Seminar 1

Exercise 1/1

Recommend a query processing strategy for (*tangerine* OR *trees*) AND (*marmalade* OR *skies*) AND (*kaleidoscope* OR *eyes*) with respect to the following postings list sizes:

eyes 213312

kaleidoscope 87009

marmalade 107913

skies 271658

tangerine 46653

trees 316812

We use a database trick where we filter out the results with the clause of the shortest intermediate result first. Operations OR is understood as addition and AND as multiplication. Compose the equations:

$$\textit{tangerine} \text{ OR } \textit{trees} = 46653 + 316812 = 363465$$

$$\textit{marmalade} \text{ OR } \textit{skies} = 107913 + 271658 = 379571$$

$$\textit{kaleidoscope} \text{ OR } \textit{eyes} = 87009 + 213312 = 300321$$

After sorting these with respect to sizes and we get the ordering

$$\textit{kaleidoscope} \text{ OR } \textit{eyes} < \textit{tangerine} \text{ OR } \textit{trees} < \textit{marmalade} \text{ OR } \textit{skies}$$

we see that the query is best processed in the following sequence:

1. $a =$ *kaleidoscope* OR *eyes*
2. $b =$ *tangerine* OR *trees*
3. $c =$ *marmalade* OR *skies*
4. $d = a$ AND b
5. $e = d$ AND c

Exercise 1/2

What is the best order for processing the query *ostrich* AND *hippo* AND *giraffe* if we know that the number of occurrences of the animals are 100, 500, 300, respectively?

(*ostrich* AND *giraffe*) AND *hippo*

Exercise 1/3

Create an inverted index composed of the following collection of documents:

Doc 1: new home sales top forecasts

Doc 2: home sales rise in July

Doc 3: increase in home sales in July

Doc 4: July new home sales rise

Very easy procedure. Start with an empty table. If the term already appears in the table as a key, add the document ID only. Otherwise, take each term of a document and add it as a key to the table with the ID of the document. This way we get the inverted index represented in the following table.

new	1	4		
home	1	2	3	4
sales	1	2	3	4
top	1			
forecasts	1			
rise	2	4		
in	2	3		
July	2	3	4	
increase	3			

Table 1: Inverted index

Exercise 1/4

Create an inverted index composed of the following collection of documents:

Doc 1: hippo ostrich ostrich giraffe

Doc 2: lion frog giraffe hippo

Doc 3: ostrich frog bat giraffe lion frog

---

hippo	1	2	
ostrich	1	3	
giraffe	1	2	3
lion	2	3	
frog	2	3	
bat	3		

Table 2: Inverted index

Seminar 2

Algorithm 1 (Soundex Code)
Transformation of a string to a 4-character soundex code

1. *Keep the first character*
2. *Rewrite* $\{A, E, I, O, U, H, W, Y\}$ *to 0*
3. *Rewrite characters*
   (a) $\{B, F, P, V\}$ *to 1*
   (b) $\{C, G, J, K, Q, S, X, Z\}$ *to 2*
   (c) $\{D, T\}$ *to 3*
   (d) $\{L\}$ *to 4*
   (e) $\{M, N\}$ *to 5*
   (f) $\{R\}$ *to 6*
4. *Remove duplicities*
5. *Remove zeros*
6. *Change to length 4 (truncate or add zeros)*

Algorithm 2 (Querying in Permuterm Index)
For query q, find keys according to the following scheme:

- *for* $q = X$*, find keys in the form* $X\$$
- *for* $q = X^*$*, find keys in the form* $\$X^*$
- *for* $q = {}^*X$*, find keys in the form* $X\$^*$
- *for* $q = {}^*X^*$*, find keys in the form* X^*
- *for* $q = X^*Y$*, find keys in the form* $Y\$X^*$

Exercise 2/1

Below is a part of index with positions in the form
doc1: $\langle pos1, pos2, pos3, \ldots\rangle$; doc2: $\langle pos1, pos2, \ldots\rangle$; ...

- angels: $2 : \langle 36, 174, 252, 651\rangle$; $4 : \langle 12, 22, 102, 432\rangle$; $7 : \langle 17\rangle$;
- fools: $2 : \langle 1, 17, 74, 222\rangle$; $4 : \langle 8, 78, 108, 458\rangle$; $7 : \langle 3, 13, 23, 193\rangle$;
- fear: $2 : \langle 87, 704, 722, 901\rangle$; $4 : \langle 13, 43, 113, 433\rangle$; $7 : \langle 18, 328, 528\rangle$;
- in: $2 : \langle 3, 37, 76, 444, 851\rangle$; $4 : \langle 10, 20, 110, 470, 500\rangle$; $7 : \langle 5, 15, 25, 195\rangle$;
- rush: $2 : \langle 2, 66, 194, 321, 702\rangle$; $4 : \langle 9, 69, 149, 429, 569\rangle$; $7 : \langle 4, 14, 404\rangle$;
- to: $2 : \langle 47, 86, 234, 999\rangle$; $4 : \langle 14, 24, 774, 944\rangle$; $7 : \langle 19, 319, 599, 709\rangle$;

- tread: $2 : \langle 57, 94, 333 \rangle$; $4 : \langle 15, 35, 155 \rangle$; $7 : \langle 20, 320 \rangle$;
- where: $2 : \langle 67, 124, 393, 1001 \rangle$; $4 : \langle 11, 41, 101, 421, 431 \rangle$; $7 : \langle 15, 35, 735 \rangle$;

The following terms are phrase queries. Which documents correspond to the following queries and on which positions?

a) *fools rush in*

b) *fools rush in* AND *angels fear to tread.*

The index is incorrect. How?

In order to retrieve the query it is necessary that the words are in a sequence. That is, if the word *angels* is in **doc2** on position 36, then the word *fear* has to be in the same document on the position 37 and so on.

For the exercise **a)** we calculate all possible positions of the phrase. Word *fools* appears in **doc2** on positions $\langle 1, 17, 74, 222 \rangle$. That means that the word *rush* has to appear on positions $\langle 2, 18, 75, 223 \rangle$ and the word *in* on positions $\langle 3, 19, 76, 224 \rangle$. Similar process is applied on **doc4** and **doc7** which retrieves the requested results.

doc2 $\langle 1, 2, 3 \rangle$, $\langle 17, 18, 19 \rangle$, $\langle 74, 75, 76 \rangle$, $\langle 222, 223, 224 \rangle$

doc4 $\langle 8, 9, 10 \rangle$, $\langle 78, 79, 80 \rangle$, $\langle 108, 109, 110 \rangle$, $\langle 458, 459, 460 \rangle$

doc7 $\langle 3, 4, 5 \rangle$, $\langle 13, 14, 15 \rangle$, $\langle 23, 24, 25 \rangle$, $\langle 193, 194, 195 \rangle$

Now we look at the original position index and search for whether there is a conjunction between requested and real positions. Take **doc2** and check whether the words *fools*, *rush* and *in* are in a sequence on positions $\langle 1, 2, 3 \rangle$. Since yes, the system returns **doc2** as relevant to our query. Same analogy is used for the remaining documents for which we get the result **doc2**: $\{\langle 1, 2, 3 \rangle\}$; **doc4**: $\{\langle 8, 9, 10 \rangle\}$; and **doc7**: $\{\langle 3, 4, 5 \rangle, \langle 13, 14, 15 \rangle\}$.

For the exercise **b)** we find the requested positions for also the term *angels fear to tread.*

doc2 $\langle 36, 37, 38, 39 \rangle$, $\langle 174, 175, 176, 177 \rangle$, $\langle 252, 253, 254, 255 \rangle$, $\langle 651, 652, 653, 654 \rangle$

doc4 $\langle 12, 13, 14, 15 \rangle$, $\langle 22, 23, 24, 25 \rangle$, $\langle 102, 103, 104, 105 \rangle$, $\langle 432, 433, 434, 435 \rangle$

doc7 $\langle 17, 18, 19, 20 \rangle$

They appear in the correct order in **doc4**: $\{\langle 12, 13, 14, 15 \rangle\}$ and in **doc7**: $\{\langle 17, 18, 19, 20 \rangle\}$. Taking the first part from **a)**, we only check whether the results overlap {**doc2**(1), **doc4**(8), **doc7**(3), **doc7**(13)} $\cap$ {**doc4**(12), **doc7**(17)} = {**doc4**(8,12), **doc7**(3,17), **doc7**(13,17)}.

The index is incorrect. We need to have a look into **doc7**, where on position 15 are two terms *in* and *where*.

Exercise 2/2

Below is a part of index with positions in the form
doc1: $\langle pos1, pos2, pos3, \ldots \rangle$; doc2: $\langle pos1, pos2, \ldots \rangle$; ...

- ostrich: 1 : <1,7>; 2 : <4,5>;

- hippo: 1 : <5,8,9>; 3 : <6,9>;
- lion: 1 : <3,6>; 2 : <3,7>;
- giraffe: 1 : <2,4>; 2 : <1,2,8>;

Which documents correspond to the phrase query *lion giraffe hippo* and on which positions? Include intermediate results.

