respect to the query.

Below, we provide the steps to construct a simple vector space model in python based on the given corpus.

Index Construction

The raw corpus contains files present in HTML format, so it is first essential to extract the actual contents of the documents. Other essential information such as document ID and document name is required at various stages of the retrieval mechanism, and therefore must also be extracted. This is made possible by a Python library called Beautiful Soup, used for easy navigation and parsing of HTML and XML files. Using Beautiful Soup, the contents of each document are extracted, along with the name of the document and its corresponding ID.

It is never advisable to leave the document contents in its raw form while processing the queries. An intermediate structure is necessary to efficiently construct the vectors and compute the scores. An Inverted Index is such a structure, which not only facilitates fast and efficient query evaluation, but is also very easy to construct. The Inverted Index maps each term to the list of documents in which the term appears. Additional values such as term frequency and document frequency are also required for evaluating the queries, but they make the structure slightly more complex. The index constructed for this assignment has the following structure.

{$term_1$: [df_1, {doc_1: $tf_{1,1}$, doc_2: $tf_{1,2}$, . . .}],
$term_2$: [df_2, {doc_1: $tf_{2,1}$, doc_2: $tf_{2,2}$, . . .}],
.
.
.
}

Here, $\mathbf{df_i}$ is the document frequency of $\mathbf{term_i}$, and $\mathbf{tf_{i,j}}$ is the term frequency of $\mathbf{term_i}$ for $\mathbf{doc_j}$.

The Inverted Index, for this assignment, is constructed using a simple in-memory based construction algorithm. For each document, the following procedure has been followed.

- A temporary index is created, and the contents of the document are fed line-by-line to the word processor to output the final set of terms which are ready to be stored in the index. The details of the word processing step are mentioned in the next section. For each term in the document, the temporary index maps the term to its corresponding term frequency in that particular document.
- The temporary index is then merged into the main Inverted Index, and the document frequencies are updated accordingly. Since the documents are processed one at a time, the document normalization factor is computed at this step itself to save time during query processing. This is stored in the document info dictionary, which maps the document ID to its corresponding name and document normalization factor.

Finally, both the inverted index and the document info dictionaries are written to the disk, so that the constructed indices can be used repeatedly by other files instead of having to process the corpus each time.

Word Processing

Word processing is an essential step for any retrieval system, especially those based on free text queries. Multiple techniques, such as stemming, lemmatization and normalization can be used to get more relevant documents for the query. For this assignment, the focus lies solely on removing unnecessary punctuations, so that appropriate matches can be obtained while processing free text queries. Each document in the corpus is processed line-by-line, and the punctuations were removed in two stages.

- Any word containing a hyphen or an underscore is split into two words. For example, **well-being** is split into **well** and **being**.
- All other punctuations present anywhere are simply removed. For example, **U.S.A** is changed to **USA**.

Finally, each word is converted to lower case before proceeding to the remainder of the Index construction.

Query Processing

The query, taken as free-text input from the user, is first pre-processed using the same word processing mechanism used for the index construction. The inverted index and doc info dictionaries are read from the disk, to avoid processing the corpus repeatedly.

A query vector is constructed, where each term in the vector is weighted using the tf-idf scheme. The idf value is obtained from the df value of the term in the inverted index. The query normalization factor is computed from the query vector itself, while the document normalization factor is extracted from the doc info dictionary.

To increase efficiency, a form of index elimination is used. The cosine similarity of the query with each document is computed only for the terms present in the query, instead of constructing a document vector for every document each time a query is processed.

The scores obtained are stored in a dictionary constructed to map document ID to score, and the top K scores are extracted using a min-heap, which is slightly more efficient than sorting the scores.

Model Evaluation

The vector space model was tested on 10 multi-term queries. The queries were chosen to cover many diverse cases, including proper nouns, rare terms, stop words, etc. The top 10 documents obtained for these queries are given below, along with their scores based on the lnc.ltc scheme, and their relevance to the query. This relevance was assigned manually based on our interpretation of the query, so it is highly subjective, and has a tendency to vary from person to person.

