# Implementation Notes for the Soft Cosine Measure

Vít Novotný
Masaryk University
Faculty of Informatics
Brno, Czech Republic
witiko@mail.muni.cz

## ABSTRACT

The standard bag-of-words vector space model (VSM) is efficient, and ubiquitous in information retrieval, but it underestimates the similarity of documents with the same meaning, but different terminology. To overcome this limitation, Sidorov et al. [14] proposed the Soft Cosine Measure (SCM) that incorporates term similarity relations. Charlet and Damnati [2] showed that the SCM is highly effective in question answering (QA) systems. However, the orthonormalization algorithm proposed by Sidorov et al. [14] has an impractical time complexity of $O(n^4)$, where $n$ is the size of the vocabulary.

In this paper, we prove a tighter lower worst-case time complexity bound of $O(n^3)$. We also present an algorithm for computing the similarity between documents and we show that its worst-case time complexity is $O(1)$ given realistic conditions. Lastly, we describe implementation in general-purpose vector databases such as Annoy, and Faiss and in the inverted indices of text search engines such as Apache Lucene, and ElasticSearch. Our results enable the deployment of the SCM in real-world information retrieval systems.

## KEYWORDS

Vector Space Model, computational complexity, similarity measure

**ACM Reference Format:**
Vít Novotný. 2018. Implementation Notes for the Soft Cosine Measure. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269317

## 1 INTRODUCTION

The standard bag-of-words vector space model (VSM) [13] represents documents as real vectors. Documents are expressed in a basis where each basis vector corresponds to a single term, and each coordinate corresponds to the frequency of a term in a document. Consider the documents

$d_1$ = "When Antony found **Julius Caesar** dead", and

$d_2$ = "I did enact **Julius Caesar**: I was killed i' the Capitol"

represented in a basis $\{\boldsymbol{\alpha}_i\}_{i=1}^{14}$ of $\mathbb{R}^{14}$, where the basis vectors correspond to the terms in the order of first appearance. Then the corresponding document vectors $\mathbf{v}_1$, and $\mathbf{v}_2$ would have the following coordinates in $\boldsymbol{\alpha}$:

$$(\mathbf{v}_1)_{\boldsymbol{\alpha}} = [1\,1\,1\,\mathbf{1}\,\mathbf{1}\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0]^\mathsf{T}, \text{ and}$$
$$(\mathbf{v}_2)_{\boldsymbol{\alpha}} = [0\,0\,0\,\mathbf{1}\,\mathbf{1}\,0\,2\,1\,1\,1\,1\,1\,1\,1]^\mathsf{T}.$$

Assuming $\boldsymbol{\alpha}$ is orthonormal, we can take the inner product of the $\ell^2$-normalized vectors $\mathbf{v}_1$, and $\mathbf{v}_2$ to measure the cosine of the angle (i.e. the *cosine similarity*) between the documents $d_1$, and $d_2$:

$$\langle \mathbf{v}_1/\|\mathbf{v}_1\|, \mathbf{v}_2/\|\mathbf{v}_2\| \rangle = \frac{\big((\mathbf{v}_1)_{\boldsymbol{\alpha}}\big)^\mathsf{T}(\mathbf{v}_2)_{\boldsymbol{\alpha}}}{\sqrt{\big((\mathbf{v}_1)_{\boldsymbol{\alpha}}\big)^\mathsf{T}(\mathbf{v}_1)_{\boldsymbol{\alpha}}}\sqrt{\big((\mathbf{v}_2)_{\boldsymbol{\alpha}}\big)^\mathsf{T}(\mathbf{v}_2)_{\boldsymbol{\alpha}}}} \approx 0.23.$$

Intuitively, this underestimates the true similarity between $d_1$, and $d_2$. Assuming $\boldsymbol{\alpha}$ is orthogonal but not orthonormal, and that the terms Julius, and Caesar are twice as important as the other terms, we can construct a diagonal change-of-basis matrix $\mathbf{W} = (w_{ij})$ from $\boldsymbol{\alpha}$ to an orthonormal basis $\boldsymbol{\beta}$, where $w_{ii}$ corresponds to the importance of a term $i$. This brings us closer to the true similarity:

$$(\mathbf{v}_1)_{\boldsymbol{\beta}} = \mathbf{W}(\mathbf{v}_1)_{\boldsymbol{\alpha}} = [1\,1\,1\,\mathbf{2}\,\mathbf{2}\,1\,0\,0\,0\,0\,0\,0\,0\,0]^\mathsf{T},$$
$$(\mathbf{v}_2)_{\boldsymbol{\beta}} = \mathbf{W}(\mathbf{v}_2)_{\boldsymbol{\alpha}} = [0\,0\,0\,\mathbf{2}\,\mathbf{2}\,0\,2\,1\,1\,1\,1\,1\,1\,1]^\mathsf{T}, \text{ and}$$

$$\langle \mathbf{v}_1/\|\mathbf{v}_1\|, \mathbf{v}_2/\|\mathbf{v}_2\| \rangle = \frac{\big(\mathbf{W}(\mathbf{v}_1)_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{W}(\mathbf{v}_2)_{\boldsymbol{\alpha}}}{\sqrt{\big(\mathbf{W}(\mathbf{v}_1)_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{W}(\mathbf{v}_1)_{\boldsymbol{\alpha}}}\sqrt{\big(\mathbf{W}(\mathbf{v}_2)_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{W}(\mathbf{v}_2)_{\boldsymbol{\alpha}}}} \approx 0.53.$$

Since we assume that the bases $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are orthogonal, the terms dead and killed contribute nothing to the cosine similarity despite the clear synonymy, because $\langle \boldsymbol{\beta}_{\text{dead}}, \boldsymbol{\beta}_{\text{killed}} \rangle = 0$. In general, the VSM will underestimate the true similarity between documents that carry the same meaning but use different terminology.

In this paper, we further develop the soft VSM described by Sidorov et al. [14], which does not assume $\boldsymbol{\alpha}$ is orthogonal and which achieved state-of-the-art results on the question answering (QA) task at SemEval 2017 [2]. In Section 2, we review the previous work incorporating term similarity into the VSM. In Section 3, we restate the definition of the soft VSM and present several computational complexity results. In Section 4, we describe the implementation in vector databases and inverted indices. We conclude in Section 5 by summarizing our results and suggesting future work.

## 2 RELATED WORK

Most works incorporating term similarity into the VSM published prior to Sidorov et al. [14] remain in an orthogonal coordinate system and instead propose novel document similarity measures. To name a few, Mikawa et al. [8] proposes the *extended cosine measure*, which introduces a metric matrix $\mathbf{Q}$ as a multiplicative factor in the cosine similarity formula. $\mathbf{Q}$ is the solution of an optimization

*CIKM '18, October 22–26, 2018, Torino, Italy*

problem to maximize the sum of extended cosine measures between each vector and the centroid of the vector's category. Conveniently, the metric matrix $\mathbf{Q}$ can be used directly with the soft VSM, where it defines the inner product between basis vectors. Jimenez et al. [6] equip the multiset VSM with a *soft cardinality* operator that corresponds to cardinality, but takes term similarities into account.

The notion of generalizing the VSM to non-orthogonal coordinate systems was perhaps first explored by Sidorov et al. [14] in the context of entrance exam question answering, where the basis vectors did not correspond directly to terms, but to $n$-grams constructed by following paths in syntactic trees. The authors derive the inner product of two basis vectors from the edit distance between the corresponding $n$-grams. *Soft cosine measure* (SCM) is how they term the formula for computing the cosine similarity between two vectors expressed in a non-orthogonal basis. They also present an algorithm that computes a change-of-basis matrix to an orthonormal basis in time $O(n^4)$. We present an $O(n^3)$ algorithm in this paper.

Charlet and Damnati [2] achieved state-of-the-art results at the QA task at SemEval 2017 [10] by training a document classifier on soft cosine measures between document passages. Unlike Sidorov et al. [14], Charlet and Damnati [2] already use basis vectors that correspond to terms rather than to $n$-grams. They derive the inner product of two basis vectors both from the edit distance between the corresponding terms, and from the inner product of the corresponding word2vec term embeddings [9].

## 3 COMPUTATIONAL COMPLEXITY

In this section, we restate the definition of the soft VSM as it was described by Sidorov et al. [14]. We then prove a tighter lower worst-case time complexity bound for computing a change-of-basis matrix to an orthonormal basis. We also prove that under certain assumptions, the inner product is a linear-time operation.