Candidates:

doc1 $\langle 3, 4, 5\rangle$, $\langle 6, 7, 8\rangle$

doc2 $\langle 3, 4, 5\rangle$, $\langle 7, 8, 9\rangle$

Result:

doc1 $\langle 3, 4, 5\rangle$

Exercise 2/3

Consider a query composed of two terms. Non-positional postings list of one term is composed of 16 items $P = [4, 6, 10, 12, 14, 16, 18, 20, 22, 32, 47, 81, 120, 215, 300, 500]$ and the second term has the postings list of only a single element $R = [47]$. Find out how many comparisons (and why) are necessary to find out the intersection of the lists that are organized as follows:

a) standard postings lists

b) postings lists with skip pointers of skip frequency $\sqrt{|P|}$

---

a) A naive algorithm compares each element from P with each element from R, which is 11.

b) With skip pointers of frequency $\sqrt{|P|} = 4$ we reduce the number of comparisons. Instead of the next element $i + 1$ only, the pointer goes directly to $i + \sqrt{|P|}$ until the referred value is larger than the searched value, after which it jumps back and then step forward by 1. Starting at position 0, the algorithm proceeds as follows: compare 4:47, jump to 14, compare 14:47, jump to 22, compare 22:47, jump to 120, compare 120:47, jump back to 22, step to 32, compare 32:47, step to 47, compare 47:47, done. The number of compare operations is 6.

Exercise 2/4

Consider a query composed of two terms. Non-positional postings list with skip pointers of one term is composed of 16 items $P_1 = [4, 6, 10, 12, 14, 16, 18, 20, 22, 32, 47, 81, 120, 215, 300, 500]$ with skip frequency of square root of its length and the second term has the standard postings list $P_2 = [18, 32, 60]$. How many comparisons are necessary to find out the intersection of the lists?

$(4, 18)$, $(14, 18)$, $(22, 18)$, $(16, 18)$, $(18, 18)$, $(20, 32)$, $(22, 32)$, $(120, 32)$, $(32, 32)$, $(47, 60)$, $(81, 60)$.

Exercise 2/5

List the comparisons performed to intersect the following sorted non-positional postings lists with skip pointers of frequency 5.

$$P_1 = [2, 10, 12, 16] \quad \text{and} \quad P_2 = [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$$

$(2, 1)$, $(2, 7)$, $(2, 3)$, $(10, 3)$, $(10, 4)$, $(10, 5)$, $(10, 6)$, $(10, 7)$, $(10, 12)$, $(10, 8)$, $(10, 9)$, $(10, 10)$, $(12, 11)$, $(12, 12)$, $(16, 13)$, $(16, 14)$, $(16, 15)$.

Exercise 2/6

List the comparisons performed to intersect the following sorted non-positional postings lists with skip pointers of frequency 5.

$$P_1 = [4, 5, 6, 7, 8, 9, 10, 13, 14, 15] \quad \text{and} \quad P_2 = [1, 2, 3, 4, 5, 10, 11, 15, 16]$$

$(4, 1)$, $(4, 10)$, $(4, 2)$, $(4, 3)$, $(4, 4)$, $(5, 5)$, $(6, 10)$, $(7, 10)$, $(8, 10)$, $(9, 10)$, $(10, 10)$, $(13, 11)$, $(13, 15)$, $(14, 15)$, $(15, 15)$.

Exercise 2/7

a) Find two different words of the same soundex code.

b) Find two phonetically similar words of different soundex codes.

---

a) *sword* and *short* have codes S630

b) *fog* and *thug* have codes F200 and T200

Exercise 2/8

Write elements in a dictionary of the permuterm index generated by the term *mama*.

---

mama\$, *ama*\$*m*, *ma*\$*ma*, *a*\$*mam*, \$*mama*.

Exercise 2/9

Which keys are usable for finding the term *s*ng* in a permuterm wildcard index?

$ng\$s^*$

Exercise 2/10

What is the complexity of intersection of two un-ordered posting lists of lengths m and n?

$\mathcal{O}(m \log m + n \log n)$

Exercise 2/11

What is the complexity (in $\mathcal{O}$-notation) of intersecting of two ordered posting lists of lengths m and n?

$\mathcal{O}(m + n)$

Exercise 2/12

What is the worst-case complexity of searching in hash tables?

Linear.

Seminar 3

Algorithm 3 (Variable byte code)
A number n is encoded in variable byte code in the following procedure:

1. *Take a binary representation of n with padding to the length of a multiple of 7.*
2. *Split into of 7 bit blocks right-to-left.*
3. *Add 1 to the beginning of the last block and 0 to the beginning of all previous blocks.*

Example: $VB(824) = 00000110\,10111000$

Definition 1 (Unary code)
Unary code, also referred to as α code, is a coding type where a number n is represented by a sequence of n 1s (or 0s) and terminated with one 0 (or 1). That is, 6 in unary code is 1111110 (or 0000001). The alternative representation in parentheses is equivalent but for this course we use the default representation.

Definition 2 (γ code)
γ code is a coding type, that consists of an offset and its length: $\gamma(n)$ = length of offset(n) in α,offset(n). Offset is a binary representation of a number n without the highest bit (1). Length of this offset encoded in the unary (α) code. Then the number 60 is encoded in γ as 111110,11100.

Definition 3 (δ code)
A number n is encoded in δ code in the following way. First calculate the offset of n and the length of n encode with γ code. Then add the offset of n. The final form is $\delta(n)$ = length of offset(n) in γ,offset(n). Analogously, 600 is encoded in δ as 1110,001,001011000.

Definition 4 (Zipf's law)
Zipf's law says that the i-th most frequent term has the frequency $\frac{1}{i}$. In this exercise we use the dependence of the Zipf's law $cf_i \propto \frac{1}{i} = ci^k$ where cf_i is the number of terms t_i in a given collection with $k = -1$.

Definition 5 (Heaps' law)
Heaps' law expresses an empiric dependency of collection size (number of all words) T and vocabulary size (number of distinct words) M by $M = kT^b$ where $30 \leq k \leq 100$ and $b \approx \frac{1}{2}$.

Exercise 3/1

Count variable byte code for the postings list $\langle 777, 17\,743, 294\,068, 31\,251\,336\rangle$. Bear in mind that the gaps are encoded. Write in 8-bit blocks.

Encode the list of gaps $\langle 777, 16\,966, 276\,325, 30\,957\,268\rangle$. Variable byte code of the gaps:

- $VB(777) = \mathbf{0}0000110\,\mathbf{1}0001001$
- $VB(16\,966) = \mathbf{0}0000001\,\mathbf{0}0000100\,\mathbf{1}1000110$
- $VB(276\,325) = \mathbf{0}0010000\,\mathbf{0}1101110\,\mathbf{1}1100101$
- $VB(30\,957\,268) = \mathbf{0}0001110\,\mathbf{0}1100001\,\mathbf{0}0111101\,\mathbf{1}1010100$

Result: $VB(\langle 777, 17\,743, 294\,068, 31\,251\,336\rangle) =$ 00000110 10001001 00000001 00000100 11000110 00010000 01101110 11100101 00001110 01100001 00111101 11010100

Exercise 3/2

Count γ and δ codes for the numbers 63 and 1023.

According to the definition 2 it is necessary to count the offsets as binary representations without the highest bit $63_{10} = 111111_2$ and offset(63) = 11111. Offset length is encoded in α as $|11111| = 5 \rightsquigarrow \alpha(5) = 111110$. Finally, $\gamma(63) = 111110, 11111$. Analogically for 1023. $1023_{10} = 1111111111_2$, offset is 111111111, its length is $|111111111| = 9 \rightsquigarrow \alpha(9) = 1111111110$. Then $\gamma(1023) = 1111111110, 111111111$.

δ is a little more complicated. First we count the offset $63 = 11111$ and its length $|11111| = 5$. The value of 5 we encode in γ so $\gamma(5) = 110, 01$. By definition 3 we have $\delta(63) = 110, 01, 11111$. And finally, $\delta(1023) = 1110, 010, 111111111$.

Exercise 3/3

Calculate the variable byte code, γ code and δ code of the postings list $P = [32, 160, 162]$. Note that gaps are encoded. Include intermediate results (offsets, lengths).

---

offset $32 = 0000$ and $\alpha(|0000|) = 11110 \rightsquigarrow \gamma(32) = 11110, 0000$
offset $128 = 0000000$ and $\alpha(|0000000|) = 11111110 \rightsquigarrow \gamma(128) = 11111110, 0000000$
offset $2 = 0$ and $\alpha(|0|) = 10 \rightsquigarrow \gamma(2) = 10, 0$
$\gamma(P) = 11110000011111100000000100$

offset $32 = 0000$ and $\gamma(|0000|) = 110, 00 \rightsquigarrow \delta(32) = 110, 00, 0000$
offset $128 = 0000000$ and $\gamma(|0000000|) = 110, 11 \rightsquigarrow \delta(128) = 110, 11, 0000000$
offset $2 = 0$ and $\gamma(|0|) = 0 \rightsquigarrow \delta(2) = 0, , 0$
$\delta(P) = 11000000011011000000000$

Exercise 3/4

Consider a posting list with the following list of gaps

$$G = [4, 6, 1, 2048, 64, 248, 2, 130].$$

Using variable byte encoding,

- What is the largest gap you can encode in 1 byte?
- What is the largest gap you can encode in 2 bytes?
- How many bytes will the above gaps list require under this encoding?