Results

Query	Document Title	Score	Relevance
superhero movie	The Wizard of Speed and Time	0.08083	Y
	Golden Heroes	0.07865	Y
	GURPS Supers	0.07821	N
	Unbreakable (film)	0.06528	Y
	Nerd	0.05018	N
	Spider-Man	0.04631	Y
	The Amazing Spider-Man	0.04489	Y
	Clark Kent	0.04399	N
	Fantasy Film	0.04395	N
	Jack Kirby	0.04367	N

Table 1: Query 1 - superhero movie

Query	Document Title	Score	Relevance
acids and bases	Johannes Nicolaus Brønsted	0.10000	Y
	Aliphatic compound	0.09261	N
	Acid	0.09163	Y
	Nucleic acid	0.08717	Y
	Amide	0.08332	Y
	Oxide	0.08243	Y
	Genetic code	0.08081	N
	Nucleotide	0.07927	Y
	Citric acid cycle	0.07176	Y
	Guanine	0.07099	Y

Table 2: Query 2 - acids and bases

Query	Document Title	Score	Relevance
sport championship	World cup competition	0.12936	Y
	List of gymnasts	0.12784	N
	Korfball	0.09353	N
	Six Nations Championship	0.08004	Y
	Grappling	0.07967	N
	America's National Game	0.07864	N
	Netball	0.07738	N
	Floorball	0.07718	N
	UEFA	0.07444	Y
	Auto racing	0.07357	N

Table 3: Query 3 - sport championship

Query	Document Title	Score	Relevance
dance forms	Vintage dance	0.24007	Y
	Nightclub two step	0.12700	Y
	Connection (dance)	0.12580	Y
	Galliard	0.12415	Y
	Folk dance	0.11428	Y
	Dance	0.10329	N
	Waltz	0.09830	Y
	Flamenco	0.08235	Y
	Performing arts	0.07466	N
	Gavoi	0.07150	N

Table 4: Query 4 - dance forms

Query	Document Title	Score	Relevance
space shuttles	Shuttlecock	0.08452	N
	Space Shuttle Challenger	0.07464	Y
	Transport in Guatemala	0.06901	N
	Space Shuttle Endeavour	0.06735	Y
	Space Shuttle Discovery	0.06499	Y
	Outline of space science	0.05963	N
	Space Shuttle Columbia	0.05653	Y
	Space Shuttle Atlantis	0.05587	Y
	Space Shuttle	0.04849	Y
	Kennedy Space Center	0.04747	N

Table 5: Query 5 - space shuttles

Query	Document Title	Score	Relevance
operating system	Kent Applicative Operating System	0.20536	Y
	Off topic	0.13619	N
	OpenGL Utility Toolkit	0.11556	N
	POSIX	0.11439	Y
	UCSD Pascal	0.11144	N
	Operating system advocacy	0.10744	Y
	Virtual machine	0.10632	Y
	VSE (operating system)	0.10176	Y
	Context switch	0.10107	Y
	KA9Q	0.10068	N

Table 6: Query 6 - operating system

Query	Document Title	Score	Relevance
baseball teams	Colorado Rockies	0.09676	Y
	Cy Young Award	0.09518	N
	Albert Spalding	0.09473	N
	America's National Game	0.09463	N
	Miami Marlins	0.08842	Y
	Arizona Diamondbacks	0.08682	Y
	San Diego Padres	0.08122	Y
	Oakland Athletics	0.07144	Y
	Cy Young	0.07027	N
	Seattle Mariners	0.06899	Y

Table 7: Query 7 - baseball teams

Query	Document Title	Score	Relevance
fighter aircrafts	List of NATO reporting names for fighter aircraft	0.13389	Y
	NATO reporting name	0.07461	N
	McDonnell FH Phantom	0.05924	Y
	Fly-by-wire	0.05873	N
	Supermarine	0.05671	Y
	Knockout	0.05348	N
	Lockheed F-117 Nighthawk	0.05310	Y
	General Dynamics F-16 Fighting Falcon	0.04973	Y
	Grumman F-14 Tomcat	0.04950	Y
	Virtua Fighter (video game)	0.04559	N

Table 8: fighter aircrafts

Query	Document Title	Score	Relevance
Godzilla movies	Kenji Sahara	0.13626	N
	Terror of Mechagodzilla	0.10201	Y
	Godzilla vs. King Ghidorah	0.09422	Y
	Son of Godzilla	0.08895	Y
	Godzilla vs. Mothra	0.08869	Y
	Ebirah, Horror of the Deep	0.08622	Y
	The Return of Godzilla	0.08219	Y
	Godzilla	0.08032	Y
	Godzilla vs. Biollante	0.07947	Y
	King Kong vs. Godzilla	0.06986	Y

