Matrix decompositions & latent semantic indexing (Chapter 18)

Definition 1 (Latent semantic analysis (LSA))
Tf-idf term and document representations are high-dimensional and sparse. This poses computational problems and reduces recall: Terms that occur in documents that are similar but distinct will have dissimilar term representations even though the terms are similar. We can tackle our problem by mapping the term-document matrix C to a low-rank representation C_k with rank k that minimizes the Frobenius norm of $C - C_k$:

$$||C - C_k|| = \sqrt{\sum_t \sum_d (C - C_k)_{t,d}}$$

The Eckart-Young theorem states that $C_k = U\Sigma_k V^T$, where $C = U\Sigma V^T$ is the singular value decomposition (SVD) of C and Σ_k is Σ with only k largest diagonal entries retained.

$U\Sigma_k$ then gives us dense k-dimensional tf-idf *term representations, while $\Sigma_k V^T$ gives us dense k-dimensional* tf-idf *document representations. The parameter k is usually in small hundreds. LSA increases recall and may increase precision by making representations of similar terms more similar due to the low dimensionality.*

Exercise 18/1

Assume we have a term-document incidence matrix C for three terms (rows) and two documents (columns) with the following singular value decomposition (SVD) to $U\Sigma V^T$:

$$C = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}, U = \begin{pmatrix} -0.816 & 0.000 \\ -0.408 & -0.707 \\ -0.408 & 0.707 \end{pmatrix}, \Sigma = \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 1.000 \end{pmatrix}, V^T = \begin{pmatrix} -0.707 & -0.707 \\ 0.707 & -0.707 \end{pmatrix}.$$

Using a Google Colaboratory notebook, compute:

a) A rank 1 representation C_1 of C,

b) 1-dimensional document representation, and

c) 1-dimensional term representation.

For a), we construct the rank 1 representation C_1 as follows:

$$C_1 = U\Sigma_1 V^T = \begin{pmatrix} 1 & 1 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}, \text{ where } \Sigma_1 = \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 0.000 \end{pmatrix}.$$

For b) and c), we construct 1-dimensional representations for the two documents and the three terms as follows:

$$\Sigma_1 V^T = \begin{pmatrix} -1.22 & -1.22 \end{pmatrix}, U\Sigma_1 = \begin{pmatrix} -1.41 & -0.71 & -0.71 \end{pmatrix}^T$$

Distributed Representations (Chapter 18)

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

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

where $\textit{tf}_{t,d}$ 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 (tf-idf term representation)
Just as a document d can be represented as a vector of tf-idf *weights of terms t, a term t can be represented as a vector of weights of t in documents d. If we represent our document collection as a term-document matrix $C_{t,d} = \text{tf-idf}(t, d)$ (see Definition 2), then* tf-idf *term representations correspond to the rows and* tf-idf *document representations correspond to the columns.*

Definition 5 (Word2Vec)
Word2Vec is a neural network language model. Given terms t_1 and t_2, Word2vec predicts the probability $p(t_1 \mid t_2)$ of term t_1 appearing in the context window of size surrounding term t_2. Word2Vec is trained on a text corpus to maximize the probabilities of terms that appear in context. As a side product, word2vec produces dense k-dimensional term representations. The parameter k is usually in low hundreds.

Definition 6 (Soft vector space model)
The tf-idf *document representation with the scalar product as the similarity score underestimates the similarity of documents that use similar but distinct terms. Replacing the scalar product*

$$\textit{score}(d, q) = d^T \cdot q = \sum_{i=1}^{T} (d_i \cdot q_i),$$

where d is a tf-idf *document representation, q is a* tf-idf *query representation, and T is the number of terms, with the soft scalar product*

$$\textit{score}(d, q) = d^T \cdot S \cdot q = \sum_{i=1}^{T} \sum_{j=1}^{T} (d_i \cdot S_{i,j} \cdot q_j),$$

where $S_{i,j}$ is the similarity of terms i and j, solves this problem.

