School of Computing and Information Systems
The University of Melbourne
COMP90049 Knowledge Technologies (Semester 1, 2019)
Workshop sample solutions: Week 6

1. In the context of an **information retrieval** engine, what does it mean for a document to be **returned** for a query? What does it mean for a document to be **relevant** for a query?

   - An information retrieval engine returns documents with respect to a query, usually based on the presence of the keywords (from the query) in the document.
   - A document is relevant for a query if it meets the user's information need (approximated by the query) note that there is no requirement for the query keywords to be present!

2. Identify some differences between **Boolean** querying and **ranked** querying, in an Information Retrieval context.

   - Boolean: documents match if they contain the terms (and don't contain the `NOT` terms; i.e. the Boolean formula evaluates to `TRUE`); matching is Yes/No; repeatable, auditable, controllable; queries allow expression of complex concepts
   - Ranked: based on evidence that the document is on the same topic as the query; matching is gradiated (to come up with a ranking!); different models give different results; queries are easy to write and results are easy to read for non-specialists

3. Identify the two (sometimes three) components of a **TF-IDF model**. Indicate the rationale behind them as in, why would they contribute to a "better" result set?

   - More weight is given to documents where the query terms appear many times (TF)
   - Less weight is given to terms that appear in many documents (IDF)
   - Less weight is given to documents that have many terms (not present in all models)

4. Many TF-IDF models are possible; consider the following one:

$$w_{d,t} = \begin{cases} 1 + \log_2 f_{d,t} & \text{if } f_{d,t} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_{q,t} = \begin{cases} \log(1 + \frac{N}{f_t}) & \text{if } f_{q,t} > 0 \\ 0 & \text{otherwise} \end{cases}$$

   (a) Construct suitable vectors for the five documents in the collection below, and then use the **cosine measure** to determine the document ranking for the (conjunctive) query `apple lemon`:

DocID	`apple`	`ibm`	`lemon`	`sun`
Doc_1	4	0	0	1
Doc_2	5	0	5	0
Doc_3	2	5	0	0
Doc_4	1	2	1	7
Doc_5	1	1	3	0

   - First, we need to calculate the weights of terms in the documents; for example, the weight of `apple` in document 1 is $1+\log_2(4)$, because we have seen 4 instances of the term `apple`

($f_{a_1} = 4 > 0$):

$$\begin{aligned}
w_{a,1} &= 1 + \log_2(4) = 3 \\
w_{i,1} &= 0 \\
w_{l,1} &= 0 \\
w_{s,1} &= 1 + \log_2(1) = 1 \\
w_{a,2} &= 1 + \log_2(5) \approx 3.32 \\
w_{i,2} &= 0 \\
w_{l,2} &= 1 + \log_2(5) \approx 3.32 \\
w_{s,2} &= 0 \\
w_{a,3} &= 1 + \log_2(2) = 2 \\
w_{i,3} &= 1 + \log_2(5) \approx 3.32 \\
w_{l,3} &= 0 \\
w_{s,3} &= 0 \\
w_{a,4} &= 1 + \log_2(1) = 1 \\
w_{i,4} &= 1 + \log_2(2) = 2 \\
w_{l,4} &= 1 + \log_2(1) = 1 \\
w_{s,4} &= 1 + \log_2(7) \approx 3.81 \\
w_{a,5} &= 1 + \log_2(1) = 1 \\
w_{i,5} &= 1 + \log_2(1) = 1 \\
w_{l,5} &= 1 + \log_2(3) \approx 2.58 \\
w_{s,5} &= 0
\end{aligned}$$

- Next, we need the weights for the query vector. This is based on the inverse document frequency (IDF), which in this case is the inverse of the proportion of documents in which the term appears (f_t) out of the total number of documents ($N = 5$). We could also do this on a per-term basis, but we would need to re-visit our notion of what "rare" means in this context.

$$\begin{aligned}
w_{a,q} &= \log_2(1 + \frac{5}{5}) = 1 \\
w_{i,q} &= 0 \\
w_{l,q} &= \log_2(1 + \frac{5}{3}) \approx 1.42 \\
w_{s,q} &= 0
\end{aligned}$$

- Let's summarise. At this point, our representation of the documents (and query) are the following 4-D vectors:

$$\begin{aligned}
\text{Doc}_1 &: \langle 3, 0, 0, 1 \rangle \\
\text{Doc}_2 &: \langle 3.32, 0, 3.32, 0 \rangle \\
\text{Doc}_3 &: \langle 2, 3.32, 0, 0 \rangle \\
\text{Doc}_4 &: \langle 1, 2, 1, 3.81 \rangle \\
\text{Doc}_5 &: \langle 1, 1, 2.58, 0 \rangle \\
q &: \langle 1, 0, 1.42, 0 \rangle
\end{aligned}$$

- To use the vector space model, we calculate the cosine similarity based on the inner product (dot product), normalised by the vector norms[1] (length; this also accounts for

[1]Note that we can pre-calculate the (vector) length of our documents as soon as we've seen them (and not at query-time, when the user is waiting). Note also that the query length is irrelevant, because it's the same factor for every document, and all we really care about is the document ordering, not the actual scores.

the document length component in our TF-IDF model):