Table 9: Query 9 - Godzilla movies

Query	Document Title	Score	Relevance
alkali metal	Carbide	0.09055	N
	Abated	0.07000	N
	Oxide	0.06460	N
	Alkali metal	0.06276	Y
	Amide	0.05862	N
	Potassium	0.05738	Y
	Lutetium	0.05537	N
	Lithium	0.05526	Y
	Gallium	0.05462	Y
	Coordination complex	0.05403	N

Table 10: Query 10 - alkali metal

Analysing Results

Clearly, this model is not very accurate, and certainly not an option for large scale search engines. Some queries display very few relevant results, and even in cases where more relevant documents are retrieved, they do not conform to any logical order and seem to be scattered around the irrelevant results.

The major drawback for this model is the lack of contextual meaning for a word. This model simply tries to match the query terms with the document terms, without trying to determine what those sequence of words could possibly indicate. This idea forms the backbone for the two improvements proposed in this report.

It must be said that perfect context determination is very difficult. Languages are complex in itself, and there is a good chance that the same sequence of words could mean something entirely different. Sometimes, the user may frame the query quite poorly, and it would be unfair to judge the performance of a retrieval system for these cases. So rather than trying to obtain a perfect contextual representation for every query, the focus will be to improve the general performance of the model.

Part 2: Zoned Indexing and Champion List

As noted earlier, the basic vector space model is incapable of recognizing and evaluating based on contextual information. The zone indexed model deals with a few specific problems in this vast maze of context representation.

The zone indexed model considers a document as a collection of zones. Most documents are not simply a sequence of terms, they have some additional information (metadata) associated with them. If the document is a book, the metadata could be the date of creation, the language used, date of publication, and so on. In a way, they represent the whole document, and so certainly convey a lot more information than the average term in the document.

The above metadata are called fields. A zone is very similar to a field, except that a zone can contain arbitrary free text, whereas a field usually takes limited set of values. Examples of zones are the title of the document, the abstract, the main body, etc. As mentioned earlier, terms present in certain zones like the title tend to convey more information than terms in the body, and it could be useful to give these terms a little boost. Obviously, this does not take care of the entire contextual meaning problem, but it can greatly improve the performance on a few types of queries. For this assignment, we only use two zones, the title, and the body.

Significant improvements can be observed in queries where the user wants to retrieve specific documents by name. For example, if the user enters a movie name, it is very likely that he wants to retrieve documents based solely on that movie, not on the actors who were in the movie. For example, if we enter “Godzilla”, we expect to see documents corresponding to movies from the Godzilla franchise, not actors. Similarly, if we enter “University of California”, we would expect to see documents about the University, not about professors in the University or famous scientists who graduated from the University. These two queries are included in the results section of this model.

Additionally, we have also modified the posting lists of the main inverted index to Champion Lists. Simply put, a champion list is a trimmed form of the posting list. The posting lists are arranged in a certain order, so as to keep the less relevant documents at the end of the list. This list is then chopped off to remove the documents at the end. Champion lists are very useful in improving the latency of retrieval without heavily compromising on the relevance of documents retrieved. Relatively simple heuristics can be used for sorting the posting list as time is the main concern here.

Implementation

There are multiple ways to implement zoned indexing. For simplicity, we have used the already built inverted index for the body, and constructed a separate index for the title. Both the indices follow the same structure as shown in the previous model.

The Champion Lists are introduced only for the index corresponding to the body. The posting lists are arranged in decreasing order of term frequency, and only the top 100 documents are considered to form the Champion List.

The document info dictionary is updated to include the document normalization factor for both inverted indices. The normalization factors are computed in the same way as done in the basic vector space model.

The scores are computed for both indices with respect to the query. The title zone is given a weight of α, and the final score is computed for each document as follows.

$$doc\_score = \alpha \times doc\_title\_score + (1 - \alpha) \times doc\_body\_score$$

doc_title_score and doc_body_score are the scores of the document corresponding to the title and body zones respectively. α is fixed for this implementation and is equal to 0.1.

These scores are then used for retrieval.

Model Evaluation

The performance improvement of the zone indexed model over the basic vector space model is shown using 3 queries. The top 10 documents retrieved for each query is displayed, along with their scores and relevance. The relevance is again assigned manually.

