REPORT

ON

TEXT PROCESSING - ASSIGNMENT

BY

Lakshya Agarwal	2017B5A70904P
Anuj Hydrabadi	2017A8PS0420P
Samarth Gupta	2017B4A70467P
Raj Shree Singh	2017B4A70808P
Aditya Vishwakarma	2017B5A70954P

Prepared in partial fulfillment of the course

CS F469 – Information Retrieval

TO

Prof. Vinti Agarwal

Assistant Professor

Computer Science and Information Systems

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

April 2021

Contents

Acknowledgement

We would like to thank Dr. Vinti Agarwal for this opportunity to pursue this course project of Information Retrieval course.

Furthermore, we would like to acknowledge with much appreciation the crucial role of our peers who helped us immensely during the course of the project.

Without the help and contribution of the above-mentioned people, this project would not have been a success.

1. Introduction

The objective of this project is to build an Information Retrieval System based on the Vector Space Model. The IR system runs on a corpus of unstructured documents obtained from the English Wikipedia. It will make an end-to-end system that reads and processes the HTML text corpus and creates an easy-to-use index on them. This index is further used to query the corpus for relevant documents through single or multi-term queries and returns a list of top K documents that it finds most relevant.

We even apply different heuristics such as searching for synonyms, making champion lists and zone indexing to improve the search results and thus better the performance of the search engine. The remainder of this assignment is as follows. In Section 2, we list the assumptions that we have used. Section 3 highlights the implementation of the IR system - describing the implementation, query analysis and their limitations. Section 4 highlights the improvements that we have proposed.

2. Assumptions

We have the following assumptions for our model:

1. Context does not matter in the free text query given by the user.
2. The indexing files containing the Posting Lists fit in the memory.
3. Terms can appear in the document independent of each other.

3. Model Description (Part 1)

3.1. Implementation Details

In this section, we describe the implementation of the IR system.

3.1.1. Index Creation

preprocess(): The first step is to create the index of words. First, we parse the content using BeautifulSoup library and tokenize the raw text.

build_index(): Then, dictionary data structure is used because of its constant look-up time. For each document parsed, the tokens are populated in the dictionary 'trm_freq' and parallelly necessary changes are made in the posting list to create inverted index, 'invtd_idx'. The dictionary consists of the term as the key and the value as another dictionary with document ids where the term appears as the key along with the frequency of the term in that document as the value. Example of a list: Posting_List[term] = {docid1: freq1, docid2: freq2, …….}

After all the documents are parsed, the indexes are dumped in the pickle files which will be used by the query parser. It is to note that the punctuation symbols are removed.

3.1.2. LNC.LTC Model

The Vector Space Model is a mathematical way of representing documents and queries as vectors. Each dimension corresponds to a separate token. If a token occurs in a document, its value in the vector is some non-zero value. We have used a tf-idf weighting scheme here.

After creating the vectors for query and documents using the lnc.ltc SMART notation (using normalization), scalar multiplication of vectors is done, and top 10 documents are retrieved. In lnc.ltc, the documents are weighed using logarithmic term frequency along with cosine normalization and query terms are weighed using **tf-idf** weighing scheme with cosine normalization.

get_term_document_weights(invtd_idx): This function uses lnc model for calculating term weights for documents. 'l' represents term frequency is considered using a logarithmic model that is the value of 1 plus logarithm of frequency of occurrence of the term in the

document. ‘n’ represents document frequency is not considered. ‘c’ represents normalization is done using cosine normalization. idf is also created here which will be used for query later in get_term_weights_for_query(). **idf** refers to the logarithm of ratio of total number of documents indexed to the number of documents containing the term.

get_term_weights_for_query(): This function tokenizes the query considering removal of the punctuations. For queries we are using ltc model. ‘l’ represents term frequency is considered using a logarithmic model that is the value of 1 plus logarithm of frequency of occurrence of the term in the query. ‘t’ represents the inverse document frequency which is already created in get_term_document_weights() function. ‘c’ is cosine normalization. So,

```
result[term] = (1 + math.log10(result[term])) * (idf.get(term, 0))
```

3.1.3. Query Search

load_index(): Loads both normal index and modified index as saved by build_index.py and assigns them as per requirements.

run_repl(): Facilitates Main menu functioning. Takes input query from the user in global variable ‘query’ and a choice from available options. Call search() function according to query_type entered by the user.

search(): calls get_term_weights_for_query() function to get query weights as per ltc scheme. To efficiently calculate document scores, term-at-a-time approach (bag of words) is used for query terms:

```
for term, query_weight in term_query_weights.items():

      for doc_id, doc_weight in term_doc_weights[term].items():

                  doc_id_score[doc_id] += query_weight * doc_weight
```

Then sort the document id as per the score in descending order. Lastly print score, document id and title for top k documents.