*Definition 3.1.* Let $\mathbb{R}^n$ be the real $n$-space over $\mathbb{R}$ equipped with the bilinear inner product $\langle \cdot, \cdot \rangle$. Let $\{\boldsymbol{\alpha}_i\}_{i=1}^n$ be the basis of $\mathbb{R}^n$ in which we express our vectors. Let $\mathbf{W}_{\boldsymbol{\alpha}} = (w_{ij})$ be a diagonal change-of-basis matrix from $\boldsymbol{\alpha}$ to a normalized basis $\{\boldsymbol{\beta}_i\}_{i=1}^n$ of $\mathbb{R}^n$, i.e. $\langle \boldsymbol{\beta}_i, \boldsymbol{\beta}_j \rangle \in [-1, 1]$, $\langle \boldsymbol{\beta}_i, \boldsymbol{\beta}_i \rangle = 1$. Let $\mathbf{S}_{\boldsymbol{\beta}} = (s_{ij})$ be the metric matrix of $\mathbb{R}^n$ w.r.t. $\boldsymbol{\beta}$, i.e. $s_{ij} = \langle \boldsymbol{\beta}_i, \boldsymbol{\beta}_j \rangle$. Then $(\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\beta})$ is a *soft VSM*.

THEOREM 3.2. *Let $G = (\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\boldsymbol{\beta}})$ be a soft VSM. Then a change-of-basis matrix $\mathbf{E}$ from the basis $\boldsymbol{\beta}$ to an orthonormal basis of $\mathbb{R}^n$ can be computed in time $O(n^3)$.*

PROOF. By definition, $\mathbf{S} = \mathbf{E}\mathbf{E}^\mathsf{T}$ for any change-of-basis matrix $\mathbf{E}$ from the basis $\boldsymbol{\beta}$ to an orthonormal basis. Since $\mathbf{S}$ contains inner products of linearly independent vectors $\boldsymbol{\beta}$, it is Gramian and positive definite [5, p. 441]. The Gramianness of $\mathbf{S}$ also implies its symmetry. Therefore, a lower triangular $\mathbf{E}$ is uniquely determined by the Cholesky factorization of the symmetric positive-definite $\mathbf{S}$, which we can compute in time $O(n^3)$ [15, p. 191]. □

*Remark.* See Table 1 for an experimental comparison.

Although the vocabulary in our introductory example contains only $n = 14$ terms, $n$ is in the millions for real-world corpora such as the English Wikipedia. Therefore, we generally need to store the $n \times n$ matrix $\mathbf{S}$ in a sparse format, so that it fits into main memory. Later, we will discuss how the density of $\mathbf{S}$ can be reduced, but

**Table 1:** The real time to compute a matrix $\mathbf{E}$ from a dense matrix $\mathbf{S}$ averaged over 100 iterations. We used two Intel Xeon E5-2650 v2 (20M cache, 2.60 GHz) processors to evaluate the $O(n^3)$ Cholesky factorization from NumPy 1.14.3, and the $O(n^4)$ iterated Gaussian elimination from LAPACK. For $n > 1000$, only sparse $\mathbf{S}$ seem practical.

| $n$ terms | Algorithm | Real computation time |
|---|---|---|
| 100 | Cholesky factorization | 0.0006 sec (0.606 ms) |
| 100 | Gaussian elimination | 0.0529 sec (52.893 ms) |
| 500 | Cholesky factorization | 0.0086 sec (8.640 ms) |
| 500 | Gaussian elimination | 22.7361 sec (22.736 sec) |
| 1000 | Cholesky factorization | 0.0304 sec (30.378 ms) |
| 1000 | Gaussian elimination | 354.2746 sec (5.905 min) |

the Cholesky factor $\mathbf{E}$ can also be arbitrarily dense and therefore expensive to store. Given a permutation matrix $\mathbf{P}$, we can instead factorize $\mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}$ into $\mathbf{F}\mathbf{F}^\mathsf{T}$. Finding the permutation matrix $\mathbf{P}$ that minimizes the density of the Cholesky factor $\mathbf{F}$ is NP-hard [16], but heuristic stategies are known [3, 4]. Using the fact that $\mathbf{P}^\mathsf{T} = \mathbf{P}^{-1}$, and basic facts about transpose, we can derive $\mathbf{E} = \mathbf{P}\mathbf{F}$ as follows: $\mathbf{S} = \mathbf{P}\mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}\mathbf{P}^\mathsf{T} = \mathbf{P}\mathbf{F}\mathbf{F}^\mathsf{T}\mathbf{P}^\mathsf{T} = \mathbf{P}\mathbf{F}(\mathbf{P}\mathbf{F})^\mathsf{T} = \mathbf{E}\mathbf{E}^\mathsf{T}$.