Results

Query	Document Title	Score	Relevance
programming language	KL0	0.31422	Y
	List of object-oriented programming languages	0.26991	Y
	KISS (system)	0.25597	N
	KRL (programming language)	0.15326	Y
	Kent Recursive Calculator	0.14911	Y
	Klerer-May System	0.12842	Y
	Operator overloading	0.12758	N
	List of Spanish-language poets	0.11639	N
	Word (disambiguation)	0.11451	N
	Niklaus Wirth	0.11341	N

Table 11: Query 1 - programming language - for basic model

Query	Document Title	Score	Relevance
programming language	Programming language	0.17654	Y
	Literate programming	0.15686	N
	Ada (programming language)	0.15112	Y
	Occam (programming language)	0.14673	Y
	Scheme (programming language)	0.14183	Y
	CLU (programming language)	0.14024	Y
	Lisp (programming language)	0.13847	Y
	Cyclone (programming language)	0.13709	Y
	APL (programming language)	0.13645	Y
	Fourth-generation programming language	0.13308	Y

Table 12: Query 1 - programming language - for zone-indexed model

Query	Document Title	Score	Relevance
University of California	Grigory Barenblatt	0.18558	N
	Nanoengineering	0.13874	N
	James Blaylock	0.13408	N
	William Alston	0.11173	N
	Kernite	0.10833	N
	Niklaus Wirth	0.10766	N
	Paul Vixie	0.10687	N
	Steve Crocker	0.10344	N
	Allan Dwan	0.10270	N
	Kim Stanley Robinson	0.09949	N

Table 13: Query 2 - University of California - for basic model

Query	Document Title	Score	Relevance
University of California	University of California	0.16191	Y
	University of California, San Francisco	0.15320	Y
	University of California, Davis	0.13906	Y
	University of California, Berkeley	0.13161	Y
	University of California, Santa Cruz	0.13109	Y
	University of Southern California	0.13107	N
	California Department of Transportation	0.12842	N
	University of California, San Diego	0.12749	Y
	University of Bergen	0.11159	N
	University of Gothenburg	0.10638	N

Table 14: Query 2 - University of California - for zone-indexed model

Query	Document Title	Score	Relevance
New Zealand	Military of Samoa	0.19490	N
	Geography of the Cook Islands	0.19293	N
	Telecommunications in New Zealand	0.18781	Y
	Economy of the Cook Islands	0.12590	N
	Politics of Niue	0.12052	N
	Economy of New Caledonia	0.10815	N
	Vincent Ward (director)	0.10543	N
	Arapaoa Island	0.09672	Y
	Geography of Samoa	0.09467	N
	Politics of the Cook Islands	0.09402	N

Table 15: Query 3 - New Zealand - for basic model

Query	Document Title	Score	Relevance
New Zealand	Telecommunications in New Zealand	0.21414	Y
	New Zealand English	0.15441	Y
	Politics of New Zealand	0.14294	Y
	New Zealand Defence Force	0.13918	Y
	Geography of New Zealand	0.13887	Y
	Demographics of New Zealand	0.13789	Y
	Transport in New Zealand	0.13300	Y
	Foreign relations of New Zealand	0.12968	Y
	Economy of New Zealand	0.12513	Y
	Economy of New Caledonia	0.10796	N

Table 16: Query 3 - New Zealand - for zone-indexed model

Analysing Results

The above 3 queries deal with a particular case as mentioned earlier - The user knows what he wants to a very specific level. For this case, we observe a significant improvement in performance using zoned indexing. Clearly, the extra boost given to the terms present in the title has propelled more relevant documents to the top.

However, the obvious disadvantage to this model is when the user enters a more generic query. Consider the query “Car brand”. The user is most likely not interested in documents which have “Car” or “brand” in the title, rather, he expects documents corresponding to different car brands to be retrieved. A boost to the title actually produces an adverse effect in this case.

We demonstrate the results of both models with respect to “Car brand” below.