- 64
- 2048
- 11

Exercise 3/5

From the following sequence of γ-encoded gaps, reconstruct first the gaps list and then the original postings list. Recall that the α code encodes a number n with n 1s followed by one 0.

$$1110001110101011111101101111011$$

$[1110001, 11010, 101, 11111011011, 11011] \rightsquigarrow [1001, 110, 11, 111011, 111] \rightsquigarrow [9, 6, 3, 59, 7] \rightsquigarrow [9, 15, 18, 77, 84]$

Exercise 3/6

What does the Zipf's law say?

Answers can vary. For official definition refer to the Manning book.

Exercise 3/7

What does the Heaps' law say?

Answers can vary. For official definition refer to the Manning book.

Exercise 3/8

A collection of documents contains 4 words: *one*, *two*, *three*, *four* of decreasing word frequencies f_1, f_2, f_3 and f_4. The total number of tokens in the collection is 5000. Assume that the Zipf's law holds for this collection perfectly. What are the word frequencies?

We use the Zipf's law in Definition 4. The least frequent term is *four*, then *three*, *two* and the most frequent is *one*. Applying the Zipf's law we get

$$\begin{aligned} cf_1 + cf_2 + cf_3 + cf_4 &= 5000 \\ c \cdot 1^{-1} + c \cdot 2^{-1} + c \cdot 3^{-1} + c \cdot 4^{-1} &= 5000 \\ c + \tfrac{1}{2}c + \tfrac{1}{3}c + \tfrac{1}{4}c &= 5000 \\ 12c + 6c + 4c + 3c &= 60000 \\ 25c &= 60000 \\ c &= 2400 \end{aligned}$$

and, plugging in to the formula $cf_i = ci^{-1}$, we obtain the term frequency values:

$$cf_1 = 2400\tfrac{1}{1} = 2400$$

$$cf_2 = 2400\tfrac{1}{2} = 1200$$

$$cf_3 = 2400\tfrac{1}{3} = 800$$

$$cf_4 = 2400\tfrac{1}{4} = 600$$

Exercise 3/9

How many distinct terms are expected in a document of 1,000,000 tokens? Use the Heaps' law with parameters $k = 44$ and $b = 0.5$

By Definition 5,

$$44 \times 1{,}000{,}000^{0.5} = 44{,}000.$$

Seminar 4

Definition 6 (Term weight)
Weight of a term t in a document d is counted as

$$w_{t,d} = \begin{cases} 1 + \log\left(tf_{t,d}\right) & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $tf_{t,d}$ is the number of terms t in a document d.

Definition 7 (Inverse document frequency)
Inverse document frequency of a term t is defined as

$$idf_t = \log\left(\frac{N}{df_t}\right)$$

where N is the number of all documents and df_t (document frequency) is the number of documents that contain t.

Definition 8 (tf-idf weighting scheme)
In the tf-idf weighting scheme, a term t in a document d has weight

$$\textit{tf-idf}_{t,d} = tf_{t,d} \cdot idf_t$$

Definition 9 (Cosine (Euclidean) normalization)
A vector v is cosine-normalized by

$$v_j = \frac{v_j}{||v||} = \frac{v_j}{\sqrt{\sum_{k=1}^{|v|} {v_k}^2}}$$

where v_j be the number on the j-th position in v.

Exercise 4/1

Consider the frequency table of the words of three documents doc_1, doc_2, doc_3 below. Calculate the *tf-idf* weight of the terms *car, auto, insurance, best* for each document. *idf* values of terms are in the table.

	doc_1	doc_2	doc_3	*idf*
car	27	4	24	1.65
auto	3	33	0	2.08
insurance	0	33	29	1.62
best	14	0	17	1.5

Table 3: Exercise.

After counting *tf-idf* weights by Definition 8 individually for each term we get the following table

	tf-idf		
	doc_1	doc_2	doc_3
car	44.55	6.6	39.6
auto	6.24	68.64	0
insurance	0	53.46	46.98
best	21	0	25.5

Table 4: Solution.

Exercise 4/2

Count document representations as normalized Euclidean weight vectors for each document from the previous exercise. Each vector has four components, one for each term.

Normalized Euclidean weight vectors are counted by Definition 9. Denominators m_{doc_n} for the individual documents are

$$m_{doc_1} = \sqrt{44.55^2 + 6.24^2 + 21^2} = 49.6451$$

$$m_{doc_2} = \sqrt{6.6^2 + 68.64^2 + 53.46^2} = 87.2524$$

$$m_{doc_3} = \sqrt{39.6^2 + 46.98^2 + 25.5^2} = 66.5247$$

and the document representations are

$$d_1 = \left(\frac{44.55}{49.6451}; \frac{6.24}{49.6451}; \frac{0}{49.6451}; \frac{21}{49.6451}\right) = (0.8974; 0.1257; 0; 0.423)$$

$$d_2 = \left(\frac{6.6}{87.2524}; \frac{68.64}{87.2524}; \frac{53.46}{87.2524}; \frac{0}{87.2524}\right) = (0.0756; 0.7876; 0.6127; 0)$$

$$d_3 = \left(\frac{39.6}{66.5247}; \frac{0}{66.5247}; \frac{46.98}{66.5247}; \frac{25.5}{66.5247}\right) = (0.5953; 0; 0.7062; 0.3833)$$

Exercise 4/3

Based on the weights from the last exercise, compute the relevance scores of the three documents for the query *car insurance*. Use each of the two weighting schemes:

a) Term weight is 1 if the query contains the word and 0 otherwise.

b) Euclidean normalized *tf-idf*.

Please note that a document and a representation of this document are different things. Document is always fixed but the representations may vary under different settings and conditions. In this exercise we fix document representations from the last exercises and will count relevance scores for query and documents under two different representations of the query. It might be helpful to view on a query as on another document, as it is a sequence of words.

We count the relevance scores for **a)** as the scalar products of the representation of the query $q = (1, 0, 1, 0)$ with representations of the documents d_n from the last exercise:

$$q \cdot d_1 = 1 \cdot 0.8974 + 0 \cdot 0.1257 + 1 \cdot 0 + 0 \cdot 0.423 = 0.8974$$

$$q \cdot d_2 = 1 \cdot 0.0756 + 0 \cdot 0.7876 + 1 \cdot 0.6127 + 0 \cdot 0 = 0.6883$$

$$q \cdot d_3 = 1 \cdot 0.5953 + 0 \cdot 0 + 1 \cdot 0.7062 + 0 \cdot 0.3833 = 1.3015$$

For **b)** we first need the normalized *tf-idf* vector q, which is obtained by dividing each component of the query by the length of *idf* vector $\sqrt{1.65^2 + 0^2 + 1.62^2 + 0^2} = 2.3123$

	tf	*idf*	*tf-idf*	q
car	1	1.65	1.65	0.7136
auto	0	2.08	0	0
insurance	1	1.62	1.62	0.7006
best	0	1.5	0	0

Table 5: Process of finding the Euclidean normalized *tf-idf*.

Now we multiply q with the document vectors and we obtain the relevance scores:

$$q \cdot d_1 = 0.7136 \cdot 0.8974 + 0 \cdot 0.1257 + 0.7006 \cdot 0 + 0 \cdot 0.423 = 0.6404$$

$$q \cdot d_2 = 0.7136 \cdot 0.0756 + 0 \cdot 0.7876 + 0.7006 \cdot 0.6127 + 0 \cdot 0 = 0.4832$$

$$q \cdot d_3 = 0.7136 \cdot 0.5953 + 0 \cdot 0 + 0.7006 \cdot 0.7062 + 0 \cdot 0.3833 = 0.9196$$

Exercise 4/4

Consider a collection of documents and the terms *dog*, *cat* and *food* that occur in 10^{-3x}, 10^{-2x} and 10^{-x} of the documents, respectively. Now document doc1 contains the words $2y$, y and $3y$ times and doc2 $2z$, $3z$ and z times. Order these two documents based on vector space similarity with the query *dog food*.

Intuitively, doc_1 is more relevant than doc_2 because doc_2 is relatively too much about cats and too little about food, which is a satisfactory answer. But precisely:

	doc_1	doc_2	q
dog	$2y \cdot 3x$	$2z \cdot 3x$	$3x$
cat	$y \cdot 2x$	$3z \cdot 2x$	0
food	$3y \cdot x$	$z \cdot x$	x

Table 6: *tf-idf*.

	doc_1	doc_2	q
dog	$6xy/7xy = {}^{6}/_{7}$	$6xz/8.5xz = {}^{12}/_{17}$	$3x/3.2x = {}^{15}/_{16}$
cat	$2xy/7xy = {}^{2}/_{7}$	$6xz/8.5xz = {}^{12}/_{17}$	0
food	$3xy/7xy = {}^{3}/_{7}$	$xz/8.5xz = {}^{1}/_{17}$	$x/3.2x = {}^{5}/_{16}$

Table 7: Representations.

