Information Retrieval SS 2024

Exercise 2: VSM, Efficient VSR, Probabilistic IR

Benedikt Ebing & Fabian David Schmidt
partially based on “An Introduction to Information Retrieval” by Manning, Raghavan and Schütze

Vector Space Model – TF-IDF

Task 1

Consider the following tables of term frequencies and document frequencies (df), and document collection sizes (N). Calculate for each term-document-pair the tf-idf weight with idf = ln(N/df), use raw term frequencies.

term	doc 1	doc 2	doc 3	df	N
car	27	4	24	18,136	94,584
auto	3	33	0	6,723	53,814
insurance	0	33	29	19,241	97,226
best	14	0	17	25,235	113,095

Answer:

		TF-IDF		
term	**IDF**	**doc 1**	**doc 2**	**doc 3**
car				
auto				
insurance				
best				

Task 1

Consider the following tables of term frequencies and document frequencies (df), and document collection sizes (N). Calculate for each term-document-pair the tf-idf weight with idf = ln(N/df), use raw term frequencies.

term	doc 1	doc 2	doc 3	df	N
car	27	4	24	18,136	94,584
auto	3	33	0	6,723	53,814
insurance	0	33	29	19,241	97,226
best	14	0	17	25,235	113,095

Answer:

		TF-IDF		
term	**IDF**	**doc 1**	**doc 2**	**doc 3**
car	1.65	44.55	6.6	39.6
auto	2.08	6.24	68.64	0
insurance	1.62	0	53.46	46.98
best	1.5	21	0	25.5

Task 2

In the previous subtask we used the natural logarithm. How does the base of the logarithm affect the tf-idf scores? How does the base of the logarithm affect the relative scores of two documents on a given query?

Task 2

In the previous subtask we used the natural logarithm. How does the base of the logarithm affect the tf-idf scores? How does the base of the logarithm affect the relative scores of two documents on a given query?

Rewrite the logarithm:

$$log_b\left(\frac{N}{df_t}\right) = log_b(10) \cdot log_{10}\left(\frac{N}{df_t}\right)$$
$$= c \cdot log_{10}\left(\frac{N}{df_t}\right)$$

Task 2

In the previous subtask we used the natural logarithm. How does the base of the logarithm affect the tf-idf scores? How does the base of the logarithm affect the relative scores of two documents on a given query?

Rewrite the logarithm:

$$log_b\left(\frac{N}{df_t}\right) = log_b(10) \cdot log_{10}\left(\frac{N}{df_t}\right)$$
$$= c \cdot log_{10}\left(\frac{N}{df_t}\right)$$

Factor out base into constant:

$$\text{tf-idf}_{t,d,b} = \text{tf}_{t,d} \cdot \text{idf}_{t,b}$$
$$= \text{tf}_{t,d} \cdot c \cdot \log_{10}\left(\frac{N}{df_t}\right)$$
$$= c \cdot \text{tf-idf}_{t,d}$$

Task 2

In the previous subtask we used the natural logarithm. How does the base of the logarithm affect the tf-idf scores? How does the base of the logarithm affect the relative scores of two documents on a given query?

Rewrite the logarithm:

$$log_b\left(\frac{N}{df_t}\right) = log_b(10) \cdot log_{10}\left(\frac{N}{df_t}\right) = c \cdot log_{10}\left(\frac{N}{df_t}\right)$$

Factor out base into constant:

$$\begin{aligned}\text{tf-idf}_{t,d,b} &= \text{tf}_{t,d} \cdot \text{idf}_{t,b} \\ &= \text{tf}_{t,d} \cdot c \cdot \log_{10}\left(\frac{N}{df_t}\right) \\ &= c \cdot \text{tf-idf}_{t,d}\end{aligned}$$

Pull out constant from the sum:

$$\text{Score}(q,d,b) = \sum_{t \in q} \text{tf-idf}_{t,d,q} = c \cdot \sum_{t \in q} \text{tf-idf}_{t,d}$$

Task 3

Recall the tf-idf weights computed in the first task. Compute the Euclidean normalized document vectors for each of the documents, where each vector has four components, one for each of the four terms.

Task 4

Using the document vectors you just constructed, rank the documents for the query “car insurance” according to their cosine distances to the query. Represent the query as a binary vector.