Query	Document Title	Score	Relevance
Car brand	Cadillac (disambiguation)	0.37396	Y
	Volvo	0.08146	Y
	International Formula 3000	0.07742	N
	Lamborghini	0.07120	Y
	Volkswagen Group	0.06788	Y
	Škoda Auto	0.06772	Y
	Koenigsegg	0.06661	Y
	Knowledge Aided Retrieval in Activity Context	0.06605	N
	Ferrari	0.06505	Y
	White Russian (cocktail)	0.06504	N

Table 17: Query - Car brand - for basic model

Query	Document Title	Score	Relevance
Car brand	Open-wheel car	0.10922	N
	Car Talk	0.10794	N
	Cable car (railway)	0.10634	N
	Armored car (military)	0.09694	N
	Stock car racing	0.09015	N
	Sports Car Club of America	0.08243	N
	International Formula 3000	0.06968	N
	Lamborghini	0.06408	Y
	Volkswagen Group	0.06109	Y
	Škoda Auto	0.06095	Y

Table 18: Query - Car brand - for zone-indexed model

A significant dip in performance is observed for the query in the zone indexed model. We need a different approach to handle more generic queries.

Part 3: Query Expansion

Query expansion involves evaluating a user's input and expanding the search query to match additional documents. The basic idea behind query expansion is to add a certain number of synonyms for each query term, so that a few of them may help capture the intended context. This can also be extended to query terms not present in the corpus, but we have limited ourselves to the corpus vocabulary for this assignment. The tf for the extended query term can be computed as follows.

$$tf_{extended} = tf_{original} \times similarity(original, extended)$$

Note: Both extended and original words are restricted to the corpus vocabulary.

The similarity value is discussed under the implementation section. The value is chosen to lie between 0 and 1.

Weighting the extended query terms with the similarity measure ensures that while these terms are considered, they are not given the same importance as the original term. Intuitively, a match with the original term is regarded as better. Of course, if multiple original terms produce the same new term, then its overall importance may increase.

Implementation

The two main tasks during query expansion are finding synonyms for the original term, and computing their similarities with the term. Both can be easily achieved by using pretrained GloVe embeddings. GloVe stands for Global Vectors for Word Representation. It is an unsupervised learning algorithm which constructs a vector for each term. The vectors are constructed so that the cosine similarity of the vectors of two terms indicate the semantic similarity of the terms.

The embeddings file used for this assignment is **glove.6B.100d.txt** [1], which is the result of training the model on the Wikipedia 2014 dump and the English Gigaword 5 corpus. As the name suggests, it contains 6 Billion tokens each having a 100-dimension vector.

To process these embeddings, we use the KeyedVectors model in Gensim, a python library for dealing with vector embeddings. Before using the embeddings, we remove the tokens which are not present in the corpus vocabulary. This does reduce the contextual information present to a certain degree, but ensures that the new query terms produced would be in the corpus vocabulary. The token reduction also does not change the similarity values, as the vectors are precomputed. The embeddings are then loaded into a KeyedVectors model, and extract the top n similar terms, along with their cosine similarity values.

The new terms and their tfs are appended to the query, and the query is processed using the basic vector space model.

Model Evaluation

The results of the basic vector space model with and without query expansion are shown for 3 queries. The top 10 documents retrieved for each query is displayed, along with their scores and relevance. The relevance is again assigned manually.

Results

Query	Document Title	Score	Relevance
sport championship	World cup competition	0.12936	Y
	List of gymnasts	0.12784	N
	Korfball	0.09353	N
	Six Nations Championship	0.08004	Y
	Grappling	0.07967	N
	America's National Game	0.07864	N
	Netball	0.07738	N
	Floorball	0.07718	N
	UEFA	0.07444	Y
	Auto racing	0.07357	N

Table 19: Query 1 - sport championship - for basic model

Query	Document Title	Score	Relevance
sport championship	World cup competition	0.20322	Y
	UEFA	0.14885	Y
	FIFA World Cup	0.13725	Y
	Cricket World Cup	0.13102	Y
	Floorball	0.12122	N
	Valencia CF	0.11160	N
	Speed skating	0.10824	N
	FA Cup	0.10464	Y
	Netball	0.10460	N
	Six Nations Championship	0.10416	Y

Table 20: Query 1 - sport championship - with query expansion

Query	Document Title	Score	Relevance
Godzilla movies	Kenji Sahara	0.13626	N
	Terror of Mechagodzilla	0.10201	Y
	Godzilla vs. King Ghidorah	0.09422	Y
	Son of Godzilla	0.08895	Y
	Godzilla vs. Mothra	0.08869	Y
	Ebirah, Horror of the Deep	0.08622	Y
	The Return of Godzilla	0.08219	Y
	Godzilla	0.08032	Y
	Godzilla vs. Biollante	0.07947	Y
	King Kong vs. Godzilla	0.06986	Y