	$q \cdot doc_1$	$q \cdot doc_2$
dog	$^6/_7 \cdot {}^{15}/_{16} \sim 0.8$	$^{12}/_{17} \cdot {}^{15}/_{16} \sim 0.66$
cat	0	0
food	$^3/_7 \cdot {}^5/_{16} \sim 0.13$	$^1/_{17} \cdot {}^5/_{16} \sim 0.02$

Table 8: Relevance.

Here $0.8 + 0.13 > 0.66 + 0.02$ and therefore doc_1 is more relevant than doc_2.

Exercise 4/5

Calculate the vector-space similarity between the query *digital cameras* and a document containing *digital cameras and video cameras* by filling in the blank columns in the table below. Assume $N = 10000000$, logarithmic term weighting (columns w) for both query and documents, *idf* weighting only for the query and cosine normalization only for the document. *and* is a STOP word.

		Query				Document			relevance
	df	*tf*	*w*	*idf*	*q*	*tf*	*w*	*d*	$q \cdot d$
digital	10 000								
video	100 000								
cameras	50 000								

Table 9: Exercise.

The *tf* value is filled according to the occurrences of the terms in both query and document.

$$\begin{aligned} \mathrm{tf}_q &= \textit{digital cameras} &= (1,0,1) \\ \mathrm{tf}_d &= \textit{digital cameras and video cameras} &= (1,1,2) \end{aligned}$$

Logarithmic weighting uses the Definition 6. For the query the values are

$$\begin{aligned} w_{digital} &= 1 + \log(1) &= 1 + 0 &= 1 \\ w_{video} &= 0 \\ w_{cameras} &= 1 + \log(1) &= 1 + 0 &= 1 \end{aligned}$$

and for the document

$$\begin{aligned} w_{digital} &= 1 + \log(1) &= 1 + 0 &= 1 \\ w_{video} &= 1 + \log(1) &= 1 + 0 &= 1 \\ w_{cameras} &= 1 + \log(2) &= 1 + 0.301 &= 1.301 \end{aligned}$$

Now we need to count the *idf* weights for the query. These are counted by Definition 7.

$$\begin{aligned} idf_{digital} &= \log\left(\tfrac{10^7}{10^4}\right) &= \log\left(10^3\right) &= 3 \\ idf_{video} &= \log\left(\tfrac{10^7}{10^5}\right) &= \log\left(10^2\right) &= 2 \\ idf_{cameras} &= \log\left(\tfrac{10^7}{5 \times 10^4}\right) &= \log(200) &= 2.301 \end{aligned}$$

and $q = w \cdot idf$. Cosine normalization for the document is counted similarly as in the last exercises by Definition 9 using w.

$$\begin{aligned} d_{digital} &= \frac{1}{\sqrt{1^2+1^2+1.301^2}} &= 0.5204 \\ d_{video} &= \frac{1}{\sqrt{1^2+1^2+1.301^2}} &= 0.5204 \\ d_{cameras} &= \frac{1.301}{\sqrt{1^2+1^2+1.301^2}} &= 0.677 \end{aligned}$$

The score is the scalar multiple of q and d. The final table is

		Query				Document			relevance
	df	*tf*	*w*	*idf*	*q*	*tf*	*w*	*d*	$q \cdot d$
digital	10 000	1	1	3	3	1	1	0.5204	1.5612
video	100 000	0	0	2	0	1	1	0.5204	0
cameras	50 000	1	1	2.301	2.301	2	1.301	0.677	1.5578

Table 10: Solution.

and the similarity score is

$$score(d, q) = \sum_{i=1}^{3} (d_i \cdot q_i) = 3.119.$$

Exercise 4/6

Show that for the query q_1 = *affection* the documents in the table below are sorted by relevance in the opposite order as for the query q_2 = *jealous gossip*. Query is *tf* weight normalized.

	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

Table 11: Exercise.

We add queries to the original table:

	SaS	PaP	WH	q_1	q_2
affection	0.996	0.993	0.847	1	0
jealous	0.087	0.120	0.466	0	1
gossip	0.017	0	0.254	0	1

Table 12: Exercise with queries.

Now we normalize the vectors q_i by Definition 9 and get

	SaS	PaP	WH	q_1	q_2	q_{1n}	q_{2n}
affection	0.996	0.993	0.847	1	0	1	0
jealous	0.087	0.120	0.466	0	1	0	0.7071
gossip	0.017	0	0.254	0	1	0	0.7071

Table 13: Exercise with queries after normalization.

In the last step we count the similarity score between the queries and documents by $score(d, q) = \sum_{i=1}^{|d|}(d_i \cdot q_i)$

$$\begin{aligned}
score(SaS, q_1) &= 0.9961 \cdot 1 + 0.087 \cdot 0 + 0.017 \cdot 0 &= 0.9961 \\
score(PaP, q_1) &= 0.993 \cdot 1 + 0.120 \cdot 0 + 0 \cdot 0 &= 0.993 \\
score(WH, q_1) &= 0.847 \cdot 1 + 0.466 \cdot 0 + 0.254 \cdot 0 &= 0.847 \\
\\
score(SaS, q_2) &= 0.9961 \cdot 0 + 0.087 \cdot 0.7071 + 0.017 \cdot 0.7071 &= 0.0735 \\
score(PaP, q_2) &= 0.993 \cdot 0 + 0.120 \cdot 0.7071 + 0 \cdot 0.7071 &= 0.0849 \\
score(WH, q_2) &= 0.847 \cdot 0 + 0.466 \cdot 0.7071 + 0.254 \cdot 0.7071 &= 0.5091
\end{aligned}$$

The ordering for q_1 is SaS > PaP > WH and for q_2 is WH > PaP > SaS, and we see that they are opposite.

Seminar 6

Definition 10 (Recall)
Recall describes how many of the relevant documents are retrieved.

$$recall = R = \frac{\#relevant\ retrieved}{\#relevant}$$

Definition 11 (Precision)
Precision describes how many of the retrieved documents are relevant.

$$precision = P = \frac{\#relevant\ retrieved}{\#retrieved}$$

Definition 12 (*F*-measure)
A balanced F-measure (F_1-measure) defines a recall-precision relationship represented by their weighted harmonic mean:

$$F = \frac{2 \cdot R \cdot P}{R + P}$$

Definition 13 (Mean Average Precision)
MAP expresses the precision in each point a new relevant document is included in the result. Is counted as

$$MAP(Q) = \frac{1}{|Q|} \cdot \left(\sum_{q \in Q} \frac{1}{rel_q} \cdot \left(\sum_{i=1}^{rel_q} prec_i \right) \right)$$

where rel_q is the number of relevant documents for query q and $prec_i$ is the precision at the i-th document.

Definition 14 (κ statistic)
Let N be the total number of documents, J is a set of judges and $P(A) = \frac{\#agree}{N}$ the number of documents on which the judges agree. Let also define R_j and NR_j be the number of relevant and non-relevant documents, respectively, according to the judge $j \in J$ and

$$P(R) = \frac{\sum_{j \in J} R_j}{|J| \cdot N} \quad \text{and} \quad P(NR) = \frac{\sum_{j \in J} NR_j}{|J| \cdot N}$$

as the number of relevant and non-relevant documents, respectively. Let finally define

$$P(E) = P(R)^2 + P(NR)^2$$

as the approximate number of disagreements between the judges. Then the κ statistic is defined as the measure of agreement between the judges

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}.$$

Definition 15 (Rocchio relevance feedback)
Rocchio relevance feedback has the form

$$q_m = \alpha q_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d_r} \in D_r} \vec{d_r} - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d_{nr}} \in D_{nr}} \vec{d_{nr}}$$

where q_0 is the original query vector, D_r is the set of relevant documents, D_{nr} is the set of non-relevant documents and the values α, β, γ depend on the system setting.

Exercise 6/1

The following ordered list of 20 letters R and N represents relevant (R) and non-relevant (N) retrieved documents as an answer for a query on a collection of 10 000 documents. The leftmost document is expected to be the most relevant. The list contains 6 relevant documents. Assume that the collection contains 8 documents relevant to the query.

$$RRNNNNNNRNRNNNRNNNNR$$

a) What is the precision on the first 20 results?

b) What is the F-measure on the first 20 results?

c) What is the non-interpolated precision of the system at 25% recall? (R=25%)

d) What is the interpolated precision of the system at 33% recall? (R>33%)

e) Assume that these 20 documents is the complete list of retrieved documents. What is the MAP of the system?

Now assume that the system returned all 10,000 documents in an ordered list and above is the top 20.

f) What is the highest possible MAP the system can achieve?

g) What is the lowest possible MAP the system can achieve?