Task 3

Recall the tf-idf weights computed in the first task. Compute the Euclidean normalized document vectors for each of the documents, where each vector has four components, one for each of the four terms.

$$\vec{v}(doc1) = \begin{pmatrix} 0.897 \\ 0.126 \\ 0 \\ 0.423 \end{pmatrix}, \vec{v}(doc2) = \begin{pmatrix} 0.076 \\ 0.787 \\ 0.613 \\ 0 \end{pmatrix}, \vec{v}(doc3) = \begin{pmatrix} 0.595 \\ 0 \\ 0.706 \\ 0.389 \end{pmatrix}$$

vocabulary: [car, auto, insurance, best]

Task 4

Using the document vectors you just constructed, rank the documents for the query “car insurance” according to their cosine distances to the query. Represent the query as a binary vector.

Task 3

Recall the tf-idf weights computed in the first task. Compute the Euclidean normalized document vectors for each of the documents, where each vector has four components, one for each of the four terms.

$$\vec{v}(doc1) = \begin{pmatrix} 0.897 \\ 0.126 \\ 0 \\ 0.423 \end{pmatrix}, \vec{v}(doc2) = \begin{pmatrix} 0.076 \\ 0.787 \\ 0.613 \\ 0 \end{pmatrix}, \vec{v}(doc3) = \begin{pmatrix} 0.595 \\ 0 \\ 0.706 \\ 0.389 \end{pmatrix}$$

vocabulary: [car, auto, insurance, best]

Task 4

Using the document vectors you just constructed, rank the documents for the query “car insurance” according to their cosine distances to the query. Represent the query as a binary vector.

Query representation:

$$q = \text{"car insurance"} = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}^T$$

Document ranking:

$$\text{doc } 3 = cos(q, \text{doc } 3) = 0.920$$
$$\text{doc } 1 = cos(q, \text{doc } 1) = 0.634$$
$$\text{doc } 2 = cos(q, \text{doc } 2) = 0.483$$

Task 5

Compute the vector space similarity between the query **“digital cameras”** and the document **“digital cameras and video cameras”** by filling out the empty columns in the table below. Assume N=10,000,000, apply the term frequency scaling, as shown in the lecture, for query and document (wf columns). Do not account for query- or document-length. Apply idf weighting for the query and cosine normalization for the document. Treat ***and*** as a stop word. Enter term counts in the tf columns. What is the final similarity score?

	query					Document			
word	tf	wf	Df	idf	q_i=wf-idf	**Tf**	**wf**	**d_i**	**q_i * d_i**
digital	1	1	10,000	3	3	1	1	0.52	
video	0	0	100,000	2	0	1	1	0.52	
cameras	1	1	50,000	2.3	2.3	2	1.3	0.67	

wf = 1 + log10(tf)

Task 5

Compute the vector space similarity between the query **“digital cameras”** and the document **“digital cameras and video cameras”** by filling out the empty columns in the table below. Assume N=10,000,000, apply the term frequency scaling, as shown in the lecture, for query and document (wf columns). Do not account for query- or document-length. Apply idf weighting for the query and cosine normalization for the document. Treat *and* as a stop word. Enter term counts in the tf columns. What is the final similarity score?

	query					document			
word	tf	wf	df	idf	q_i=wf-idf	**tf**	**wf**	**d_i**	**q_i * d_i**
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56

Final similarity score:

$$\text{dot(query, document)} = 3.12$$
$$\text{cos(query, document)} = 0.826$$

Task 6

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

Task 7

Why is the idf of a term always finite?

Task 8

What is the minimum and maximum value of the idf that a term can have (assuming a document frequency larger than zero)? In which cases does this happen?

Task 6

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

It's zero. If a word occurs in every document then $N = df_t$ and $\frac{N}{df_t} = 1$ and $\text{idf}_t = \log(1) = 0$.

Adding the word to the stop word list has the same effect as idf weighting: the word is ignored.

Task 7

Why is the idf of a term always finite?

Task 8

What is the minimum and maximum value of the idf that a term can have (assuming a document frequency larger than zero)? In which cases does this happen?

Task 6

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

It's zero. If a word occurs in every document then $N = df_t$ and hence $\frac{N}{df_t} = 1$ and $\text{idf}_t = \log(1) = 0$.

Adding the word to the stop word list has the same effect as idf weighting: the word is ignored.

Task 7

Why is the idf of a term always finite?