LEMMA 3.3. *Let $G = (\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\boldsymbol{\beta}})$ be a soft VSM. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Then $\langle \mathbf{x}, \mathbf{y} \rangle = (\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}})^\mathsf{T}\mathbf{S}\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}}$.*

PROOF. Let $\mathbf{E}$ be the change-of-basis matrix from the basis $\boldsymbol{\beta}$ to an orthonormal basis $\boldsymbol{\gamma}$ of $\mathbb{R}^n$. Then:

$$\begin{aligned}\langle \mathbf{x}, \mathbf{y} \rangle &= \big((\mathbf{x})_{\boldsymbol{\gamma}}\big)^\mathsf{T}(\mathbf{y})_{\boldsymbol{\gamma}} = \big(\mathbf{E}(\mathbf{x})_{\boldsymbol{\beta}}\big)^\mathsf{T}\mathbf{E}(\mathbf{y})_{\boldsymbol{\beta}} = \big(\mathbf{E}\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{E}\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}} \\ &= \left(\sum_{i=1}^n (\boldsymbol{\alpha}_i)_{\boldsymbol{\gamma}} \cdot w_{ii} \cdot (x_i)_{\boldsymbol{\alpha}}\right) \cdot \left(\sum_{j=1}^n (\boldsymbol{\alpha}_j)_{\boldsymbol{\gamma}} \cdot w_{jj} \cdot (y_j)_{\boldsymbol{\alpha}}\right) \\ &= \sum_{i=1}^n \sum_{j=1}^n w_{ii} \cdot (x_i)_{\boldsymbol{\alpha}} \cdot \langle \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_j \rangle \cdot w_{jj} \cdot (y_j)_{\boldsymbol{\alpha}} \\ &= \sum_{i=1}^n \sum_{j=1}^n w_{ii} \cdot (x_i)_{\boldsymbol{\alpha}} \cdot s_{ij} \cdot w_{jj} \cdot (y_j)_{\boldsymbol{\alpha}} = \big(\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{S}\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}}. \quad \square\end{aligned}$$

*Remark.* From here, we can directly derive the cosine of the angle between $\mathbf{x}$ and $\mathbf{y}$ (i.e. what Sidorov et al. [14] call the SCM) as follows:

$$\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle = \frac{\big(\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{S}\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}}}{\sqrt{\big(\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{S}\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}}\sqrt{\big(\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}}\big)^\mathsf{T}\mathbf{S}\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}}}}.$$

The SCM is actually the starting point for Charlet and Damnati [2], who propose matrices $\mathbf{S}$ that are not necessarily metric. If, like them, we are only interested in computing the SCM, then we only require that the square roots remain real, i.e. that $\mathbf{x} \neq 0 \implies (\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}})^\mathsf{T}\mathbf{S}\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}} \geq 0$. For arbitrary $\mathbf{x} \in \mathbb{R}^n$, this holds iff $\mathbf{S}$ is positive semi-definite. However, since the coordinates $(\mathbf{x})_{\boldsymbol{\alpha}}$ correspond to non-negative term frequencies, it is sufficient that $\mathbf{W}$ and $\mathbf{S}$ are non-negative as well. If we are only interested in computing the inner product, then $\mathbf{S}$ can be arbitrary.

THEOREM 3.4. *Let $G = (\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\boldsymbol{\beta}})$ be a soft VSM such that no column of $\mathbf{S}$ contains more than $C$ non-zero elements, where $C$ is a constant. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and let $m$ be the number of non-zero elements in $(\mathbf{x})_{\boldsymbol{\beta}}$. Then $\langle \mathbf{x}, \mathbf{y} \rangle$ can be computed in time $O(m)$.*

Proof. Assume that $(\mathbf{x})_{\boldsymbol{\alpha}}, (\mathbf{y})_{\boldsymbol{\alpha}}$, and $\mathbf{S}$ are represented by data structures with constant-time column access and non-zero element traversal, e.g. compressed sparse column (CSC) matrices. Further assume that $\mathbf{W}$ is represented by an array containing the main diagonal of $\mathbf{W}$. Then Algorithm 1 computes $\left(\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}\right)^{\mathsf{T}}\mathbf{S}\mathbf{W}(\mathbf{y})_{\boldsymbol{\alpha}}$ in time $O(m)$, which by Lemma 3.3, corresponds to $\langle \mathbf{x}, \mathbf{y} \rangle$. □