Applying the Definition 11 do calculate (a) we get $P = \frac{6}{20} = \frac{3}{10}$. For (b) it is necessary to count the recall with Definition 10 as $R = \frac{6}{8} = \frac{3}{4}$. Using Definition 12 with $\alpha = 0.5$ we count

$$\frac{(\beta^2+1)PR}{\beta^2 P + R} = \frac{(1^2+1)\cdot\frac{3}{10}\cdot\frac{3}{4}}{\frac{3}{10}+\frac{3}{4}} = \frac{\frac{9}{20}}{\frac{21}{20}} = \frac{3}{7}.$$

To count the non-interpolated precision of the system at 25% recall for (c) we need the precisions for the documents of recall equal to 25%:

$$\begin{aligned}
&1. \quad P = \tfrac{1}{1} \qquad R = \tfrac{1}{8} \quad = 12.5\% \\
&2. \quad P = \tfrac{2}{2} \qquad R = \tfrac{2}{8} \quad = 25\% \\
&3. \quad P = \tfrac{2}{3} \qquad R = \tfrac{2}{8} \quad = 25\% \\
&\qquad\qquad \dots \\
&8. \quad P = \tfrac{2}{8} \qquad R = \tfrac{2}{8} \quad = 25\% \\
&9. \quad P = \tfrac{3}{9} \qquad R = \tfrac{3}{8} \quad = 37.5\%
\end{aligned}$$

We see that the first document has $R = 12.5\%$ but this value is less than 25%. Documents 2 to 8 have the desired recall of 25%. Document 9 has already a higher value so we do not include it in the result. Non-interpolated precision is then the set of precisions of these 7 documents

$$P = \left\{ \frac{2}{2}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}, \frac{2}{6}, \frac{2}{7}, \frac{2}{8} \right\}.$$

For the interpolated precision (d) we are looking for the highest precision for the relevant documents of recall higher than 33%. Recall changes if and only if the result contains a relevant document. Therefore, the values are calculated for the documents 11, 15 and 20.

$$\begin{aligned}
&11. \quad P = \tfrac{4}{11} \qquad R = \tfrac{4}{8} \quad = 50\% \\
&15. \quad P = \tfrac{5}{15} \qquad R = \tfrac{5}{8} \quad = 62.5\% \\
&20. \quad P = \tfrac{6}{20} \qquad R = \tfrac{6}{8} \quad = 75\%
\end{aligned}$$

The requested recall value is exceeded by retrieving the document 9 (37.5 %). Now we only have to find $max\{P_9, P_{11}, P_{15}, P_{20}\} = max\{\frac{3}{9}, \frac{4}{11}, \frac{5}{15}, \frac{6}{20}\} = \frac{4}{11} = 0.\overline{36}$.

To estimate the MAP of the system (e) we use the Definition 13. Since we only have one query $N = |Q| = 1$ and the first 20 documents contain $rel_q = 6$ relevant documents

$$MAP(Q) = \frac{1}{1}\left(\sum_{j=1}^{1}\frac{1}{8}\left(\sum_{i=1}^{6} P(doc_i)\right)\right) = \frac{1}{1}\cdot\frac{1}{8}\left(\underbrace{\frac{1}{1}}_{P(1.)} + \underbrace{\frac{2}{2}}_{P(2.)} + \underbrace{\frac{3}{9}}_{P(9.)} + \underbrace{\frac{4}{11}}_{P(11.)} + \underbrace{\frac{5}{15}}_{P(15.)} + \underbrace{\frac{6}{20}}_{P(20.)}\right)$$

$$= \frac{1}{1}\cdot\frac{1}{8}\cdot\frac{1099}{330} = \frac{1099}{2640} = 0.416\overline{287}$$

From Definition 11 and Definition13 follows that if the two remaining relevant documents were on positions 21 and 22 then MAP with $rel_q = 8$ relevant documents is the highest possible (f)

$$MAP(Q) = \frac{1}{8}\left(\frac{1}{1} + \frac{2}{2} + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20} + \frac{7}{21} + \frac{8}{22}\right) = 0.503\overline{409}$$

and, on the other hand, if they were on the last places the MAP would be minimal (g)

$$MAP(Q) = \frac{1}{8}\left(\frac{1}{1} + \frac{2}{2} + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20} + \frac{7}{9999} + \frac{8}{10000}\right) = 0.4164\overline{7538}.$$

Exercise 6/2

The following ordered list of 5 letters R and N represent relevant (R) and non-relevant (N) retrieved documents as an answer for a query on a collection of 100 documents. The leftmost document is expected to be the most relevant. The list contains 3 relevant documents. Assume that the collection contains 5 documents relevant to the query.

$$RNNRR$$

- What is the F-measure on the first 5 results?

Assume that these 5 documents is the complete list of retrieved documents.

- What is the MAP of the system?

Now assume that the system returned all 100 documents in an ordered list and above is the top 5.

- What are the highest and lowest possible MAPs the system can achieve?

- $R = \frac{3}{5}$, $P = \frac{3}{5}$, then $F = \frac{3}{5}$.
- $\frac{1}{5}\left(\frac{1}{1} + \frac{2}{4} + \frac{3}{5}\right)$.
- highest $\frac{1}{5}\left(\frac{1}{1} + \frac{2}{4} + \frac{3}{5} + \frac{4}{6} + \frac{5}{7}\right)$ and lowest $\frac{1}{5}\left(\frac{1}{1} + \frac{2}{4} + \frac{3}{5} + \frac{4}{99} + \frac{5}{100}\right)$.

Exercise 6/3

The following two sequences of letters R and N represent the complete lists of relevant (R) and non-relevant (N) retrieved documents as answers for two queries on a collection of 100 documents. The leftmost document is expected to be the most relevant. Assume that the collection contains 10 documents relevant to the first query and 20 documents relevant to the second query. Find the F-measure and the MAP of this system.

$$NRNNNRN \quad \text{and} \quad NNRRR$$

- $R_1 = \frac{2}{10}$, $P_1 = \frac{2}{7}$, then $F_1 = \frac{\frac{4}{35}}{\frac{17}{35}} = \frac{4}{17}$.
- $R_2 = \frac{3}{20}$, $P_2 = \frac{3}{5}$, then $F_2 = \frac{\frac{9}{50}}{\frac{3}{4}} = \frac{6}{25}$.
- $F = \frac{F_1+F_2}{2} = \frac{101}{425}$.
- $\text{MAP}_1 = \frac{1}{10}\left(\frac{1}{2} + \frac{2}{6}\right) = \frac{5}{2*10*6} = \frac{1}{12}$.
- $\text{MAP}_2 = \frac{1}{20}\left(\frac{1}{3} + \frac{2}{4} + \frac{3}{5}\right) = \frac{20+30+12}{2*20*60} = \frac{31}{600}$.
- $\text{MAP} = \frac{\text{MAP}_1+\text{MAP}_2}{2} = \frac{81}{1200}$.

Exercise 6/4

Below is a table showing how two judges judged the relevance (0 = non-relevant, 1 = relevant) of the set of 12 documents with respect to a query. Assume that you developed an IR system, that for this query returns the documents $\{4, 5, 6, 7, 8\}$.

Doc ID	Judge 1	Judge 2
1	0	0
2	0	0
3	1	1
4	1	1
5	1	0
6	1	0
7	1	0
8	1	0
9	0	1
10	0	1
11	0	1
12	0	1

Table 14: Judges judging the relevance of documents.

a) Calculate the κ statistic.

b) Calculate the recall, precision and F-measure of your system in which a document is considered relevant if the judges agree.

c) Calculate the recall, precision and F-measure of your system in which a document is considered relevant if at least one of the judges thinks so.

For (a) it is necessary to use the Definition 14. First we count $P(A)$ as the number of documents on which the judges agree. Since these are the documents $\{1, 2, 3, 4\}$ and the total number of them is $N = 12$, then $P(A) = \frac{|\{1,2,3,4\}|}{12} = \frac{4}{12} = \frac{1}{3}$. Now we need the counts of disagreements between the judges. Judge 1 considers the documents $NR_1 =$

$\{1, 2, 9, 10, 11, 12\}$ and judge 2 $NR_2 = \{1, 2, 5, 6, 7, 8\}$ as non-relevant. Plugging in to the formula for $P(NR)$ we get $P(NR) = \frac{|NR_1|+|NR_2|}{2 \cdot 12} = \frac{2 \cdot 6}{24} = \frac{1}{2}$. We repeat this for $P(R)$. Since the number of relevant is equal to non-relevant, then $P(R) = P(NR) = \frac{1}{2}$. We finally can count $P(E)$ as

$$P(E) = P(NR)^2 + P(R)^2 = \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

and

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} = \frac{\frac{1}{3} - \frac{1}{2}}{1 - \frac{1}{2}} = -\frac{1}{3}.$$

If $\kappa < 0$ then the agreement between the judges is more than random.