$df_t \geq 1 \rightarrow idf_t \geq N \rightarrow$ idf always finite

Task 8

What is the minimum and maximum value of the idf that a term can have (assuming a document frequency larger than zero)? In which cases does this happen?

Task 6

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

It's zero. If a word occurs in every document then $N = df_t$ and hence $\frac{N}{df_t} = 1$ and $\text{idf}_t = \log(1) = 0$.

Adding the word to the stop word list has the same effect as idf weighting: the word is ignored.

Task 7

Why is the idf of a term always finite?

$df_t \geq 1 \rightarrow idf_t \geq N \rightarrow$ idf always finite

Task 8

What is the minimum and maximum value of the idf that a term can have (assuming a document frequency larger than zero)? In which cases does this happen?

Maximum value of idf_t for a term t is $\log(N)$ (term occurs in single document).

Minimum value of idf_t for a term t is 0 (term occurs in every document).

Optimizing Vector Space Retrieval

Task 1

Imagine your are running a rudimentary retrieval engine on a small embedded chip which can barely perform addition and multiplication. To that end, you want to minimize the number of cosine similarity computations but also the number of multiplications and additions in each individual cosine computation. You are given the following toy collection consisting of N=9 documents, with the following TF-IDF vectors (of length k=10):

$$
\begin{aligned}
d_1 &= [0.17 \quad 0.21 \quad 0.35 \quad 0.44 \quad 0.49 \quad 0.39 \quad 0.09 \quad 0.07 \quad 0.37 \quad 0.24] \\
d_2 &= [0.49 \quad 0.48 \quad 0.44 \quad 0.09 \quad 0.24 \quad 0.20 \quad 0.41 \quad 0.16 \quad 0.10 \quad 0.15] \\
d_3 &= [0.41 \quad 0.36 \quad 0.27 \quad 0.19 \quad 0.15 \quad 0.42 \quad 0.23 \quad 0.42 \quad 0.02 \quad 0.42] \\
d_4 &= [0.31 \quad 0.41 \quad 0.21 \quad 0.19 \quad 0.47 \quad 0.28 \quad 0.21 \quad 0.39 \quad 0.16 \quad 0.38] \\
&\dots
\end{aligned}
$$

In order to reduce the cost of computation of individual documents, we need to cut the length of document vectors in half. To achieve this, we perform random projections using the following five random vectors (from the same vector space as the original documents):

$$
\begin{aligned}
r_1 &= [0.33 \quad 0.33 \quad 0.42 \quad 0.12 \quad 0.20 \quad 0.34 \quad 0.58 \quad 0.19 \quad 0.07 \quad 0.24] \\
r_2 &= [0.29 \quad 0.16 \quad 0.38 \quad 0.48 \quad 0.43 \quad 0.11 \quad 0.12 \quad 0.33 \quad 0.03 \quad 0.44] \\
r_3 &= [0.01 \quad 0.17 \quad 0.11 \quad 0.27 \quad 0.23 \quad 0.37 \quad 0.35 \quad 0.48 \quad 0.54 \quad 0.24] \\
r_4 &= [0.09 \quad 0.05 \quad 0.39 \quad 0.25 \quad 0.45 \quad 0.48 \quad 0.04 \quad 0.45 \quad 0.35 \quad 0.12] \\
r_5 &= [0.13 \quad 0.17 \quad 0.40 \quad 0.40 \quad 0.07 \quad 0.40 \quad 0.35 \quad 0.39 \quad 0.44 \quad 0.06]
\end{aligned}
$$

$$
h(d_i, r_j) = \begin{cases} 1, & \text{if } d_i \cdot r_j > 0.75 \\ 0, & \text{otherwise} \end{cases}
$$

Task 1 (a)

Compute the hashed (i.e., projected, shortened) document vectors.

Task 1 (a)

Compute the hashed (i.e., projected, shortened) document vectors.