**Algorithm 1** The inner product of $\mathbf{x}$ and $\mathbf{y}$

```
r ← 0
for each i such that (x_i)_α is non-zero do        ▷ = m iterations
    for each j such that s_ij is non-zero do       ▷ ≤ C iterations
        r ← r + w_ii · (x_i)_α · s_ij · w_jj · (y_j)_α
return r
```

*Remark.* Similarly, we can show that if a column of $\mathbf{S}$ contains $C$ non-zero elements on average, $\langle \mathbf{x}, \mathbf{y} \rangle$ has the average-case time complexity of $O(m)$. Note also that most information retrieval systems impose a limit on the length of a query document. Therefore, $m$ is usually bounded by a constant and $O(m) = O(1)$.

Since we are usually interested in the inner products of all document pairs in two corpora (e.g. one containing queries and the other actual documents), we can achieve significant speed improvements with vector processors by computing $(\mathbf{WX})^{\mathsf{T}}\mathbf{SWY}$, where $\mathbf{X}$, and $\mathbf{Y}$ are *corpus matrices* containing the coordinates of document vectors in the basis $\boldsymbol{\alpha}$ as columns. To compute the SCM, we first need to normalize the document vectors by performing an entrywise division of every column in $\mathbf{X}$ by $\operatorname{diag}\sqrt{(\mathbf{WX})^{\mathsf{T}}\mathbf{SWX}} = \sqrt{(\mathbf{WX})^{\mathsf{T}}\mathbf{S} \circ (\mathbf{WX})^{\mathsf{T}}}$, where $\circ$ denotes entrywise product. $\mathbf{Y}$ is normalized analogously.

There are several strategies for making no column of $\mathbf{S}$ contain more than $C$ non-zero elements. If we do not require that $\mathbf{S}$ is metric (e.g. because we only wish to compute the inner product, or the SCM), a simple strategy is to start with an empty matrix, and to insert the $C-1$ largest elements and the diagonal element from every column of $\mathbf{S}$. However, the resulting matrix will likely be asymmetric, which makes the inner product formula asymmetric as well. We can regain symmetry by always inserting an element $s_{ij}$ together with the element $s_{ji}$ and only if this does not make the column $j$ contain more than $C$ non-zero elements. This strategy is greedy, since later columns contain non-zero elements inserted by earlier columns. Our preliminary experiments suggest that processing colums that correspond to increasingly frequent terms performs best on the task of Charlet and Damnati [2]. Finally, by limiting the sum of all non-diagonal elements in a column to be less than one, we can make $\mathbf{S}$ strictly diagonally dominant and therefore positive definite, which enables us to compute $\mathbf{E}$ through Cholesky factorization.

## 4 IMPLEMENTATION IN VECTOR DATABASES AND INVERTED INDICES

In this section, we present coordinate transformations for retrieving nearest document vectors according to the inner product, and the soft cosine measure from general-purpose vector databases such as Annoy, or Faiss [7]. We also discuss the implementation in the inverted indices of text search engines such as Apache Lucene [1].

*Remark.* With a vector database, we can transform document vectors to an orthonormal basis $\boldsymbol{\gamma}$. In the transformed coordinates, the dot product $((\mathbf{x})_{\boldsymbol{\gamma}})^{\mathsf{T}}(\mathbf{y})_{\boldsymbol{\gamma}}$ corresponds to the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ and the cosine similarity corresponds to the cosine of an angle $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle$ (i.e. the soft cosine measure). A vector database that supports nearest neighbor search according to either the dot product, or the cosine similarity will therefore retrieve vectors expressed in $\boldsymbol{\gamma}$ according to either the inner product, or the soft cosine measure. We can compute a change-of-basis matrix $\mathbf{E}$ of order $n$ in time $O(n^3)$ by Theorem 3.2 and use it to transform every vector $\mathbf{x} \in \mathbb{R}^n$ to $\boldsymbol{\gamma}$ by computing $\mathbf{E}\mathbf{W}(\mathbf{x})_{\boldsymbol{\alpha}}$. However, this approach requires that $\mathbf{S}$ is symmetric positive-definite and that we recompute $\mathbf{E}$, and reindex the vector database each time $\mathbf{S}$ has changed. We will now discuss transformations that do not require $\mathbf{E}$ and for which a non-negative $\mathbf{S}$ is sufficient as discussed in the remark for Lemma 3.3.