For (b) it is necessary to calculate recall and precision. Our system retrieves the documents $\{4, 5, 6, 7, 8\}$ as relevant whereas the judges only agree on $\{3, 4\}$. The intersection is $\{4\}$. As to the Definition 11 we obtain

$$P = \frac{|\{4\}|}{|\{4, 5, 6, 7, 8\}|} = \frac{1}{5}$$

and, as the number of relevant documents is $|\{3, 4\}| = 2$, the recall is $R = \frac{1}{2}$ by Definition 10. Then, by Definition 12 the F-measure is equal to

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} = \frac{(1 + 1)\frac{1}{5}\frac{1}{2}}{\frac{1}{5} + \frac{1}{2}} = \frac{2}{7}.$$

We similarly count these for (c) which says that a document is relevant if at least one judge considers it relevant. This makes the documents $\{3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ relevant. Their intersection with our result $\{4, 5, 6, 7, 8\}$ is the set $\{4, 5, 6, 7, 8\}$ of size 5. Recall is $R = \frac{|\{4,5,6,7,8\}|}{|\{3,4,5,6,7,8,9,10,11,12\}|} = \frac{5}{10} = \frac{1}{2}$, precision is $P = \frac{5}{5} = 1$ and F-measure is

$$F = \frac{(1 + 1)\frac{1}{2}}{1 + \frac{1}{2}} = \frac{2}{3}.$$

Exercise 6/5

What is the main purpose of Rocchio relevance feedback?

Answers can vary. For official definition refer to the Manning book.

Exercise 6/6

A user's primary query is *cheap CDs cheap DVDs extremely cheap CDs*. The user has a look on two documents: doc1 a doc2, marking doc1 *CDs cheap software cheap CDs* as relevant and doc2 *cheap thrills DVDs* as non-relevant. Assume that we use a simple *tf* scheme without vector length normalization. What would be the restructured query vector after considering the Rocchio relevance feedback with values $\alpha = 1$, $\beta = 0.75$ and $\gamma = 0.25$?

We rewrite the exercise to the table for an easier processing.

	relevant	non-relevant	
terms	doc1	doc2	query
Cds	2	0	2
cheap	2	1	3
software	1	0	0
thrills	0	1	0
DVDs	0	1	1
extremely	0	0	1

Table 15:

Now we mark the input if the algorithm by Definition 15.

$$d_r \in D_r = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \ , \ d_{nr} \in D_{nr} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \ , \ q = \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

By filling the values to the formula for q_m we get

$$\begin{aligned} q_m &= 1 \cdot \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} + 0.75 \cdot \tfrac{1}{1} \begin{pmatrix} 2 \\ 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} - 0.25 \cdot \tfrac{1}{1} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 2 \\ 3 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1.5 \\ 1.5 \\ 0.75 \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 0.25 \\ 0 \\ 0.25 \\ 0.25 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 3.5 \\ 4.25 \\ 0.75 \\ -0.25 \\ 0.75 \\ 1 \end{pmatrix} \end{aligned}$$

Seminar 7

Definition 16 (Naive Bayes Classifier)
Naive Bayes (NB) Classifier assumes that the effect of the value of a predictor x on a given class c is class conditional independent. Bayes theorem provides a way of calculating the

posterior probability $P(c|x)$ from class prior probability $P(c)$, predictor prior probability $P(x)$ and probability of the predictor given the class $P(x|c)$

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

and for a vector of predictors $X = (x_1, \ldots, x_n)$

$$P(c|X) = P(x_1|c) \ldots P(x_n|c)P(c).$$

The class with the highest posterior probability is the outcome of prediction.

Exercise 7/1

What is naive about Naive Bayes classifier? Briefly outline its major idea.

Answers can vary. For official definition refer to the Manning book.

Exercise 7/2

Considering the table of observations, use the Naive Bayes classifier to recommend whether to *Play Golf* given a day with *Outlook = Rainy*, *Temperature = Mild*, *Humidity = Normal* and *Windy = True*. Do not deal with the zero-frequency problem.

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Table 16: Exercise.

First build the likelihood tables for each predictor

		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

		Play Golf		
		Yes	No	
Temperature	Hot	2/9	2/5	4/14
	Mild	4/9	2/5	6/14
	Cool	3/9	1/5	4/14
		9/14	5/14	

		Play Golf		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

		Play Golf		
		Yes	No	
Windy	True	3/9	2/5	5/14
	False	6/9	3/5	9/14
		9/14	5/14	

We see that probability of *Sunny* given *Yes* is $3/9 = 0.33$, probability of *Sunny* is $5/14 = 0.36$ and probability of *Yes* is $9/14 = 0.64$. Then we count the likelihoods of *Yes* and *No*

$$\begin{aligned}
&P(Yes|Rainy, Mild, Normal, True) = \\
&\quad = P(Rainy|Yes) \cdot P(Mild|Yes) \cdot P(Normal|Yes) \cdot P(True|Yes) \cdot P(Yes) \\
&\quad = \frac{2}{9} \cdot \frac{4}{9} \cdot \frac{6}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} = 0.014109347 \\
&P(No|Rainy, Mild, Normal, True) = \\
&\quad = P(Rainy|No) \cdot P(Mild|No) \cdot P(Normal|No) \cdot (True|No) \cdot P(No) \\
&\quad = \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} = 0.010285714
\end{aligned} \tag{1}$$

and suggest *Yes*. We can normalize the likelihoods to obtain the % confidence:

$$P(Yes|Rainy, Mild, Normal, True) = \frac{0.014109347}{0.014109347 + 0.010285714} = 57.84\%$$

$$P(No|Rainy, Mild, Normal, True) = \frac{0.010285714}{0.014109347 + 0.010285714} = 42.16\%$$

Definition 17 (Support Vector Machines Classifier (two-class, linearly separable)) *Support Vector Machines (SVM) finds the hyperplane that bisects and is perpendicular to the connecting line of the closest points from the two classes. The separating (decision) hyperplane is defined in terms of a normal (weight) vector* $\mathbf{w}$ *and a scalar intercept term* b *as*

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b$$

where $\cdot$ *is the dot product of vectors. Finally, the SVM classifier becomes*

$$class(x) = sgn(f(x)).$$

Exercise 7/3

Draw a sketch explaining the concept of SVM classifier. Include the equation of the separation hyperplane. What are limitations of SVM?

Answers can vary. For official definition refer to the Manning book.

Exercise 7/4

Build the SVM classifier for the training set $\{([1,1],-1),([2,0],-1),([2,3],+1)\}$.

We first take the closest two points from the respective classes: $[1,1]$ and $[2,3]$. We have $\mathbf{w} = a \cdot ([1,1] - [2,3]) = [a, 2a]$. Now we calculate a and b

$$a + 2a + b = -1$$

$$2a + 6a + b = 1$$

for the points $[1,1]$ and $[2,3]$, respectively. The solution is

$$a = \frac{2}{5} \qquad b = \frac{-11}{5}$$

building the weight vector

$$\mathbf{w} = \left[\frac{2}{5}, \frac{4}{5}\right]$$

and the final classifier becomes

$$class(x) = sgn\left(\frac{2}{5}x_1 + \frac{4}{5}x_2 - \frac{11}{5}\right).$$

Exercise 7/5

Explain the concept of classification based on neural networks. Draw a sketch and comment on all components.

Answers can vary. For official definition refer to the Manning book.

Exercise 7/6

What is the difference between supervised and unsupervised learning? Give examples.

Answers can vary. For official definition refer to the Manning book.

Seminar 9

Algorithm 1 K-means($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$, stopping criterion)

```
(s_1, ..., s_K) ← SelectRandomSeeds({x_1, ..., x_N}, K)
for k ← 1 to K do
    μ_k ← s_k
end for
while stopping criterion has not been met do
    for k ← 1 to K do
        ω_k ← {}
    end for
    for n ← 1 to N do
        j ← argmin_{j'} |μ_{j'} − x_n|
        ω_j ← ω_j ∪ {x_n}                          ▷ reassigning vectors
    end for
    for k ← 1 to K do
        μ_k ← (1/|ω_k|) Σ_{x∈ω_k} x                ▷ recomputing centroids
    end for
end while
return {μ_1, ..., μ_K}
```

Exercise 9/1

Use the K-means algorithm with Euclidean distance to cluster the following $N = 8$ examples into $K = 3$ clusters: $A_1 = (2, 10)$, $A_2 = (2, 5)$, $A_3 = (8, 4)$, $A_4 = (5, 8)$, $A_5 = (7, 5)$, $A_6 = (6, 4)$, $A_7 = (1, 2)$, $A_8 = (4, 9)$. Suppose that the initial seeds (centers of each cluster) are A_1, A_4 and A_7. Run the K-means algorithm for 3 epochs. After each epoch, draw a 10×10 space with all the 8 points and show the clusters with the new centroids.

$d(A, B)$ denotes the Euclidean distance between $A = (a_1, a_2)$ and $B = (b_1, b_2)$. It is calculated as $d(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$.

Take seeds $\vec{s}_1 = A_1 = (2, 10)$, $\vec{s}_2 = A_4 = (5, 8)$, $\vec{s}_3 = A_7 = (1, 2)$.