Distance matrix (#docs x #randVecs):

$$D = \begin{pmatrix} 0.7048 & 0.8313 & 0.7742 & 0.8734 & 0.8001 \\ 0.9429 & 0.7255 & 0.5987 & 0.6076 & 0.7130 \\ 0.8785 & 0.8326 & 0.7299 & 0.7324 & 0.7553 \\ 0.8361 & 0.8853 & 0.7972 & 0.8092 & 0.7337 \\ 0.8806 & 0.7596 & 0.816 & 0.7924 & 0.8383 \\ 0.8605 & 0.7776 & 0.6584 & 0.4858 & 0.7008 \\ 0.6898 & 0.6119 & 0.9344 & 0.8742 & 0.8323 \\ 0.9119 & 0.7589 & 0.6655 & 0.6139 & 0.6596 \\ 0.7513 & 0.6484 & 0.8000 & 0.6060 & 0.7103 \end{pmatrix}$$

Hashed document vectors:

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$R = [r_1; r_2; r_3; r_4] \in \mathbb{R}^{5 \times 10}$$

$$\hat{d_1} = hash(d_1 R^T) = H_{1,\cdot} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Task 1 (b)

We also need to reduce the total number of cosine comparisons, so we will perform pre-clustering of documents. In the reduced projected vector space, obtained using the random projections from the previous task, you are given the following $\sqrt{N} = 3$ leader vectors that will determine the document clusters:

$$l_1 = [0 \quad 0 \quad 1 \quad 1 \quad 0]$$
$$l_2 = [0 \quad 1 \quad 1 \quad 1 \quad 0]$$
$$l_3 = [0 \quad 0 \quad 0 \quad 0 \quad 0]$$

Compute the clusters by assigning each document vector (i.e., it's random projection) to the closest (i.e. most similar) leader vector.

Clusters:

$l_1 : d7, d9$
$l_2 : d1, d4, d5$
$l_3 : d2, d3, d6, d8$

Cluster assignment for document 1:

$$sim(d_i, l_j) = \#\text{matching bits}$$

$$leader(d_1) = \text{argmax}_{l \in \{l_1, l_2, l_3\}} = \{sim(\hat{d_1}, l_1\} = 3, \; sim(\hat{d_1}, l_2) = 4, \; sim(\hat{d_1}, l_3) = 1\}$$

Task 2

You are given the following query vector (in the original vector space):

$$q = [0.15 \quad 0.39 \quad 0.36 \quad 0.25 \quad 0.36 \quad 0.15 \quad 0.52 \quad 0.37 \quad 0.08 \quad 0.27]$$

Retrieve the top $M = 5$ documents using the random projection vectors of documents from 1 (a) and clusters obtained in 1 (b). What is the total number of cosine similarities you compute and the total number of element-wise multiplication operations in all dot-products?

Projected query:

$$\hat{q} = [1 \quad 1 \quad 1 \quad 0 \quad 1]$$

Leader similarities:

$$sim(\hat{q}, l_1) = 1$$
$$sim(\hat{q}, l_2) = 2$$
$$sim(\hat{q}, l_3) = 1$$

Ranking (clusters):

1. d3
2. d5
3. d1
4. d4
5. d6

Ranking (cosine):

1. d_8
2. d_4
3. d_6
4. d_2
5. d_3

Task 2 (cont.)

Compare that with the numbers you would get if we didn't perform any pre-clustering nor random projections.

Regular retrieval:

- **9** similarity computations (1 query, 9 docs) and **10*9=90** element-wise operations.

Cluster retrieval (Offline / Indexing):

- Document projections **9*5=45** dot products (**45*10** element-wise operations)
- Cluster assignment **9*3=27** similarity computations / “dot-products” (**27*5=135**)
- Total: **72** dot products (**585** element-wise operations)

Cluster retrieval (Online / Querying):

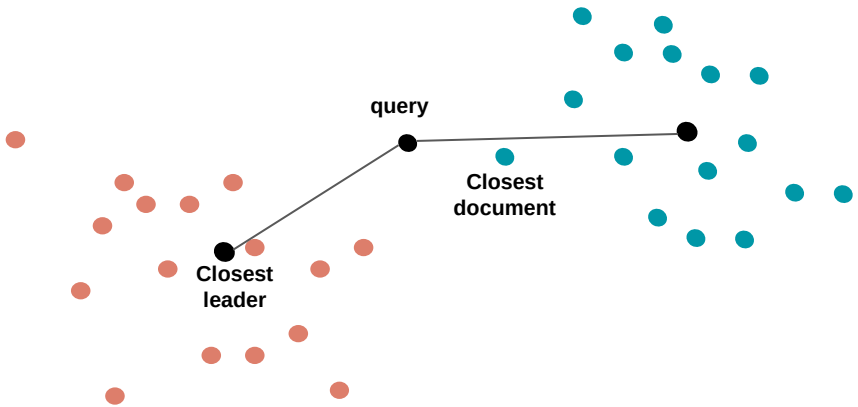
- Query projections → **5** similarity computations / “dot-products” (**5*10=50** element-wise operations)
- Comparsion with cluster leader → **3** similarity computations / “dot-products” (**3*5=15**)
- Querying **7** similarity computations/“dot-products” (**5*7=35** element-wise operations)
- Total: **15** dot products (**100** element-wise operations)

Task 3

In the previous two tasks we used cluster pruning to avoid computing the distance from the query vector to every document vector. Sketch down an example so that with two leaders, the answer returned by cluster pruning is incorrect (it is not the data point closest to the query vector).