Theorem 4.1. *Let* $G = (\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\boldsymbol{\beta}})$ *be a soft VSM. Let* $\mathbf{x}, \mathbf{x}', \mathbf{y} \in \mathbb{R}^n$ *such that* $(\mathbf{x}')_{\boldsymbol{\beta}} = \mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\boldsymbol{\beta}}$. *Then* $\langle \mathbf{x}, \mathbf{y} \rangle = ((\mathbf{x}')_{\boldsymbol{\beta}})^{\mathsf{T}}(\mathbf{y})_{\boldsymbol{\beta}}$.

Proof. $\left((\mathbf{x}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}}(\mathbf{y})_{\boldsymbol{\beta}} = \left((\mathbf{x})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\boldsymbol{\beta}} = \langle \mathbf{x}, \mathbf{y} \rangle$ from Lemma 3.3. □

*Remark.* By transforming a query vector $\mathbf{x}$ into $(\mathbf{x}')_{\boldsymbol{\beta}}$, we can retrieve documents according to the inner product in vector databases that only support nearest neighbor search according to the dot product. Note that we do not introduce $\mathbf{S}$ into $(\mathbf{y})_{\boldsymbol{\beta}}$, which allows us to change $\mathbf{S}$ without changing the documents in a vector database and that $\mathbf{S}$ can be arbitrary as discussed in the remark for Lemma 3.3.

Theorem 4.2. *Let* $G = (\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\boldsymbol{\beta}})$ *be a soft VSM. Let* $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}' \in \mathbb{R}^n$ *s.t.* $\mathbf{x}, \mathbf{y}, \mathbf{z} \neq 0, (\mathbf{x}')_{\boldsymbol{\beta}} = \mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\boldsymbol{\beta}}, (\mathbf{y}')_{\boldsymbol{\beta}} = \frac{(\mathbf{y})_{\boldsymbol{\beta}}}{\sqrt{\left((\mathbf{y})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\boldsymbol{\beta}}}}$, *and* $(\mathbf{z}')_{\boldsymbol{\beta}} = \frac{(\mathbf{z})_{\boldsymbol{\beta}}}{\sqrt{\left((\mathbf{z})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}(\mathbf{z})_{\boldsymbol{\beta}}}}$. *Then* $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle \leq \langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{z}/\|\mathbf{z}\| \rangle$ *iff* $\left((\mathbf{x}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}}(\mathbf{y}')_{\boldsymbol{\beta}} \leq \left((\mathbf{x}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}}(\mathbf{z}')_{\boldsymbol{\beta}}$.

Proof. $\left((\mathbf{x}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}}(\mathbf{y}')_{\boldsymbol{\beta}} = \frac{((\mathbf{x})_{\boldsymbol{\beta}})^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\boldsymbol{\beta}}}{\sqrt{((\mathbf{y})_{\boldsymbol{\beta}})^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\boldsymbol{\beta}}}}$. From Lemma 3.3, this equals $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle$ except for the missing term $\sqrt{\left((\mathbf{x})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}(\mathbf{x})_{\boldsymbol{\beta}}}$ in the divisor. The term is constant in both $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle$, and $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{z}/\|\mathbf{z}\| \rangle$, so ordering is preserved. □

*Remark.* By transforming a query vector $\mathbf{x}$ into $(\mathbf{x}')_{\boldsymbol{\beta}}$ and document vectors $\mathbf{y}$ into $(\mathbf{y}')_{\boldsymbol{\beta}}$, we can retrieve documents according to the SCM in vector databases that only support nearest neighbor search according to the dot product.

