Seminar 4

Definition 1 (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 (the document frequency of t) is the number of documents that contain t.

Definition 2 (tf-idf weighting scheme)
In the tf-idf weighting scheme, a term t in a document d has weight

$$tf\text{-}idf_{t,d} = tf_{t,d} \cdot idf_t$$

where $tf_{t,d}$ is the number of tokens t (the term frequency of t) in a document d.

Definition 3 (ℓ^2 (cosine) normalization)
A vector v is cosine-normalized by

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

where v_j is the element at the j-th position in v.

Definition 4 (Sublinear term frequency scaling)
The weight of a term t in a document d is determined as

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

where $tf_{t,d}$ is the number of tokens t (the term frequency of t) in a document d.

Algorithm 1 (Levenshtein Distance – imperative approach)

```
function LevenshteinDistance($s_1$, $s_2$)
    for $i$ = 0 to $|s_1|$ do
        $m[i, 0] = i$
    end for
    for $j$ = 0 to $|s_2|$ do
        $m[0, j] = j$
    end for
    for $i$ = 1 to $|s_1|$ do
        for $j$ = 1 to $|s_2|$ do
            if $s_1[i] == s_2[j]$ then
                $m[i, j] = \min\{m[i-1, j] + 1, m[i, j-1] + 1, m[i-1, j-1]\}$
            else
                $m[i, j] = \min\{m[i-1, j] + 1, m[i, j-1] + 1, m[i-1, j-1] + 1\}$
            end if
        end for
    end for
    return $m[|s_1|, |s_2|]$
end function
```

Algorithm 2 (Levenshtein Distance – declarative approach)
Distance between two strings a and b is given by $lev_{a,b}(|a|,|b|)$ *where*

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \textit{if } \min(i,j)=0 \\ \min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \textit{otherwise} \end{cases}$$

where $1_{(a_i \neq b_j)}$ *is the indicator function equal to 1 when* $a_i \neq b_j$*, and 0 otherwise.* $lev_{a,b}(i,j)$ *is the distance between the first i characters of string a and the first j characters of string b.*

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 1: Exercise.

After counting *tf-idf* weights by Definition 2 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 2: 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 3. 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 3: 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 4: *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 5: 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 6: 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 $N = 10000000$, sublinear term frequency scaling from Definition 4 (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 7: Exercise.

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

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

Sublinear term frequency scaling uses the Definition 4. 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 1.

$$\begin{aligned}
idf_{digital} &= \log\left(\tfrac{10^7}{10^4}\right) &&= \log\left(10^3\right) &&= 3\\
idf_{video} &= \log\left(\tfrac{10^7}{10^5}\right) &&= \log\left(10^2\right) &&= 2\\
idf_{cameras} &= \log\left(\tfrac{10^7}{5\times10^4}\right) &&= \log(200) &&= 2.301
\end{aligned}$$

and $q = w \cdot idf$. Cosine normalization for the document is counted similarly as in the last exercises by Definition 3 using w.

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

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 8: 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 9: 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 10: Exercise with queries.

Now we normalize the vectors q_i by Definition 3 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 11: 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.

Exercise 4/7

Compute the Levenshtein distance between *paris* and *alice*. Write down the matrix of distances between all prefixes as computed by the algorithm 1.

Follow the algorithm 1 and put one word horizontally and the other vertically into the matrix, both starting with ε (empty string). Then initialize the first rows and columns (see the lines 2 to 7 of the algorithm).

	ε	p	a	r	i	s
ε	0	1	2	3	4	5
a	1					
l	2					
i	3					
c	4					
e	5					

Table 12: Initialization of the matrix.

Then compare the horizontal prefixes (of word *paris*) with the vertical prefixes (of word *alice*), character by character, and fill in each cell of the matrix based on the values in the surrounding cells by the criterion in the algorithm. Select the minimum of numbers incremented by one in the cells up and left, and upper left incremented by one if the characters are not equal, without increment otherwise (a condition starting at line 10).

For example, for the cell **ap**, select $min\{1+1, 1+1, 0+\mathbf{1}\} = min\{2, 2, 1\} = 1$. The upper left value is $0 + \mathbf{1}$ because $a \neq p$. For the cell **aa**, select $min\{2+1, 1+1, 1+\mathbf{0}\} = min\{3, 2, 1\} = 1$. The upper left value is $1 + \mathbf{0}$ because $a = a$.

	ε	p	a	r	i	s
ε	0	1	2	3	4	5
a	1	1	1			
l	2					
i	3					
c	4					
e	5					

Table 13: First two iterations of the main dynamic programming step.

Now continue and fill out the whole matrix, the value in the bottom right cell is the Levenshtein distance for words *paris* and *alice*, which is 4.

	ε	p	a	r	i	s
ε	0	1	2	3	4	5
a	1	1	1	2	3	4
l	2	2	2	2	3	4
i	3	3	3	3	2	3
c	4	4	4	4	3	3
e	5	5	5	5	4	**4**

Table 14: The final matrix with the Levenshtein distance in bold.