3.2. Query analysis

3.2.1. Documents Retrieved

We now test our model on 10 different multi-term queries. We report the relevance of the documents that are fetched by manually reading the documents for content and terms that match the searched terms. The Y/N in the relevance column thus represents a subjective score. These answers are reported upto a relevant decimal place only and are reported without the use of any heuristic.

Query	Top 10 Documents	Score	Is the Document Relevant?
Civil Rights	mlk (disambiguation)	0.2871	Yes
	international human rights instruments	0.1417	Yes
	franc poincaré	0.1294	No
	telecommunications in the republic of the congo	0.1137	Yes
	partizan press	0.1077	No
	procedural law	0.1058	Yes
	personal property	0.1041	Yes
	socialist law	0.1036	Yes
	communications in liberia	0.1001	No
	lao people's armed forces	0.0977	No

Query	Top 10 Documents	Score	Is the Document Relevant?
Cauchy Sequences	net (mathematics)	0.0911	Yes
	sequence	0.0790	Yes
	hausdorff space	0.0782	Yes
	metric space	0.0705	Yes
	real analysis	0.0639	Yes
	constructivism (mathematics)	0.0636	Yes
	series (mathematics)	0.0628	Yes
	fx-87	0.0545	No
	central moment	0.0506	Yes
	southern blot	0.0467	No

Query	Top 10 documents	Score	Is the document relevant to the query?
Peptide bond	peptide bond	0.1468	Yes
	peptide	0.1043	Yes
	protein primary structure	0.0967	Yes
	hydroxy group	0.0846	Yes
	proteolysis	0.0832	Yes
	peptidoglycan	0.0752	No
	protein biosynthesis	0.0684	Yes
	piperidine	0.0679	No
	hydrolysis	0.0671	Yes
	katal	0.0660	Yes

Query	Top 10 documents	Score	Is the document relevant to the query?
radioactive material	hershey–chase experiment	0.0917	Yes
	sn 1987a	0.0853	Yes
	particle radiation	0.0843	Yes
	kryptonite	0.0791	Yes
	half-life	0.0741	Yes
	neutron activation analysis	0.0736	Yes
	northern blot	0.0706	Yes
	radiometric dating	0.0706	Yes
	list of humorists	0.0704	No
	lutetium	0.0673	Yes

Query	Top 10 documents	Score	Is the document relevant to the query?
african countries	historical african place names	0.2870	Yes
	list of south africans	0.1995	Yes
	foreign relations of kenya	0.1483	Yes
	international african institute	0.1420	Yes
	foreign relations of south africa	0.1253	Yes
	demographics of martinique	0.1235	No
	panga	0.1226	No
	foreign relations of the republic of the congo	0.1169	Yes
	demographics of honduras	0.1073	No
	transport in guinea-bissau	0.1066	Yes

Query	Top 10 documents	Score	Is the document relevant to the query?
infinite series	infinite descending chain	0.1820	Yes
	geometric series	0.1039	Yes
	primitive notion	0.0909	No
	technology in star trek	0.0876	No
	symmetry group	0.0829	Yes
	series (mathematics)	0.0809	Yes
	maus (disambiguation)	0.0763	No
	product of rings	0.0760	No
	sequence	0.0758	Yes
	finite set	0.0752	Yes

Query	Top 10 documents	Score	Is the document relevant to the query?
Adam Smith	smith	0.3829	No
	joseph smith (disambiguation)	0.1830	No
	kuznetsov	0.1149	No
	johann friedrich endersch	0.0896	No
	garden of eden	0.0786	No
	smuggling in fiction	0.0759	No
	john sealy hospital	0.0752	No
	hendecasyllable	0.0744	No
	factors of production	0.0710	Yes
	patrilineality	0.0695	No