Table 21: Query 2 - Godzilla movies - for basic model

Query	Document Title	Score	Relevance
Godzilla movies	Terror of Mechagodzilla	0.12564	Y
	Godzilla vs. King Ghidorah	0.11777	Y
	King Kong vs. Godzilla	0.11746	Y
	Godzilla vs. Mothra	0.11463	Y
	Destroy All Monsters	0.10649	Y
	Ebirah, Horror of the Deep	0.10069	Y
	Son of Godzilla	0.10002	Y
	Godzilla	0.09918	Y
	The Return of Godzilla	0.09433	Y
	Godzilla vs. Biollante	0.08635	Y

Table 22: Query 2 - Godzilla movies - with query expansion

Query	Document Title	Score	Relevance
space shuttles	Shuttlecock	0.08452	N
	Space Shuttle Challenger	0.07464	Y
	Transport in Guatemala	0.06901	N
	Space Shuttle Endeavour	0.06735	Y
	Space Shuttle Discovery	0.06499	Y
	Outline of space science	0.05963	N
	Space Shuttle Columbia	0.05653	Y
	Space Shuttle Atlantis	0.05587	Y
	Space Shuttle	0.04849	Y
	Kennedy Space Center	0.04747	N

Table 23: Query 3 - space shuttles - for basic model

Query	Document Title	Score	Relevance
space shuttles	Space Shuttle Atlantis	0.15323	Y
	Space Shuttle Discovery	0.14796	Y
	Space Shuttle Challenger	0.14699	Y
	Space Shuttle Columbia	0.13886	Y
	Space Shuttle Endeavour	0.13548	Y
	Space Shuttle	0.11557	Y
	Kennedy Space Center	0.11474	N
	Astronaut	0.09742	N
	Apollo 9	0.09668	N
	Apollo 15	0.09352	N

Table 24: Query 3 - space shuttles - with query expansion

Analysing Results

The positive effect of adding synonyms is clearly visible in the above queries. The performance has improved for both generic and specific queries.

However, this does not solve the context issue completely. The synonyms may sometimes interfere with a straightforward query and tamper with the results. One such example is demonstrated below.

Query	Document Title	Score	Relevance
baseball teams	Colorado Rockies	0.09676	Y
	Cy Young Award	0.09518	N
	Albert Spalding	0.09473	N
	America's National Game	0.09463	N
	Miami Marlins	0.08842	Y
	Arizona Diamondbacks	0.08682	Y
	San Diego Padres	0.08122	Y
	Oakland Athletics	0.07144	Y
	Cy Young	0.07027	N
	Seattle Mariners	0.06899	Y

Table 25: Query - baseball teams - for basic model

Query	Document Title	Score	Relevance
baseball teams	American Football Conference	0.14583	N
	American Football League	0.13867	N
	NBA (disambiguation)	0.13504	N
	Arizona Diamondbacks	0.13444	Y
	Pittsburgh Steelers	0.13311	N
	Miami Marlins	0.13153	Y
	National Hockey League`	0.12899	N
	Arizona Cardinals	0.12670	N
	National Basketball Association	0.12458	N
	Cincinnati Bengals	0.12299	N

Table 26: Query - baseball teams - with query expansion

To understand why the results are worse in the case of query expansion, we look at the synonyms generated for baseball. Values in brackets indicate overall tf, not similarity with just baseball.

basketball(0.76), leagues(0.76), football(1.49), league(1.48), hockey(0.73), sports(0.71), nfl(0.70)

These “synonyms” produce results from all these sports, not just baseball. In fact, football is given more importance compared to baseball, because the term “teams” also generates football as a synonym.

Future Scope

There are various other techniques and heuristics aimed at achieving better performance for information retrieval. Some of them are mentioned below.

- Apart from removing punctuations, stemming, lemmatization and spelling correction can also be applied to terms in both query and documents, to obtain more word matches and display better results.
- The parameters that were used in the above models, such as zone weights (α), can be tuned using machine learning techniques instead of assigning their values manually. This may produce a better context representation for the terms in the corpus.
- The latency of retrieval can be further optimized by using techniques such as Cluster Pruning and Tiered Indexes

References

[1] Pennington, Jeffrey & Socher, Richard & Manning, Christopher. (2014). Glove: Global Vectors for Word Representation. EMNLP. 14. 1532-1543. 10.3115/v1/D14-1162.