Task 3

In the previous two tasks we used cluster pruning to avoid computing the distance from the query vector to every document vector. Sketch down an example so that with two leaders, the answer returned by cluster pruning is incorrect (it is not the data point closest to the query vector).



The selected leader returns a ranking in which the closest document is not included.

Probabilistic Information Retrieval

Task 1 (a)

Consider deriving a language model from the following piece of text:

```
the martian has landed on the latin pop sensation ricky martin
```

Compute the probabilities $\mathrm{P}(the)$ and $\mathrm{P}(martian)$ in the an MLE-estimated unigram probability model.

Task 1 (b)

Compute the probabilities $\mathrm{P}(pop|the)$ and $\mathrm{P}(sensation|pop)$ under an MLE-estimated bigram probability model.

Task 1 (a)

Consider deriving a language model from the following piece of text:

```
the martian has landed on the latin pop sensation ricky martin
```

Compute the probabilities $\mathrm{P}(the)$ and $\mathrm{P}(martian)$ in the an MLE-eistmated unigram probability model.

$$\mathrm{P}(the) = \frac{2}{11} = 18.18\% \qquad \mathrm{P}(martian) = \frac{1}{11} = 9.09\%$$

Task 1 (b)

Compute the probabilities $\mathrm{P}(pop|the)$ and $\mathrm{P}(sensation|pop)$ under an MLE-estimated bigram probability model.

Task 1 (a)

Consider deriving a language model from the following piece of text:

```
the martian has landed on the latin pop sensation ricky martin
```

Compute the probabilities $\mathrm{P}(the)$ and $\mathrm{P}(martian)$ in the an MLE-eistmated unigram probability model.

$$\mathrm{P}(the) = \frac{2}{11} = 18.18\% \qquad \mathrm{P}(martian) = \frac{1}{11} = 9.09\%$$

Task 1 (b)

Compute the probabilities $\mathrm{P}(pop|the)$ and $\mathrm{P}(sensation|pop)$ under an MLE-estimated bigram probability model.

$$\mathrm{P}(pop|the) = \frac{count(the,pop)}{count(the)} = 0$$

$$\mathrm{P}(sensation|pop) = \frac{count(pop,sensation)}{count(pop)} = 1$$

Task 2

Suppose we have a collection *D* that consists of the four documents given in the following table.

docID	text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

Build a query likelihood language model for this document collection. Use the Jelinek-Mercer smoothing model with the interpolation of λ = 0.5 and apply maximum likelihood estimation (MLE) to estimate unigram probabilities. Work out the model probabilities of the queries `q1=`**`click`**, `q2=`**`shears`**, and `q3=`**`click shears`** for each document, and use those probabilities to rank the documents returned by each of the queries.

$$P(t_i|M_d) = \lambda \cdot P(t_i|M_d) + (1 - \lambda) \cdot P(t_i|M_D)$$

query	doc 1	doc 2	doc 3	doc 4	ranking
click					
shears					
click shears					

Task 2

Suppose we have a collection *D* that consists of the four documents given in the following table.

docID	text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

Build a query likelihood language model for this document collection. Use the Jelinek-Mercer smoothing model with the interpolation of λ = 0.5 and apply maximum likelihood estimation (MLE) to estimate unigram probabilities. Work out the model probabilities of the queries `q1=`**`click`**, `q2=`**`shears`**, and `q3=`**`click shears`** for each document, and use those probabilities to rank the documents returned by each of the queries.

query	doc 1	doc 2	doc 3	doc 4	ranking
click	0.47	0.72	0.22	0.34	2, 1, 4, 3
shears	0.13	0.06	0.06	0.19	4, 1, 2, 3
click shears	0.06	0.04	0.01	0.06	{1, 4}, 2, 3

Task 3

Re-compute the rankings while using the (i) Binary Independence Model, (ii) the TwoPoisson model with k = 1 and (iii) the BM25 model with b = 0.75.