Query	Top 10 documents	Score	Is the document relevant to the query?
American football	italian football league	0.1601	No
	fc den bosch	0.1372	No
	line of scrimmage	0.1344	Yes
	lateral pass	0.1053	Yes
	phutball	0.1038	No
	snap (gridiron football)	0.0952	Yes
	mercy rule	0.0776	Yes
	rosmalen	0.0760	No
	knute rockne	0.0749	Yes
	john ford (disambiguation)	0.0740	No

Query	Top 10 documents	Score	Is the document relevant to the query?
digital signal processing	speech processing	0.3414	Yes
	pcm (disambiguation)	0.2739	Yes
	signal	0.2730	Yes
	filter	0.1915	Yes
	source separation	0.1633	Yes
	phase modulation	0.1357	Yes
	lowball	0.1285	No
	speech coding	0.1259	Yes
	linear prediction	0.1154	Yes
	linear timecode	0.1134	Yes

Query	Top 10 documents	Score	Is the document relevant to the query?
star trek	technology in star trek	0.1531	Yes
	list of star trek: enterprise episodes	0.1386	Yes
	maquis (star trek)	0.1176	Yes
	impulse drive	0.1159	Yes
	redshirt (character)	0.1046	Yes
	k. w. jeter	0.1032	Yes
	list of star trek: the original series episodes	0.0975	Yes
	leonard mccoy	0.0949	Yes
	starship enterprise	0.0945	Yes
	kathryn janeway	0.0933	Yes

3.2.2. Evaluation benchmarks

We now evaluate the **precision** of the system for each of the queries tested above.

$$Precision = \frac{total\ relevant\ documents\ retrieved}{total\ documents\ retrieved (= 10)}$$

Query	**Precision**
Civil rights	0.6
Cauchy sequences	0.8
Peptide bonds	0.8
Radioactive material	0.9
African countries	0.7
Infinite series	0.6
Adam smith	0.1
American football	0.5
Digital signal processing	0.9
Star trek	1

Recall is another measure to evaluate our system, and is defined as

$$Recall = \frac{Total\ relevant\ documents\ retrieved}{Total\ relevant\ documents\ in\ the\ corpus}$$

Evaluating the relevancy for each document in the corpus for each query is infeasible because of the large size of the corpus, hence we do not evaluate recall for the queries.

3.3. Limitations

Following are the limitations of the model:

1. Does not consider the context of text. For example, for the query ‘Adam Smith’, the context expected was the economist Adam Smith, but the system fails to identify it, giving a low precision.
2. In case the query words are not present in the corpus itself, there is no mechanism of giving suggestions of similar rooted words that are there in the corpus. For example, ‘african countries’ does not search for the similar rooted terms ‘country’ and ‘africa’.
3. The search results are based on a simple search score based on cosine similarity only. There is no means of weighting the documents with relevance based on title or any zone in the document. This is more relevant in case of Wikipedia documents.
4. Vector-space model provides a not-so-relevant retrieval model because of the bag of words nature of it.
5. The terms in the user query which have probably been spelt wrong are not corrected with the most probable suggestion and re-run. For example, ‘proecssing’ is not corrected to ‘processing’.
6. Context free synonyms may sometimes be irrelevant. For example, in case of proper nouns such as the names of persons like John, it may give irrelevant synonyms.
7. Smaller documents, especially those for disambiguation in Wikipedia, appear higher than longer, but more relevant documents because of their small size and higher corresponding score.

4. **Improvements (Part 2)**

4.1. Synonyms

Synonyms are useful when the query words in the document are acronyms or less commonly used words such as tv, occult, phantom. In this case, getting a list of synonyms first such as television for tv, magical for occult greatly increases the relevant queries as the queries search become larger by incorporating more common terms.

1. What are the issues with the vector space model built in part 1?

The IR System built matches the query term exactly to the term present in the document. It does not take any morphological analysis or the root word of the term into consideration.

2. What improvement are you proposing?

We are proposing to consider relevant words during search using nltk corpus.

3. How will the proposed improvement address that issue?

After suggested improvement query term now also includes synonyms of original query term which are there in corpus.

4. A corner case (if any) where this improvement might not work or can have an adverse effect.

Sometimes words may have different meanings and correspondingly different synonyms so it may predict wrong result. Example: consider the query, "bank of river", this query results "state street corporation" document as its 3rd result which is wrong.