Theorem 4.3. *Let* $G = (\mathbb{R}^n, \mathbf{W}_{\boldsymbol{\alpha}}, \mathbf{S}_{\boldsymbol{\beta}})$ *be a soft VSM s.t.* $\mathbf{S}_{\boldsymbol{\beta}}$ *is non-negative. Let* $\mathbf{x}, \mathbf{y}, \mathbf{y}', \mathbf{z}, \mathbf{z}' \in \mathbb{R}^n$, *and* $\mathbf{x}', \mathbf{y}'', \mathbf{z}'' \in \mathbb{R}^{n+1}$ *s.t.* $\mathbf{x} \neq 0, \mathbf{y}, \mathbf{z} > 0, (\mathbf{x}')_{\boldsymbol{\beta}'} = \begin{bmatrix} \frac{\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\boldsymbol{\beta}}}{\sqrt{\left(\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\boldsymbol{\beta}}}} & 0 \end{bmatrix}^{\mathsf{T}}, (\mathbf{y}')_{\boldsymbol{\beta}} = \frac{(\mathbf{y})_{\boldsymbol{\beta}}}{\sqrt{\left((\mathbf{y})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\boldsymbol{\beta}}}}$, $(\mathbf{y}'')_{\boldsymbol{\beta}'} = \begin{bmatrix} \left((\mathbf{y}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}} & \sqrt{1 - \left((\mathbf{y}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}}(\mathbf{y}')_{\boldsymbol{\beta}}} \end{bmatrix}^{\mathsf{T}}, (\mathbf{z}')_{\boldsymbol{\beta}} = \frac{(\mathbf{z})_{\boldsymbol{\beta}}}{\sqrt{\left((\mathbf{z})_{\boldsymbol{\beta}}\right)^{\mathsf{T}}\mathbf{S}(\mathbf{z})_{\boldsymbol{\beta}}}}$, *and* $(\mathbf{z}'')_{\boldsymbol{\beta}'} = \begin{bmatrix} \left((\mathbf{z}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}} & \sqrt{1 - \left((\mathbf{z}')_{\boldsymbol{\beta}}\right)^{\mathsf{T}}(\mathbf{z}')_{\boldsymbol{\beta}}} \end{bmatrix}^{\mathsf{T}}$, *where* $\boldsymbol{\beta}' = \boldsymbol{\beta} \cup \{[0 \ldots 0\ 1]^{\mathsf{T}} \in \mathbb{R}^{n+1}\}$. *Then* $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle \leq \langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{z}/\|\mathbf{z}\| \rangle$ *iff* $\frac{\left((\mathbf{x}')_{\boldsymbol{\beta}'}\right)^{\mathsf{T}}(\mathbf{y}'')_{\boldsymbol{\beta}'}}{\sqrt{\left((\mathbf{x}')_{\boldsymbol{\beta}'}\right)^{\mathsf{T}}(\mathbf{x}')_{\boldsymbol{\beta}'}}\sqrt{\left((\mathbf{y}'')_{\boldsymbol{\beta}'}\right)^{\mathsf{T}}(\mathbf{y}'')_{\boldsymbol{\beta}'}}} \leq \frac{\left((\mathbf{x}')_{\boldsymbol{\beta}'}\right)^{\mathsf{T}}(\mathbf{z}'')_{\boldsymbol{\beta}'}}{\sqrt{\left((\mathbf{x}')_{\boldsymbol{\beta}'}\right)^{\mathsf{T}}(\mathbf{x}')_{\boldsymbol{\beta}'}}\sqrt{\left((\mathbf{z}'')_{\boldsymbol{\beta}'}\right)^{\mathsf{T}}(\mathbf{z}'')_{\boldsymbol{\beta}'}}}$.

Proof. $\left((\mathbf{x}')_{\beta'}\right)^{\mathsf{T}}(\mathbf{x}')_{\beta'} = 1$. Since $\mathbf{S}$ is non-negative, and $(\mathbf{y})_{\beta} > 0$, $\sqrt{((\mathbf{y})_{\beta})^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\beta}} \geq \sqrt{((\mathbf{y})_{\beta})^{\mathsf{T}}(\mathbf{y})_{\beta}}$ and therefore $\left((\mathbf{y}')_{\beta'}\right)^{\mathsf{T}}(\mathbf{y}')_{\beta'} \leq 1$, and $\left((\mathbf{y}'')_{\beta'}\right)^{\mathsf{T}}(\mathbf{y}'')_{\beta'} = 1$ [11, sec. 4.2]. Therefore:

$$\frac{\left((\mathbf{x}')_{\beta'}\right)^{\mathsf{T}}(\mathbf{y}'')_{\beta'}}{\sqrt{\left((\mathbf{x}')_{\beta'}\right)^{\mathsf{T}}(\mathbf{x}')_{\beta'}}\sqrt{\left((\mathbf{y}'')_{\beta'}\right)^{\mathsf{T}}(\mathbf{y}'')_{\beta'}}} = \left((\mathbf{x}')_{\beta'}\right)^{\mathsf{T}}(\mathbf{y}'')_{\beta'}$$

$$= \frac{((\mathbf{x})_{\beta})^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\beta}}{\sqrt{\left(\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\beta}\right)^{\mathsf{T}}\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\beta}}\sqrt{((\mathbf{y})_{\beta})^{\mathsf{T}}\mathbf{S}(\mathbf{y})_{\beta}}}.$$

