Assignment 1

Theory Questions

1. What is information retrieval, and how is it different from data retrieval? For the following scenarios, which one of the data retrieval or information retrieval would you recommend to use? Explain your reasoning.

a) A clerk in pharmacy uses the following query:

Medicine_name = Ibuprofen_200mg

This query would be most suitable as a data retrieval query, searching a database with attribute/index “Medicine_name”; thus the query *should* lead to a direct result for the query.

b) A clerk in pharmacy uses the following query:

An antibiotic medicine.

This query is most suitable for information retrieval, searching documents for the query to find relevant articles (inventory) fitting the query. Since there are many possible valid results we want to retrieve information for all relevant documents.

c) A clerk in pharmacy uses the following query:

Medicine from the antibiotic family, without headache as side effect.

This query is also most suitable as information retrieval, although more specific than the last. Here we can as in the former question have many relevant results of antibiotics, with a negation of side effects for example; or looking for without headache specifically.

d) Searching an E-commerce website using the following query to find an specific shoe:

Brooks Ghost 15

Although we are looking for a specific document with this query, it would not be suitable for data retrieval as it doesn’t directly search for any attributes. Rather we would again use IR to find relevant articles on the website, which *could* result in one result depending on the IR model.

e) Searching the same E-commerce website using the following query:

Nice running shoes

This would use IR to find relevant articles, due to the broad search it would likely have a lot of results as it would be searching for text within the articles.

f) Searching for the schedule of a flight using the following query:

Flight_ID = ZEFV2

This query would be most suitable as data retrieval, as we know a key in the table we'd be searching; thus we are only looking for the one result with the corresponding ID.

2. Describe what stemming is. How does it affect precision and recall? For which of the following scenarios would you recommend to use stemming?

Stemming is reducing words in a query to its root forms. This is done to normalise the document collection for easier search, this way no matter the form of the word, relevant documents should appear.

a) Searching Trump's tweets with the following query:

Building a wall in the border

I would recommend using stemming for this query, as relevant documents could be using the word 'build' rather than building, thus stemming could result in more relevant searches for the user.

b) Searching Trump's tweets with the following query:

Hunter Biden's laptop.

This query does seem to me stemmable, nor would it need to as you would be searching for a specific sequence of words in the document.

c) Searching in the MyFakeShop.com for a pants using the following query:

Levi's 502 TAPER

This query would also not be suitable for stemming, as it is searching for specific data on the collection, stemming TAPER, would likely lead to irrelevant results.

d) Searching in the MyFakeShop.com for a pants using the following query:

A nice blue jeans with straight leg style

Although it doesn't contain many stemmable words, this sort of query would be suitable for stemming.

The Boolean Model

Consider the toy document collection as described below:

doc1 = AltaVista is a very old web search engine.
doc2 = Google is a new web search engine.
doc3 = AltaVista is offline.
doc4 = Google is not offline.
doc5 = Bing is a new web search engine.
doc6 = Bing is not offline.

1. Construct a Boolean term-document matrix for the toy document collection. (You can omit stopwords (e.g., is, and a) from the vocabulary.)

	AltaVista	Google	Bing	Web	Search	Engine	Very	Old	Offline
doc1	1	0	0	1	1	1	1	1	0
doc2	0	1	0	1	1	1	0	0	0
doc3	1	0	0	0	0	0	0	0	1
doc4	0	1	0	0	0	0	0	0	1
doc5	0	0	1	1	1	1	0	0	0
doc6	0	0	1	0	0	0	0	0	1

2. Using the matrix, identify the answer set of the following queries:

Q1 : web ∧ search ∧ engine
Q2 = offline ∨ (very ∧ old)
Q3 = (offline ∨ (very ∧ old)) ∧ (web ∧ search ∧ engine)

Q1: Query searching for documents including ‘web’, ‘search’, and ‘engine’.

Using the term-document matrix we see that doc1, doc2, and doc5 matches the query.

Q2: Query searching for documents including either offline or ‘very’ and ‘old’.

Again using the matrix we can see that doc3, doc4, and doc6 matches the first part of our query, doc one matches the second part of the or. Thus the query should result in doc1, doc3, doc4, and doc6

Q3: Query searching for documents including ‘web’, ‘search’, and ‘engine’, in addition to either ‘offline’, or ‘very’ and ‘old’.

Looking at the second part of the query, doc1, doc2, and doc5 matches the requisite. Only one of these documents meet the second requirement of the query, doc1, thus doc1 will be the only result.

3. Propose the simplest possible query (containing a minimum number of vocabulary), which is only true for the first document. Is there more than one answer?

Setting our query to only search for, ‘very’, or ‘old’ will result in only doc1 being returned, these are thus the shortest one word queries.

IR Models

Assuming the following document collection, which contains only the words from the set *O = {Large, Small, Elephant, Mouse}*.

doc1 = {Small Mouse Large Elephant}
doc2 = {Elephant}
doc3 = {Elephant Mouse}
doc4 = {Small Mouse Small Large Mouse Elephant}
doc5 = {Small Large}
doc6 = {Large Mouse Elephant Small}
doc7 = {Small Small Small}
doc8 = {Elephant Mouse Mouse}
doc9 = {Mouse Large}
doc10 = {Large Large Small Elephant}

1. Calculate the weights for the documents and the terms using tf and idf weighting. Put these values into a document-term-matrix. Use the formula below:

$$tf_{i,j} = Raw\ frequency$$

$$idf_t = lg\frac{N}{df_t}$$

TF$_{i,j}$	*doc1*	*doc2*	*doc3*	*doc4*	*doc5*	*doc6*	*doc7*	*doc8*	*doc9*	*doc10*
Large	1	0	0	1	1	1	0	0	1	2
Small	1	0	0	2	1	1	3	0	0	1
Elephant	1	1	1	1	0	1	0	1	0	1
Mouse	1	0	1	2	0	1	0	2	1	0