Result may be biased to a particular term in query if have large number of synonyms existing in corpus**.** Example: consider the query, "listing of fictions", results are dominated by word list which is synonym of listing in this case.

5. Demonstrate the actual impact of the improvement. Give three queries, where the improvement yields better results compared to the part 1 implementation.

Query 1: *tv*

Before:

Query	Top k documents	Score	Is the document relevant for the query?
tv	rms laconia	0.1500	Yes

tv	telecommunications in slovenia	0.1132	Yes
tv	kiyoshi atsumi	0.0914	Yes
tv	telecommunications in greece	0.0890	Yes
tv	gordon michael woolvett	0.0877	Yes
tv	sorious samura	0.0871	Yes
tv	phoenix (tv series)	0.0861	Yes
tv	telecommunications in ghana	0.0822	Yes
tv	naked news	0.0803	Yes
tv	mv blue marlin	0.0782	Yes

After implementing synonyms:

Synonyms: *['television', 'telecasting', 'video', 'tv', 'telly']*

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
tv	gordon michael woolvett	0.0527	Yes
tv	gavin macleod	0.0445	Yes
tv	home improvement (tv series)	0.0418	Yes
tv	production team	0.0372	Yes
tv	standard-definition television	0.0370	Yes
tv	letterboxing (filming)	0.0351	Yes
tv	telecommunications in slovenia	0.0349	Yes
tv	mpeg-2	0.0344	Yes
tv	multicast	0.0335	Yes
tv	shock site	0.0331	No

Query 2: *occult*

Before:

Query	Top k documents	Score	Is the document relevant for the query?
occult	magical organization	0.1042	Yes
occult	occult	0.0966	Yes
occult	peter j. carroll	0.0755	Yes
occult	rudolf ii, holy roman emperor	0.0484	Yes
occult	hermetic order of the golden dawn	0.0429	Yes
occult	robert anton wilson	0.0420	Yes
occult	angel moroni	0.0416	No
occult	magick (thelema)	0.0412	Yes
occult	isaac bonewits	0.0403	Yes
occult	grimoire	0.0402	Yes

After implementing synonyms:

Synonyms*: ['supernatural', 'occult', 'eclipse', 'mysterious', 'mystic', 'mystical', 'secret', 'orphic']*

Query	Top k documents	Score	Is the document relevant for the query?
occult	occult	0.0742	Yes
occult	grimoire	0.0552	Yes
occult	mage: the ascension	0.0496	Yes
occult	gnosis	0.0469	Yes
occult	kabbalah	0.0431	Yes
occult	magick (thelema)	0.0425	Yes
occult	spirituality	0.0422	No
occult	high fantasy	0.0396	Yes
occult	persephone	0.0393	Yes
occult	magical organization	0.0388	Yes

Query 3: *large animals*

Before:

Query	Top k documents	Score	Is the document relevant for the query?
large animals	list of freshwater aquarium invertebrate species	0.1427	No
large animals	warm-blooded	0.0924	No
large animals	nucleariid	0.0911	No
large animals	shooting	0.0905	No
large animals	neoproterozoic	0.0866	No
large animals	sandpit	0.0818	No
large animals	gestation	0.0793	No
large animals	indriidae	0.0791	Yes
large animals	phantom kangaroo	0.0789	Yes
large animals	in vivo	0.0775	No

After:

Synonyms: *['large', 'big', 'bombastic', 'declamatory', 'turgid', 'magnanimous', 'prominent', 'enceinte', 'expectant', 'great', 'heavy', 'boastfully', 'animal', 'beast', 'brute', 'creature', 'fauna', 'animals']*

Query	Top k documents	Score	Is the document relevant for the query?
large animals	mokele-mbembe	0.0279	Yes
large animals	neoproterozoic	0.0243	No
large animals	lupus (constellation)	0.0242	No
large animals	phantom kangaroo	0.0235	Yes
large animals	loch ness monster	0.0214	Yes

large animals	pet	0.0207	No
large animals	mecha	0.0203	Yes
large animals	sidehill gouger	0.0202	Yes
large animals	king kong (1933 film)	0.0202	Yes
large animals	llama	0.0199	Yes