Algorithm 1 (Levenshtein Distance – declarative approach)
Distance between two strings a and b is given by $\textit{lev}_{a,b}(|a|, |b|)$ where

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

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

Exercise 18/2

Consider the Euclidean normalized *tf-idf* weights from Exercises 6/1 through 6/3.

a) What are the *tf-idf* representations of terms *car, auto, insurance,* and *best*?

b) What is the similarity score (scalar product) between the *tf-idf* representations of terms *car* and *auto*?

c) What is the similarity score (scalar product) between the *tf-idf* representations of documents doc_1 and doc_2?

d) What is the similarity score (soft scalar product) between the *tf-idf* representations of documents doc_1 and doc_2 if we use the vector dot product of *tf-idf* term representations as the term similarity S? Use Google Colaboratory to perform the computations.

e) What is the similarity score (soft scalar product) between the *tf-idf* representations of documents doc_1 and doc_2 if we use the inverse of the Levenshtein distance (see Algorithm 1) as the term similarity S? Use Google Colaboratory to perform the computations.

f) What is the similarity score (soft scalar product) between the *tf-idf* representations of documents doc_1 and doc_2 if we use the vector dot product of Word2Vec representations as the term similarity S? Use Google Colaboratory to train a Word2Vec model and to perform the computations.

For **a)**, normalized Euclidean weight vectors are counted by Definition 3. Denominators m_{doc_n} for the individual terms are

$$m_{\text{car}} = \sqrt{44.55^2 + 6.6^2 + 39.6^2} = 59.97$$
$$m_{\text{auto}} = \sqrt{6.24^2 + 68.64^2 + 0^2} = 68.92$$
$$m_{\text{insurance}} = \sqrt{0^2 + 53.46^2 + 46.98^2} = 71.17$$
$$m_{\text{best}} = \sqrt{21^2 + 0^2 + 25.5^2} = 33.03$$

and the term representations are

$$t_1 = \left(\frac{44.55}{59.97}; \frac{6.6}{59.97}; \frac{39.6}{59.97}\right) = (0.7429; 0.1101; 0.6603)$$
$$t_2 = \left(\frac{6.24}{68.92}; \frac{68.64}{68.92}; \frac{0}{68.92}\right) = (0.0905, 0.9959, 0)$$
$$t_3 = \left(\frac{0}{71.17}; \frac{53.46}{71.17}; \frac{46.98}{71.17}\right) = (0, 0.7512, 0.6601)$$
$$t_4 = \left(\frac{21}{33.03}; \frac{0}{33.03}; \frac{25.5}{33.03}\right) = (0.6357, 0, 0.7719)$$

For **b)**, we multiply t_1 with t_2 and we obtain the similarity score:

$$t_1 \cdot t_2 = 0.7429 \cdot 0.0905 + 0.1101 \cdot 0.9959 + 0.6603 \cdot 0 = 0.1769$$

For **c)**, we multiply d_1 with d_2 and we obtain the similarity score:

$$d_1 \cdot d_2 = 0.8974 \cdot 0.0756 + 0.1257 \cdot 0.7876 + 0 \cdot 0.6127 + 0.423 \cdot 0 = 0.1668$$

For **d)** through **f)**, see the Google Colaboratory code.

Exercise 18/3

Consider the *tf* representations of the following two documents:

- *obama speaks to the media in illinois*
- *the president greets the press in chicago*

What is the similarity score of the two documents if we use

a) the scalar product,

b) the soft scalar product using the following term similarity S:

	obama	speaks	to	the	media	in	illinois	president	greets	press	chicago
obama	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	**0.7**	0.0	0.0	0.0
speaks	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
to	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
the	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
media	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	**0.8**	0.0
in	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0
illinois	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	**0.2**
president	**0.7**	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0
greets	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0
press	0.0	0.0	0.0	0.0	**0.8**	0.0	0.0	0.0	0.0	**1.0**	0.0
chicago	0.0	0.0	0.0	0.0	0.0	0.0	**0.2**	0.0	0.0	0.0	**1.0**

Use Google Colaboratory to perform the computations.

For **a)**, *tf* weight vectors are

$$d_1 = (1; 1; 1; 1; 1; 1; 1; 0; 0; 0; 0)\,,$$

$$d_2 = (0; 0; 0; 2; 0; 1; 0; 1; 1; 1; 1)\,.$$

We multiply d_1 with d_2 and we obtain the similarity score:

$$d_1 \cdot d_2 = 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 2 + 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = 2 + 1 = 3$$

For **b)**, see the Google Colaboratory code.