IDF$_t$	
Large	$lg\frac{10}{6} = 0.737$
Small	$lg\frac{10}{6} = 0.737$
Elephant	$lg\frac{10}{7} = 0.515$
Mouse	$lg\frac{10}{6} = 0.737$

TF-IDF	*doc1*	*doc2*	*doc3*	*doc4*	*doc5*	*doc6*	*doc7*	*doc8*	*doc9*	*doc10*
Large	0.737	0	0	0.737	0.737	0.737	0	0	0.737	1.474
Small	0.737	0	0	1.474	0.737	0.737	2.211	0	0	0.737
Elephant	0.515	0.515	0.515	0.515	0	0.515	0	0.515	0	0.515
Mouse	0.737	0	0.737	1.474	0	0.737	0	1.474	0.737	0

****not including normal tf weight using lg, as only one value would be affected.*

2. Calculate the similarity score of all the documents by their relevance to the query: *q = Mouse.* **Use**

1. Cosine similarity,
2. Euclidean distance
3. Euclidean distance normalised.

Are there any differences in the ranking of the documents? Why is that?

$$\hat{d} = \sqrt{\sum_{i=1}^{|V|} w_i^2}$$

	doc1	*doc2*	*doc3*	*doc4*	*doc5*	*doc6*	*doc7*	*doc8*	*doc9*	*doc10*
$\hat{d}$	2.270	0.515	0.899	2.270	1.042	2.270	2.211	1.561	1.042	1.726

1. Cosine similarity:

	doc1	*doc2*	*doc3*	*doc4*	*doc5*	*doc6*	*doc7*	*doc8*	*doc9*	*doc10*
$\frac{d \bullet q}{\hat{d}}$	0.649	0	1.640	0.479	0	0.649	0	0.696	1.415	0

2. Euclidean distance:

$dis(q, d) = \sqrt{\sum_{i=0}^{d.len} (q_i - d_i)^2}$ *~ replace q = null with 0 (?)*

	doc1	*doc2*	*doc3*	*doc4*	*doc5*	*doc6*	*doc7*	*doc8*	*doc9*	*doc10*
	1.163	0.899	0.515	1.877	1.277	1.163	2.331	0.899	0.737	1.877

3. Given the following queries:

q_1 = "Mouse Elephant"
q_2 = "Large"

a) What are the main differences between the BM25 model and the probabilistic model introduced by Robertson-Jones?

The probabilistic model uses the same similarity function as BM25, BM25 adds the B function in addition to this which is multiplied by the sim-score.

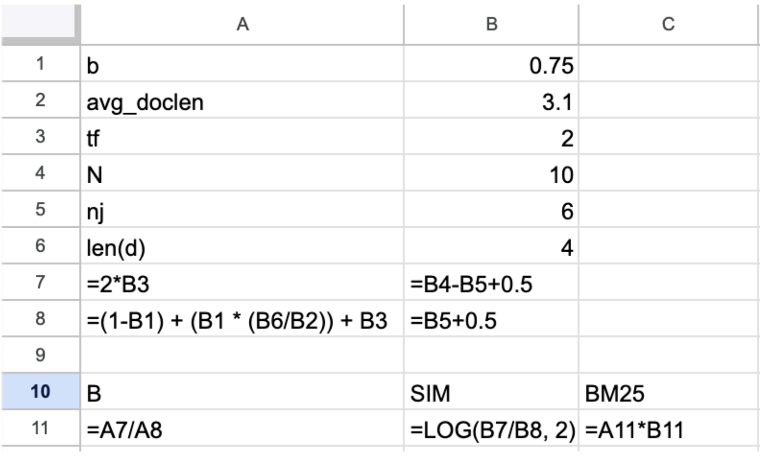
b) Assuming absence of relevance information, rank the documents according to the two queries, using the BM25 model. Set the parameters of the equation as suggested in the literature. Write clearly all the calculations.

$$BM25(d_i, q) \sim \sum_{w_j \in q \wedge w_j \in d_i} \frac{(K_1 + 1) \cdot tf_{i,j}}{K_1[(1-b) + b \cdot \frac{len(d_i)}{avglen}] + tf_{i,j}} \cdot lg\frac{N - n_j + 0.5}{n_j + 0.5}$$

$K_1 = 1$, $b = 0.75$, $avg\_len = 3.1$

	doc1	*doc2*	*doc3*	*doc4*	*doc5*	*doc6*	*doc7*	*doc8*	*doc9*	*doc10*
$q_{1\text{-}1}$	-0.478	0	-0.612	-0.573	0	-0.478	0	-0.713	-0.612	0
$q_{1\text{-}2}$	-0.992	-1.474	-1.268	-0.814	0	-0.992	0	-1.113	0	-0.992
q_1	-1.470	-1.474	-1.880	-1.387	0	-1.470	0	-1.826	-0.612	-0.992
q_2	-0.478	0	0	-0.393	-0.612	-0.478	0	0	-0.612	-0.659

Calculation provided by Google Sheets:



	A	B	C
1	b	0.75	
2	avg_doclen	3.1	
3	tf	2	
4	N	10	
5	nj	6	
6	len(d)	4	
7	=2*B3	=B4-B5+0.5	
8	=(1-B1) + (B1 * (B6/B2)) + B3	=B5+0.5	
9			
10	B	SIM	BM25
11	=A7/A8	=LOG(B7/B8, 2)	=A11*B11