4.2. Champion list

We tried to increase the speed of execution by precomputing a list of 200 documents which have the highest weight for a particular term per their term frequency. Through this, we aim to increase the speed of execution by avoiding the computation of all document rankings at query time. The speed of execution increases by roughly 3 times even though we are using a relatively small corpus of 9610 documents. Its importance in case of large corpus will be highly amplified. The following example illustrates this:

create_chmp_lst(): Creates champion list for each word in the corpus. Gets the posting list corresponding to each word and get a minimum of most common top 200 documents or length posting list after sorting them in reverse order of number of occurrences of word in each doc. 200 makes approximately 2% of 9610 documents in the selected corpus of wiki files wiki_11 to wiki_29 of AA folder. To calculate document scores more efficiently, term-at-a-time approach (bag of words) is used for query terms:

```
for term, qry_wgt in trm_qry_wgts.items():

      for doc_id, doc_weight in trm_doc_wgt[term].items():

            if (qry_typ == 2 or qry_typ == 4) and doc_id not in chmp_dcs:

            continue

      doc_id_score[doc_id] += qry_wgt * doc_weight
```

1. What are the issues with the vector space model built in part 1?

In vector space model time consumed will be more because it will consider all documents some of which are less relevant. The effect gets highlighted as the size of the corpus increases.

2. What improvement are you proposing?

Selecting minimum of most common top 200 documents or length posting list after sorting them in reverse order of number of occurrences of word in each doc. 200 makes approximately 2% of 9610 documents in the selected corpus of wiki files wiki_11 to wiki_29 of AA folder.

3. How will the proposed improvement address that issue?

Time of calculating score reduces to 33% as compared to one calculated without using champion list.

4. A corner case (if any) where this improvement might not work or can have an adverse effect.

Relevant documents may get missed out. Query: “list of science fictions”

Additionally, for large queries will not have much effect as union of documents occurring in champion list of at least one term is taken. Example: the query "There is no need to build a browser/GUI interface. Display of the output on the terminal/notebook would suffice." will have very less time reduction.

5. Demonstrate the actual impact of the improvement. Give three queries, where the improvement yields better results compared to the part 1 implementation.

Query 1: *science fiction*

Before: (Time taken**: 0.797 ms**)

Query	Top k documents	Score	Is the document relevant for the query?
science fiction	list of science fiction themes	0.4730	Yes
science fiction	list of science fiction and fantasy artists	0.2473	Yes
science fiction	lists of science fiction films	0.2045	Yes
science fiction	philip k. dick award	0.1788	Yes
science fiction	first fandom	0.1784	Yes
science fiction	outline of literature	0.1721	Yes
science fiction	laws of infernal dynamics	0.1619	Yes
science fiction	katherine maclean	0.1584	Yes
science fiction	peter nilson	0.1543	Yes

science fiction	james tiptree jr. award	0.1499	Yes

After: (Time taken: **0.612 ms**)

Query	Top k documents	Score	Is the document relevant for the query?
science fiction	list of science fiction and fantasy artists	0.2392	Yes
science fiction	lists of science fiction films	0.1956	Yes
science fiction	philip k. dick award	0.1731	Yes
science fiction	first fandom	0.1660	Yes
science fiction	katherine maclean	0.1477	Yes
science fiction	james tiptree jr. award	0.1403	Yes
science fiction	greg egan	0.1395	Yes
science fiction	smuggling in fiction	0.1357	Yes
science fiction	geoff ryman	0.1347	Yes
science fiction	ken macleod	0.1270	Yes

Query 2: *computer science is a good subject*

Before: (Time taken: **6.262 ms**)

Query	Top k documents	Score	Is the document relevant for the query?
computer science is a good subject	list of science fiction themes	0.1865	No
computer science is a good subject	knowledge systems laboratory	0.1732	Yes
computer science is a good subject	kl0	0.1411	Yes
computer science is a good subject	goodtimes virus	0.1367	Yes
computer science is a good subject	fred brooks	0.1316	Yes

computer science is a good subject	niklaus wirth	0.1290	Yes
computer science is a good subject	kiss (system)	0.1250	Yes
computer science is a good subject	self-reference	0.1237	Yes
computer science is a good subject	quantum information	0.1116	Yes
computer science is a good subject	phil zimmermann	0.1114	Yes

After: (Time taken: **2.890 ms**)