By 1 we count the alignment for epoch 1: $A_1 \in \omega_1$, $A_2 \in \omega_3$, $A_3 \in \omega_2$, $A_4 \in \omega_2$, $A_5 \in \omega_2$, $A_6 \in \omega_2$, $A_7 \in \omega_3$, $A_8 \in \omega_2$; and we get the clusters: $\omega_1 = \{A_1\}$, $\omega_2 = \{A_3, A_4, A_5, A_6, A_8\}$, $\omega_3 = \{A_2, A_7\}$.

Centroids of the clusters: $\vec{\mu}_1 = (2, 10)$, $\vec{\mu}_2 = ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = (6, 6)$, $\vec{\mu}_3 = ((2 + 1)/2, (5 + 2)/2) = (1.5, 3.5)$.

After epoch 2 the clusters are $\omega_1 = \{A_1, A_8\}$, $\omega_2 = \{A_3, A_4, A_5, A_6\}$, $\omega_3 = \{A_2, A_7\}$ with centroids $\vec{\mu}_1 = (3, 9.5)$, $\vec{\mu}_2 = (6.5, 5.25)$ and $\vec{\mu}_3 = (1.5, 3.5)$. And finally after epoch 3, the clusters are $\omega_1 = \{A_1, A_4, A_8\}$, $\omega_2 = \{A_3, A_5, A_6\}$, $\omega_3 = \{A_2, A_7\}$ with centroids $\vec{\mu}_1 = (3.66, 9)$, $\vec{\mu}_2 = (7, 4.33)$ and $\vec{\mu}_3 = (1.5, 3.5)$.

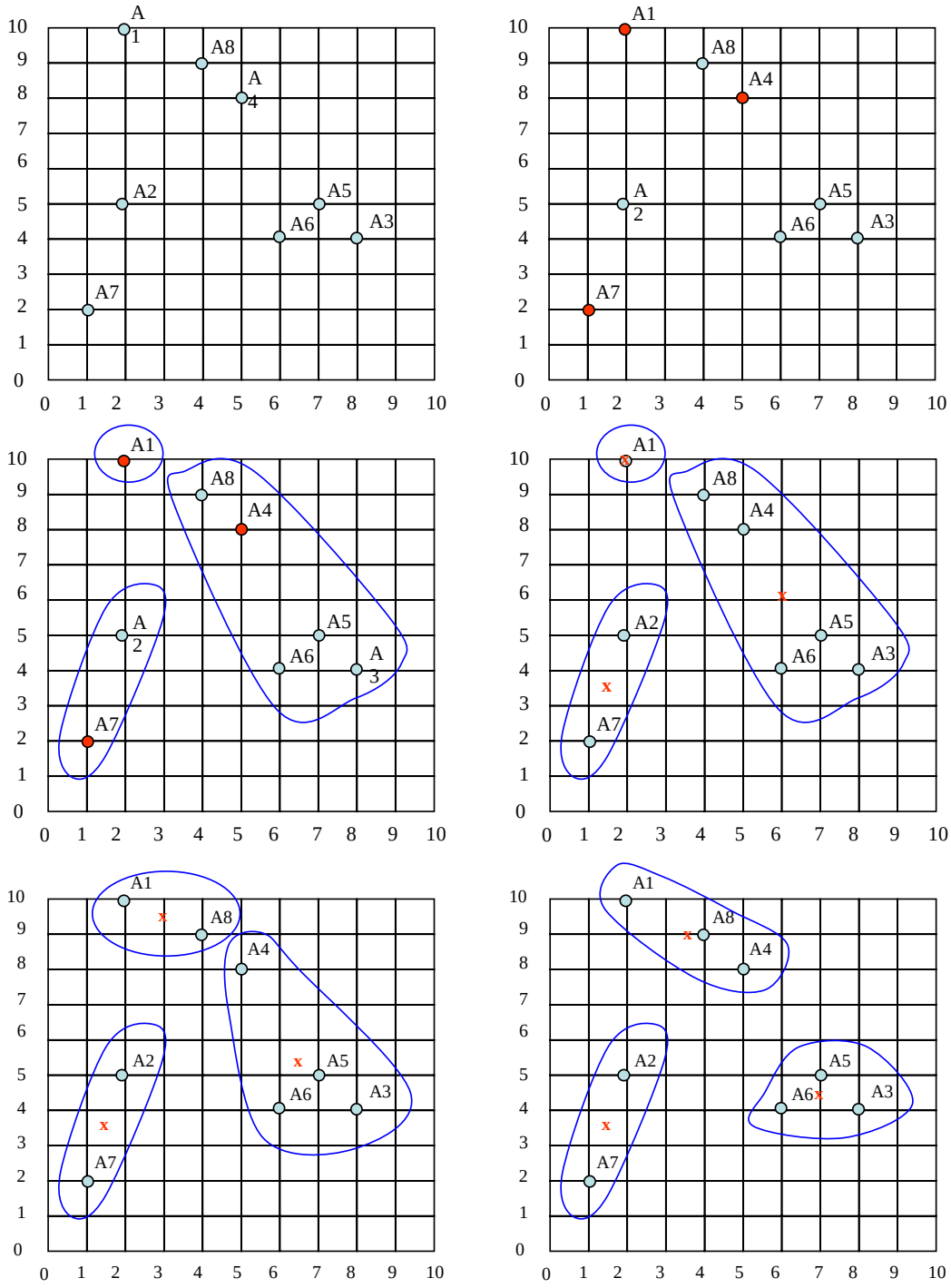

Figure 1: Visualization of K-means clustering algorithm.

Exercise 9/2

Consider three points: $A_1 = [1, 1]$, $A_2 = [3, 1]$, $A_3 = [6, 1]$. Give an example of a point A_4 such that the K-means clustering algorithm with seeds $\{A_2, A_4\}$ and the agglomerative hierarchical clustering algorithm result in different clusterings of $\{A_1, A_2, A_3, A_4\}$ into 2 classes.

For example, if $A_4 = [2, 1]$, then K-means results in $\{\{A_1, A_4\}, \{A_2, A_3\}\}$ and agglomerative in $\{\{A_1, A_2, A_4\}, \{A_3\}\}$.

Exercise 9/3

What makes a good clustering? Give some clustering evaluation metrics.

Answers can vary. For official definition refer to the Manning book.

Seminar 11

Definition 18 (Markov Transition Matrix)
Given the graph $G = (V, E)$ and teleport probability α, let $N = |V|$ and A be the $N \times N$ link matrix with elements

$$\forall u, v \in V \,:\, A_{uv} = \begin{cases} 1 & (u, v) \in E \\ 0 & \textit{otherwise} \end{cases}$$

The transition probability matrix P is then calculated in the following way:

1. *If a row of A has all 0s, then substitute all of them with 1s.*
2. *Divide each 1 by the number of 1s in that row.*
3. *Multiply each entry by $1 - \alpha$.*
4. *Add $\frac{\alpha}{N}$ to each entry.*

Algorithm 4 (PageRank)

```
function PageRank(P)
    i ← 0
    x⃗_i = (1, 0, ..., 0)
    x⃗_{i+1} = (0, 0, ..., 0)
    repeat
        x⃗_{i+1} = x⃗_i · P
        i = i + 1
    until x_i = x_{i−1}
end function
```

Definition 19 (Hubs and authorities)
Given the link matrix A, let $h(v)$ denote the hub score and $a(v)$ the authority score. First, set the $h(v)$ a $a(v)$ vectors to 1^N for all vertices $v \in V$. The scores are calculated as

$$\begin{aligned} h(v) &= A \cdot a(v) \\ a(v) &= A^T \cdot h(v) \end{aligned}$$

which is equivalent to

$$\begin{aligned} h(v) &= A \cdot A^T \cdot h(v) \\ a(v) &= A^T \cdot A \cdot a(v) \end{aligned}$$

Exercise 11/1

Assume the web graph $G = (V = \{a, b, c\},\ E = \{(a, b), (a, c), (b, c), (c, b)\})$. Count PageRank, hub and authority scores for each of the web pages. Rank the pages by the individual scores and observe the connections. You can assume that in each step of the random walk we teleport to a random page with probability 0.1 and uniform distribution. Normalize the hub and authority scores so that the maximum is 1.