(i) Binary Independence Model: $P(D,Q,r) = \sum_{t \in Q} log\left(\frac{P(D_t|Q,r)}{P(D_t|Q,\bar{r})}\right) = \sum_{t \in Q} log\left(0.5 \cdot \frac{N}{N_t}\right)$

t=click	doc 1	doc 2	doc 3	doc 4
$P(D_t\|q,r)$	0.5	0.5	-	0.5
$P(D_t\|q,\neg r)$				
$\sum w_t$				

t=shears	doc 1	doc 2	doc 3	doc 4
$P(D_t\|q,r)$	0.5	-	-	0.5
$P(D_t\|q,\neg r)$				
$\sum w_t$				

Task 3

Re-compute the rankings while using the (i) Binary Independence Model, (ii) the TwoPoisson model with k = 1 and (iii) the BM25 model with b = 0.75.

(i) Binary Independence Model: $P(D,Q,r) = \sum_{t \in Q} log\left(\frac{P(D_t|Q,r)}{P(D_t|Q,\bar{r})}\right) = \sum_{t \in Q} log\left(0.5 \cdot \frac{N}{N_t}\right)$

t=click	doc 1	doc 2	doc 3	doc 4
$P(D_t\|q,r)$	0.5	0.5	-	0.5
$P(D_t\|q,\neg r)$	4/3	4/3	-	4/3
$\sum w_t$	-0.405	-0.405	-	-0.405

Ranking: `click`

1. doc 1 / doc 2 / doc 4
2. doc 3

t=shears	doc 1	doc 2	doc 3	doc 4
$P(D_t\|q,r)$	0.5	-	-	0.5
$P(D_t\|q,\neg r)$	4/2	-	-	4/2
$\sum w_t$	0	-	-	0

Ranking: `shears`

1. doc 1 / doc 4
2. doc 2 / doc 3

	doc 1		doc 2	doc 3	doc 4	
T	click	shears	click	-	click	shears
$P(D_t \mid q, r)$	0.5	0.5	0.5	-	0.5	0.5
$P(D_t \mid q, \neg r)$	4/3	4/2	4/3	-	4/3	4/2
w_t	-0.405	0	-0.405	-	-0.405	0
$\sum w_t$	-0.405		-0.405	-	-0.405	

Same as for click

(ii) Two-Poisson model: $rel(D,Q) = \sum_{t \in Q} \frac{f_{t,D}(k+1)}{f_{t,D}+k} \cdot w_t$

t=click	doc 1	doc 2	doc 3	doc 4
$\frac{f_{t,D}(k+1)}{f_{t,D}+k}$				
w_t				
$rel(D,q)$				

	doc 1		doc 2	doc 3	doc 4	
T	click	shears	click	-	click	shears
$P(D_t \mid q, r)$	0.5	0.5	0.5	-	0.5	0.5
$P(D_t \mid q, \neg r)$	4/3	4/2	4/3	-	4/3	4/2
w_t	-0.405	0	-0.405	-	-0.405	0
$\sum w_t$	-0.405		-0.405	-	-0.405	

Ranking: `click shears`

1. doc 1 / doc 2 / doc 4
2. doc 3

(ii) Two-Poisson model: $rel(D, Q) = \sum_{t \in Q} \frac{f_{t,D}(k+1)}{f_{t,D}+k} \cdot w_t$

t=click	doc 1	doc 2	doc 3	doc 4
$\frac{f_{t,D}(k+1)}{f_{t,D}+k}$			-	
w_t			-	
$rel(D, q)$			-	

	doc 1		doc 2	doc 3	doc 4	
T	click	shears	click	-	click	shears
$P(D_t \mid q, r)$	0.5	0.5	0.5	-	0.5	0.5
$P(D_t \mid q, \neg r)$	4/3	4/2	4/3	-	4/3	4/2
w_t	-0.405	0	-0.405	-	-0.405	0
$\sum w_t$	-0.405		-0.405	-	-0.405	

Ranking: `click shears`

1. doc 1 / doc 2 / doc 4
2. doc 3

(ii) Two-Poisson model: $rel(D, Q) = \sum_{t \in Q} \frac{f_{t,D}(k+1)}{f_{t,D}+k} \cdot w_t$

t=click	doc 1	doc 2	doc 3	doc 4
$\frac{f_{t,D}(k+1)}{f_{t,D}+k}$	1.600	1.330	-	1.000
w_t	-0.405	-0.405	-	-0.405
$rel(D, q)$	-0.648	-0.539	-	-0.405

Ranking: `click`

1. doc 4
2. doc 2
3. doc 1
4. doc 3

Ranking for `shears` remains unchanged because of zero weights.
Ranking for `click shears` the same like for click.