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
computer science is a good subject	goodtimes virus	0.1490	Yes
computer science is a good subject	fred brooks	0.1459	Yes
computer science is a good subject	niklaus wirth	0.1446	Yes
computer science is a good subject	indian institute of technology kanpur	0.1104	Yes
computer science is a good subject	microassembler	0.1069	Yes
computer science is a good subject	stan kelly-bootle	0.1067	Yes
computer science is a good subject	programmer	0.1009	Yes
computer science is a good subject	metasyntactic variable	0.1002	Yes
computer science is a good subject	john brunner (novelist)	0.0999	No
computer science is a good subject	international olympiad in informatics	0.0993	Yes

Query 3: *genetic disorder*

Before: (Time taken: **0.350 ms**)

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
genetic disorder	genetic disorder	0.1124	Yes

genetic disorder	monoamine oxidase	0.0901	No
genetic disorder	foix–alajouanine syndrome	0.0844	No
genetic disorder	lafora disease	0.0797	Yes
genetic disorder	kay redfield jamison	0.0796	No
genetic disorder	lithium citrate	0.0771	No
genetic disorder	muscular dystrophy	0.0740	Yes
genetic disorder	mutation	0.0727	Yes
genetic disorder	inclusion body myositis	0.0715	Yes
genetic disorder	haemophilia	0.0690	Yes

After: (Time taken**: 0.279 ms**)

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
genetic disorder	genetic disorder	0.1030	Yes
genetic disorder	monoamine oxidase	0.0879	No
genetic disorder	foix–alajouanine syndrome	0.0858	No
genetic disorder	lithium citrate	0.0789	No
genetic disorder	lafora disease	0.0780	Yes
genetic disorder	kay redfield jamison	0.0754	No
genetic disorder	muscular dystrophy	0.0721	Yes
genetic disorder	mutation	0.0687	Yes
genetic disorder	inclusion body myositis	0.0682	Yes
genetic disorder	haemophilia	0.0648	Yes

4.3. Zone indexing

The reason that zone indexing works very well is that we are working on HTML pages of the Wikipedia corpus. We give **more weight to documents that have the query words in the title itself** as it is quite intuitive that the title has more importance since we are dealing with Wikipedia

corpus. Hence instead of applying zone weighting from the very beginning, we increase the overall score of the documents in which a query term occurs in the title by a factor of 0.2 for every query term. The improved score is illustrated in the following examples - jaguar, lunar eclipse, and Jaffrey Dahmer.

get_trm_ttl_wgt() is created similar to get_term_document_weights() for titles of documents. It also uses same lnc scheme and creates title_idf to be used by get_term_weights_for_query(). To efficiently calculate document scores, term-at-a-time approach (bag of words) is used for query terms:

```
for term, qry_wgt in trm_qry_wgts.items():

      for doc_id, title_weight in trm_ttl_wgt[term].items():

            if (qry_typ == 2 or qry_typ == 4) and doc_id not in chmp_dcs:

            continue

      doc_id_score[doc_id] += qry_wgt * title_weight * 0.2
```

1. What are the issues with the vector space model built in part 1?

Titles of documents are not given any importance, but it is quite intuitive that while dealing with Wikipedia Corpus titles have more importance.

2. What improvement are you proposing?

Improved proposed is that assigned score for a (query, document) pair will be linear combination of zone score. Two zones '**content'** and '**title'** have weights of **0.8** and **0.2**, respectively.

3. How will the proposed improvement address that issue?

Now titles also have weights, so if titles have query terms, then they will be ranked higher as compared to other documents.

4. A corner case (if any) where this improvement might not work or can have an adverse effect.

When our query words are not in any of the title then effectively it has no effect on ranking of the document. Example: consider the query "Swakopmund missionary" is having exact same top 10 documents whether zone indexing is used or not.

For the query ‘marvel superhero’, undesirable documents like ‘history of marvel’ might get a precedence over documents like ‘spider-man’ or ‘iron man’, as they have the term ‘marvel’ in the title.

5. Demonstrate the actual impact of the improvement. Give three queries, where the improvement yields better results compared to the part 1 implementation.