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

- So, the scores for our 5 documents:

$$\begin{aligned}
cos(\text{Doc}_1, q) &= \frac{\text{Doc}_1 \cdot q}{|\ \text{Doc}_1\ |\ \ |\ q\ |} \\
&= \frac{\langle 3,0,0,1\rangle \cdot \langle 1,0,1.42,0\rangle}{|\ \langle 3,0,0,1\rangle\ |\ \ |\ \langle 1,0,1.42,0\rangle\ |} \\
&= \frac{3*1+0*0+0*1.42+1*0}{\sqrt{3^2+0^2+0^2+1^2}\sqrt{1^2+0^2+1.42^2+0^2}} \\
&= \frac{3}{\sqrt{10}\sqrt{3.02}} \approx 0.55 \\
cos(\text{Doc}_2, q) &= \frac{\langle 3.32,0,3.32,0\rangle \cdot \langle 1,0,1.42,0\rangle}{|\ \langle 3.32,0,3.32,0\rangle\ |\ \ |\ \langle 1,0,1.42,0\rangle\ |} \\
&= \frac{3.32*1+0*0+3.32*1.42+0*0}{\sqrt{3.32^2+0^2+3.32^2+0^2}\sqrt{1^2+0^2+1.42^2+0^2}} \\
&\approx \frac{8.03}{\sqrt{22}\sqrt{3.02}} \approx 0.99 \\
cos(\text{Doc}_3, q) &= \frac{\langle 2,3.32,0,0\rangle \cdot \langle 1,0,1.42,0\rangle}{|\ \langle 2,3.32,0,0\rangle\ |\ \ |\ \langle 1,0,1.42,0\rangle\ |} \\
&\approx \frac{2}{\sqrt{15}\sqrt{3.02}} \approx 0.30 \\
cos(\text{Doc}_4, q) &= \frac{\langle 1,2,1,3.81\rangle \cdot \langle 1,0,1.42,0\rangle}{|\ \langle 1,2,1,3.81\rangle\ |\ \ |\ \langle 1,0,1.42,0\rangle\ |} \\
&\approx \frac{2.42}{\sqrt{20.5}\sqrt{3.02}} \approx 0.31 \\
cos(\text{Doc}_5, q) &= \frac{\langle 1,1,2.58,0\rangle \cdot \langle 1,0,1.42,0\rangle}{|\ \langle 1,1,2.58,0\rangle\ |\ \ |\ \langle 1,0,1.42,0\rangle\ |} \\
&\approx \frac{4.66}{\sqrt{8.66}\sqrt{3.02}} \approx 0.91
\end{aligned}$$

- The best values for the cosine similarity measure are those closest to 1, because then the angle is small (close to 0), which means that the vectors point in the same direction, which means that the (weighted) distribution of terms in the document is similar to the distribution of terms in the query (which is what we want!)
- In this case, document 2 will be at the top of the ranking (with a score of 0.99), followed closely by 5, then 1, 4 and 3.

2.5.1.4 3

(b) If Documents 4 and 5 were the only **truly** relevant documents in the collection, calculate $P@1$, $P@3$, and $P@5$ for the above system.

- $P@k$ is a common shorthand for Precision at k; that is, we calculate Precision, based only on the top k returned results.
- For $P@1$, we observed that the system returned Document 2 at the top of the ranking. However, Document 2 wasn't actually relevant consequently, $P@1$ for this system is $\frac{0}{1} = 0$.
- For $P@3$, we observed that the system returned Documents 2, 5, and 1 as the top three in the ranking. Of these, Document 5 was relevant, but the others weren't consequently, $P@3$ for this system is $\frac{1}{3}$.
- For $P@5$, we observed that the system returned Documents 2, 5, 1, 4, and 3 as the top five in the ranking (the entire collection!). Of these, Documents 4 and 5 were relevant, but the others weren't consequently, $P@5$ for this system is $\frac{2}{5}$.

5. What does **Recall** correspond to, in an Information Retrieval context? Why is Recall not usually considered to be a useful evaluation metric for an IR Engine? How does this relate to $P@k$?

   - Recall is the proportion of relevant results returned by the system:

$$\text{Recall} = \frac{\text{Number of relevant results returned}}{\text{Total number of relevant results}}$$

   - To see the lack of utility of this metric, it is best to think about what is meant by "relevant", the size of a typical IR document collection, and the behaviour of users:
     - A document is relevant, if it meets the users information needs — that is to say, the document contains information that allows the user to solve their problem;
     - A typical IR collection contains **many** documents. This means that it is often the case that there are numerous documents which contain the required information — that is to say, there are many relevant documents;
     - A typical IR user is only able to look at a small number of documents (often, this means fewer than 10, but certainly not 1000s of documents, except in very specific domains).
   - The point here is that, for a ranked IR engine, as soon as a user has seen a single relevant document, they will typically stop examining the query results. Whether there are more relevant results further down the ranking has little bearing on the user's behaviour. (Or equivalently, if there are no relevant results in the first few documents, the user will usually re-formulate the query instead of reading through pages and pages of results).
   - $P@k$ incorporates these ideas into a more useful metric: given that a user will examine only the top k documents, what is the likelihood that they will find a relevant result there?