By Definition 18, rewrite the graph as a link matrix

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Trying to apply the step 1 the algorithm, does not contain a row with all 0s, so we proceed to step 2. First row contains two 1s so we divide both of them by 2. Second and third lines only contain one 1, so dividing them by 1 makes no change

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{matrix} :2 \\ :1 \\ :1 \end{matrix} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Now apply step 3. Since $\alpha = 0.1$, multiply all entries by 0.9

$$\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot\ 0.9 = \begin{pmatrix} 0 & \frac{9}{20} & \frac{9}{20} \\ 0 & 0 & \frac{9}{10} \\ 0 & \frac{9}{10} & 0 \end{pmatrix}$$

and by step 4 add $\frac{\alpha}{N} = \frac{0.1}{3} = \frac{1}{30}$

$$\begin{pmatrix} 0 & \frac{9}{20} & \frac{9}{20} \\ 0 & 0 & \frac{9}{10} \\ 0 & \frac{9}{10} & 0 \end{pmatrix} + \frac{1}{30} = \begin{pmatrix} \frac{1}{30} & \frac{29}{60} & \frac{29}{60} \\ \frac{1}{30} & \frac{1}{30} & \frac{14}{15} \\ \frac{1}{30} & \frac{14}{15} & \frac{1}{30} \end{pmatrix} = P.$$

Using the Algorithm 4 for PageRank, we select $\overrightarrow{x}_0 = (1, 0, 0)$ and the transition probability matrix P from the previous calculation.

$$\overrightarrow{x}_1 = \overrightarrow{x}_0 \cdot P \tag{2}$$

$$\overrightarrow{x}_1 = (1,0,0) \cdot \begin{pmatrix} \frac{1}{30} & \frac{29}{60} & \frac{29}{60} \\ \frac{1}{30} & \frac{1}{30} & \frac{14}{15} \\ \frac{1}{30} & \frac{14}{15} & \frac{1}{30} \end{pmatrix} \tag{3}$$

$$\overrightarrow{x}_1 = \left(\frac{1}{30}, \frac{29}{60}, \frac{29}{60}\right) \tag{4}$$

$$\overrightarrow{x}_2 = \overrightarrow{x}_1 \cdot P \tag{5}$$

$$\overrightarrow{x}_2 = \left(\frac{1}{30}, \frac{29}{60}, \frac{29}{60}\right) \cdot \begin{pmatrix} \frac{1}{30} & \frac{29}{60} & \frac{29}{60} \\ \frac{1}{30} & \frac{1}{30} & \frac{14}{15} \\ \frac{1}{30} & \frac{14}{15} & \frac{1}{30} \end{pmatrix} \tag{6}$$

$$\overrightarrow{x}_2 = \left(\frac{1}{30}, \frac{29}{60}, \frac{29}{60}\right) \tag{7}$$

Since $x_i = x_{i-1}$, we claim the entries of x_2 as PageRanks of the individual web pages. By Definition 19, we set the hub score $h(v)$ and the authority score $a(v)$ to 1s

$$a(v) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad h(v) = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Substituting into the second equation in the definition, we get the hub score

$$\begin{aligned} h(v) &= A \cdot A^T \cdot h(v) \\ h(v) &= \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\ h(v) &= \begin{pmatrix} 2 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\ h(v) &= \begin{pmatrix} 4 \\ 2 \\ 2 \end{pmatrix} \end{aligned}$$

and the authority score

$$\begin{aligned} a(v) &= A^T \cdot A \cdot a(v) \\ a(v) &= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\ a(v) &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\ a(v) &= \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix}. \end{aligned}$$

Finally, we normalize them to obtain

$$h(v) = \begin{pmatrix} 4 \\ 2 \\ 2 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 \\ 0.5 \\ 0.5 \end{pmatrix} \qquad \text{and} \qquad a(v) = \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

Seminar 12

Exercise 12/1

Pick a topic of your interest and describe it by 5-10 words.
Open Sketch Engine `https://ske.fi.muni.cz/`. Go to WebBootCaT. Create a corpus using the description words as seed. Wait until data are downloaded. Search the word corpus for collocations.

Solutions can vary.

Definition 20 (Index Relations)
Suppose that we could pick a random page from the index of E_1 and test whether it is in E_2's index and symmetrically, test whether a random page from E_2 is in E_1. These experiments give us fractions x and y such that our estimate is that a fraction x of the pages in E_1 are in E_2, while a fraction y of the pages in E_2 are in E_1. Then, letting $|E_i|$ denote the size of the index of search engine E_i, we have

$$x|E_1| \approx y|E_2|,$$

from which we have the form we will use

$$\frac{|E_1|}{|E_2|} \approx \frac{y}{x}.$$

Exercise 12/2

Two web search engines A and B each generate a large number of pages uniformly at random from their indexes. 30% of A's pages are present in B's index, while 50% of B's pages are present in A's index. What is the number of pages in A's index relative to B's?

Substituting to the Definition 20 we get the fractions

- $\frac{3}{10}$ of A is in B
- $\frac{5}{10}$ of B is in A

and we get the equation

$$\begin{aligned} 0.3|A| &\approx 0.5|B| \\ \frac{|A|}{|B|} &\approx \frac{0.5}{0.3} \\ \frac{|A|}{|B|} &\approx \frac{5}{3} \end{aligned}$$

Definition 21 (Path Similarity)
Similarity between a query XPath c_q and a document path c_d is calculated as

$$CR(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ can be expanded to } c_d \text{ by adding nodes to the path} \\ 0 & \text{otherwise} \end{cases}$$

Definition 22 (Structural Term)
Structural term is defined as an XML-context/term pair denoted by `<c,t>` *of existing path to a value and the value itself, where the value itself is also a node in the XML document. For example, an XML document containing only a* `root` *element with* `test`*.*

```
<root>
   test
</root>
```

contains two structural terms `</root/,test>` *and* `</,test>`*.*

Exercise 12/3

Consider the following the XML document:

```
<Course_Catalog>
  <Department Code="CS">
    <Title>Computer Science</Title>
    <Chair>
      <Professor>
        <First_Name>Jennifer</First_Name>
        <Last_Name>Widom</Last_Name>
      </Professor>
    </Chair>
    <Course Number="CS106A" Enrollment="1070">
      <Title>Programming Methodology</Title>
      <Description>Introduction to the engineering of computer applications
      emphasizing modern software engineering principles.
      </Description>
      <Instructors>
        <Lecturer>
          <First_Name>Jerry</First_Name>
          <Middle_Initial>R.</Middle_Initial>
          <Last_Name>Cain</Last_Name>
        </Lecturer>
        <Professor>
          <First_Name>Eric</First_Name>
          <Last_Name>Roberts</Last_Name>
        </Professor>
        <Professor>
          <First_Name>Mehran</First_Name>
          <Last_Name>Sahami</Last_Name>
        </Professor>
```

```
      </Instructors>
    </Course>
    <Course Number="CS106B" Enrollment="620">
      <Title>Programming Abstractions</Title>
      <Description>Abstraction and its relation to programming.</Description>
      <Instructors>
        <Professor>
          <First_Name>Eric</First_Name>
          <Last_Name>Roberts</Last_Name>
        </Professor>
        <Lecturer>
          <First_Name>Jerry</First_Name>
          <Middle_Initial>R.</Middle_Initial>
          <Last_Name>Cain</Last_Name>
        </Lecturer>
      </Instructors>
      <Prerequisites>
        <Prereq>CS106A</Prereq>
      </Prerequisites>
    </Course>
    </Department>
</Course_Catalog>
```

1. Write the following expressions:
   a) Return all titles (including both departments and courses).
   b) Return all course titles that contain the word *programming*.
   c) Return the surnames of all instructors teaching at least one course that contains the word *software* in its description.
   d) Return the surnames of all professors teaching at least one course that contains the word *software* in its description.
2. Calculate the similarity between the queries and the corresponding document paths.
   a) `//Instructors//Last_Name#Cain`
   b) `//Course/Instructors/Lecturer/Last_Name#Cain`

---

1. a) `//Title`
   b) `//Course//Title[contains(current(),'programming')]`
   c) `//Course[contains(Description,'software')]/Instructors//Last_Name`
   d) `//Course[contains(Description,'software')]/Instructors/Professor/Last_Name`

2. By Definition 21, a query c_q corresponds to document c_d if and only if it can be expanded. Original query for a) can be expanded to `Course_Catalog/Department/Course/Instructors/Lecturer/Last_Name`.
   Since c_q is expandable to c_d, use the equation from the definition. Substituting for the query length $c_q = 2$ and the document length $c_d = 6$ to the formula we get

$$CR(c_q, c_d) = \frac{1 + |c_q|}{1 + |c_d|} = \frac{1+2}{1+6} = \frac{3}{7}.$$

   For b) we only change the query length $c_q = 4$ and obtain

$$CR(c_q, c_d) = \frac{1 + |c_q|}{1 + |c_d|} = \frac{1+4}{1+6} = \frac{5}{7}.$$

Exercise 12/4

Count how many structural terms are present in the XML tree:

```
<Course>
  <Title>Programming Abstractions</Title>
  <Description>Abstraction and its relation to programming</Description>
  <Instructors>
    <Professor>
      <First_Name>Eric</First_Name>
      <Last_Name>Roberts</Last_Name>
    </Professor>
  </Instructors>
</Course>
```

To save space, mark each element with its first letter only (D stands for Description, P for Professor, ...). By Definition 22, we count all combinations and write them into the table:

C T Programming Abstractions	C I P F Eric	C I P L Roberts
T Programming Abstractions	I P F Eric	I P L Roberts
Programming Abstractions	P F Eric	P L Roberts
C D Abstraction ...	F Eric	L Roberts
D Abstraction ...	Eric	Roberts
Abstraction ...		

There are 16 structural terms in total.