Query 1: *jaguar*

Before:

Query	Top k documents	Score	Is the document relevant for the query?
jaguar	godzilla vs. megalon	0.0686	No
jaguar	jaguar	0.0584	Yes
jaguar	murray gell-mann	0.0525	No
jaguar	lunar eclipse	0.0496	No
jaguar	hearse	0.0492	Yes
jaguar	transport in qatar	0.0472	Yes
jaguar	land rover	0.0429	Yes
jaguar	jacksonville jaguars	0.0418	No
jaguar	international formula 3000	0.0365	Yes
jaguar	inspector morse	0.0356	No

After:

Query	Top k documents	Score	Is the document relevant for the query?
jaguar	jaguar	0.2467	Yes
jaguar	godzilla vs. megalon	0.0549	No
jaguar	murray gell-mann	0.0420	No
jaguar	lunar eclipse	0.0397	No
jaguar	hearse	0.0394	Yes

jaguar	transport in qatar	0.0378	Yes
jaguar	land rover	0.0343	Yes
jaguar	jacksonville jaguars	0.0334	No
jaguar	international formula 3000	0.0292	Yes
jaguar	inspector morse	0.0285	No

Query 2: *lunar eclipse*

Before:

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
lunar eclipse	lunar eclipse	0.1207	Yes
lunar eclipse	lunar phase	0.0864	Yes
lunar eclipse	metonic cycle	0.0812	Yes
lunar eclipse	lunar calendar	0.0703	No
lunar eclipse	new moon	0.0692	Yes
lunar eclipse	full moon	0.0644	Yes
lunar eclipse	hipparchus	0.0626	Yes
lunar eclipse	galatia	0.0566	No
lunar eclipse	solar deity	0.0563	No
lunar eclipse	moon	0.0522	Yes

After:

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
lunar eclipse	lunar eclipse	0.2958	Yes

lunar eclipse	lunar phase	0.1598	Yes
lunar eclipse	lunar calendar	0.1470	No
lunar eclipse	lunar roving vehicle	0.1070	No
lunar eclipse	lunar society of birmingham	0.0982	No
lunar eclipse	metonic cycle	0.0650	Yes
lunar eclipse	new moon	0.0553	Yes
lunar eclipse	full moon	0.0515	Yes
lunar eclipse	hipparchus	0.0501	Yes
lunar eclipse	galatia	0.0453	No

Query 3: *civil war*

Before:

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
civil war	partizan press	0.2119	Yes
civil war	mlk (disambiguation)	0.1719	No
civil war	karl andree	0.1486	No
civil war	kab-500l	0.1147	No
civil war	telecommunications in the republic of the congo	0.1109	No
civil war	geoff ryman	0.1037	No
civil war	knaresborough castle	0.1020	No
civil war	li people	0.1016	No
civil war	gustave de molinari	0.1013	No
civil war	george abbot (author)	0.0986	No

After:

Query	Top k documents	Score	Is the document relevant for the query?
civil war	irish civil war	0.2136	Yes
civil war	russian civil war	0.2067	Yes
civil war	finnish civil war	0.2049	Yes
civil war	partizan press	0.1695	Yes
civil war	reforms of amānullāh khān and civil war	0.1561	Yes
civil war	mlk (disambiguation)	0.1375	No
civil war	international civil aviation organization	0.1274	No
civil war	spanish–american war	0.1192	No
civil war	karl andree	0.1189	No
civil war	on war	0.1162	No

4.4. Combination

Finally, we use a combination of **synonyms, zone indexing** and **champion list** simultaneously. This is to indicate the scalability of our system that all the changes can work independently as well as in combination with other improvements in the system.

Query 1: *rifle*

Before: (Time taken: **0.1113 ms**)

Query	Top k documents	Score	Is the document relevant for the query?
rifle	shooting	0.0932	Yes
rifle	shooting sport	0.0800	Yes
rifle	heckler & koch	0.0777	Yes
rifle	military of moldova	0.0726	Yes
rifle	military of mauritius	0.0711	Yes

rifle	mikhail kalashnikov	0.0686	Yes
rifle	armed forces of the republic of kyrgyzstan	0.0675	Yes
rifle	king's royal rifle corps	0.0666	Yes
rifle	single-shot	0.0637	Yes
rifle	armed forces of the republic of kazakhstan	0.0612	Yes

After: (Time taken**: 0.1293 ms**)