(iii) BM-25: $$rel(D,Q) = \sum_{t \in Q} \frac{f_{t,D}(k+1)}{f_{t,D} + k\frac{l_d}{l_{avg}}b + k(1-b)} \cdot w_t$$

click	doc 1	doc 2	doc 3	doc 4
$\frac{f_{t,D}(k+1)}{f_{t,D} + k\frac{l_d}{l_{avg}}b + k(1-b)}$	1.391			
w_t	-0.405			
$rel(D,q)$	-0.563			

Task 4

Does the Binary Independence Model make the document independence assumption? Does it make the term independence assumption? Explain your answer.

(iii) BM-25: $$rel(D,Q) = \sum_{t \in Q} \frac{f_{t,D}(k+1)}{f_{t,D} + k\frac{l_d}{l_{avg}}b + k(1-b)} \cdot w_t$$

t=click	doc 1	doc 2	doc 3	doc 4
$\frac{f_{t,D}(k+1)}{f_{t,D} + k\frac{l_d}{l_{avg}}b + k(1-b)}$	1.391	1.523	-	1.000
w_t	-0.405	-0.405	-	-0.405
$rel(D,q)$	-0.563	-0.617	-	-0.405

Ranking: `click` $l_{avg} = 4$

1. doc 4
2. doc 1
3. doc 2
4. doc 3

Ranking for `shears` remains unchanged because of zero weights.
Ranking for `click shears` the same like for click.

Task 4

Does the Binary Independence Model make the document independence assumption? Does it make the term independence assumption? Explain your answer.

(iii) BM-25: $$rel(D,Q) = \sum_{t \in Q} \frac{f_{t,D}(k+1)}{f_{t,D} + k\frac{l_d}{l_{avg}}b + k(1-b)} \cdot w_t$$

	doc 1	doc 2	doc 3	doc 4
$\frac{f_{t,D}(k+1)}{f_{t,D} + k\frac{l_d}{l_{avg}}b + k(1-b)}$	1.391	1.523	-	1.000
w_t	-0.405	-0.405	-	-0.405
$rel(D,q)$	-0.563	-0.617	-	-0.405

Ranking: `click` $l_{avg} = 4$

1. doc 4
2. doc 1
3. doc 2
4. doc 3

Ranking for `shears` remains unchanged because of zero weights.
Ranking for `click shears` the same like for click.

Task 4

Does the Binary Independence Model make the document independence assumption? Does it make the term independence assumption? Explain your answer.

Document independence: Yes, each documents similarity score is calculated independently.

Term independence: Yes, because it factor $\log\left(\frac{P(D|Q,r)}{P(D|Q,\bar{r})}\right)$ into $\log\left(\Pi_{i=1}^{N}\frac{P(D_i|Q,r)}{P(D_i|Q,\bar{r})}\right)$.

Task 5

Why is the (conditional) independence assumption in language models an incorrect assumption that doesn't hold when dealing with natural language? Give an example.

Task 6

What are the differences between standard vector space tf-idf weighting and the BIM probabilistic retrieval model (in the case where no document relevance information is available)?

Task 5

Why is the (conditional) independence assumption in language models an incorrect assumption that doesn't hold when dealing with natural language? Give an example.

Because natural language follow rules about structure and meaning (Syntax and Semantic). For example, after an article usually a noun follows.

Task 6

What are the differences between standard vector space tf-idf weighting and the BIM probabilistic retrieval model (in the case where no document relevance information is available)?

Task 5

Why is the (conditional) independence assumption in language models an incorrect assumption that doesn't hold when dealing with natural language? Give an example.

Because natural language follow rules about structure and meaning (Syntax and Semantic). For example, after an article usually a noun follows.

Task 6

What are the differences between standard vector space tf-idf weighting and the BIM probabilistic retrieval model (in the case where no document relevance information is available)?

TF-IDF weighting is proportional to term frequency of the query term while BIM takes into account only presence/absence of a term.