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
rifle	rifle	0.0621	Yes
rifle	m16 rifle	0.0430	Yes
rifle	gaza strip	0.0362	No
rifle	king's royal rifle corps	0.0340	Yes
rifle	go down moses	0.0247	No
rifle	medieval warfare	0.0196	Yes
rifle	huns	0.0194	No
rifle	roger the dodger	0.0178	No
rifle	shooting	0.0174	Yes
rifle	joe orton	0.0154	No

Query 2: *Marvel superhero*

Before: (Time taken: **0.0729 ms**)

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
marvel superhero	marvel super heroes (role-playing game)	0.0780	Yes

marvel superhero	marvel comics	0.0750	Yes
marvel superhero	shang-chi	0.0714	Yes
marvel superhero	latveria	0.0657	Yes
marvel superhero	golden heroes	0.0642	Yes
marvel superhero	gurps supers	0.0639	No
marvel superhero	spider-man	0.0626	Yes
marvel superhero	retroactive continuity	0.0593	No
marvel superhero	james spader	0.0578	Yes
marvel superhero	pat mills	0.0572	Yes

After: (Time taken**: 0.1167 ms**)

Query	**Top k documents**	**Score**	**Is the document relevant for the query?**
marvel superhero	marvel comics	0.1315	Yes
marvel superhero	marvel universe	0.1204	Yes
marvel superhero	marvel super heroes (role-playing game)	0.0953	Yes
marvel superhero	spider-man	0.0478	Yes
marvel superhero	robin wright	0.0467	No
marvel superhero	shang-chi	0.0462	Yes
marvel superhero	golden heroes	0.0452	No
marvel superhero	latveria	0.0441	Yes
marvel superhero	gurps supers	0.0414	No
marvel superhero	james spader	0.0413	Yes

Query 3: *south georgia*

Before: (Time taken: **1.860 ms**)

Query	Top k documents	Score	Is the document relevant for the query?
south georgia	foreign relations of georgia	0.1730	Yes
south georgia	demographics of georgia (country)	0.1399	No
south georgia	telecommunications in georgia (country)	0.1122	No
south georgia	geography of georgia (country)	0.1051	No
south georgia	geography of suriname	0.0987	No
south georgia	transport in georgia (country)	0.0962	No
south georgia	politics of georgia (country)	0.0834	No
south georgia	grits	0.0828	No
south georgia	list of south africans	0.0808	No
south georgia	south georgia and the south sandwich islands	0.0789	Yes

After: (Time taken: **2.587 ms**)

Query	Top k documents	Score	Is the document relevant for the query?
south georgia	foreign relations of georgia	0.1487	Yes
south georgia	demographics of georgia (country)	0.1204	No
south georgia	south georgia and the south sandwich islands	0.1139	Yes
south georgia	telecommunications in georgia (country)	0.1028	No
south georgia	geography of georgia (country)	0.1020	No
south georgia	transport in georgia (country)	0.0969	No
south georgia	politics of georgia (country)	0.0908	No
south georgia	mozambique channel	0.0888	No
south georgia	economy of georgia (country)	0.0808	No
south georgia	south america	0.0739	No

5. Innovations

We list some novel methods or techniques that allow our code to run more efficiently and make our information retrieval system robust.

1. BeautifulSoup: The BeautifulSoup library has been used to extract data from the dataset, which is a set of Wikipedia documents in html form. We parse the document using BeautifulSoup and then tokenize the raw text.

2. NLTK: NLTK, or Natural Language Toolkit, is one of the leading suites of libraries for natural language processing for the English language. We use NLTK to tokenize the raw text obtained after parsing the document.

3. Synonyms (NLTK Synset): To obtain the synonyms of a word, we have used WordNet, which is lexical database for NLTK. The synsets() method provides instances are the groupings of synonymous words that express the same concept.

4. Bag of words model: To efficiently calculate document scores, term-at-a-time approach (bag of words) is used for query terms, that makes the calculation of score for each document quicker.

5. Using defaultdict data structure: Defaultdict is a container like dictionaries present in the module collections. Defaultdict is a sub-class of the dict class that returns a dictionary-like object. The functionality of both dictionaries and defualtdict are almost same except for the fact that defualtdict never raises a KeyError.

6. Pickle files: The indexes created by the build_index.py file have been stored in a .pickle file format. Pickling is serialization of python data structures for storage, followed by deserialization when needed for use (in the test_queries.py file).