From Lemma 3.3, this equals $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle$ except for the missing term $\sqrt{((\mathbf{x})_{\beta})^{\mathsf{T}}\mathbf{S}(\mathbf{x})_{\beta}}$, and the extra term $\sqrt{\left(\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\beta}\right)^{\mathsf{T}}\mathbf{S}^{\mathsf{T}}(\mathbf{x})_{\beta}}$ in the divisor. The terms are constant in both $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{y}/\|\mathbf{y}\| \rangle$, and $\langle \mathbf{x}/\|\mathbf{x}\|, \mathbf{z}/\|\mathbf{z}\| \rangle$, so ordering is preserved. □

*Remark.* By transforming a query vector $\mathbf{x}$ into $(\mathbf{x}')_{\beta'}$ and document vectors $\mathbf{y}$ into $(\mathbf{y}'')_{\beta'}$, we can retrieve documents according to the SCM in vector databases that only support nearest neighbor search according to the cosine similarity.

Whereas most vector databases are designed for storing low-dimensional and dense vector coordinates, document vectors have the dimension $n$, which can be in the millions for real-world corpora such as the English Wikipedia. Apart from that, a document contains only a small fraction of the terms in the vocabulary, which makes the coordinates extremely sparse. Therefore, the coordinates need to be converted to a dense low-dimensional representation, using e.g. the latent semantic analysis (LSA), before they are stored in a vector database or used for queries.

Unlike vector databases, inverted-index-based search engines are built around a data structure called the *inverted index*, which maps each term in our vocabulary to a list of documents (a *posting*) containing the term. Documents in a posting are sorted by a common criterion. The search engine tokenizes a text query into terms, retrieves postings for the query terms, and then traverses the postings, computing similarity between the query and the documents.

We can directly replace the search engine's document similarity formula with the formula for the inner product from Lemma 3.3, or the formula for the SCM. After this straightforward change, the system will still only retrieve documents that have at least one term in common with the query. Therefore, we first need to *expand* the query vector $\mathbf{x}$ by computing $((\mathbf{x})_{\beta})^{\mathsf{T}}\mathbf{S}$ and retrieving postings for all terms corresponding to the nonzero coordinates in the expanded vector. The expected number of these terms is $O(mC)$, where $m$ is the number of non-zero elements in $(\mathbf{x})_{\alpha}$, and $C$ is the maximum number of non-zero elements in any column of $\mathbf{S}$. Assuming $m$ and $C$ are bounded by a constant, $O(mC) = O(1)$.

## 5 CONCLUSION AND FUTURE WORK

In this paper, we examined the soft vector space model (VSM) of Sidorov et al. [14]. We restated the definition, we proved a tighter lower time complexity bound of $O(n^3)$ for a related orthonormalization problem, and we showed how the inner product, and the soft cosine measure between document vectors can be efficiently computed in general-purpose vector databases, in the inverted indices of text search engines, and in other applications. To complement this paper, we also provided an implementation of the SCM to Gensim[1] [12], a free open-source natural language processing library.

In our remarks for Theorem 3.4, we discuss strategies for making no column of matrix $\mathbf{S}$ contain more than $C$ non-zero elements. Future research will evaluate their performance on the semantic text similarity task with public datasets. Various choices of the matrix $\mathbf{S}$ based on word embeddings, Levenshtein distance, thesauri, and statistical regression as well as metric matrices from previous work [8] will also be evaluated both amongst themselves and against other document similarity measures such as the LDA, LSA, and WMD.

*Acknowledgements.* We gratefully acknowledge the support by TAČR under the Omega program, project TD03000295. We also sincerely thank three anonymous reviewers for their insightful comments.

## REFERENCES

[1] Andrzej Białecki et al. 2012. Apache Lucene 4. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, 17.

[2] Delphine Charlet and Geraldine Damnati. 2017. SimBow at SemEval-2017 Task 3: Soft-Cosine Semantic Similarity between Questions for Community Question Answering. In *Proc. of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. ACL, Vancouver, Canada, 315–319. DOI: 10.18653/v1/S17-2051.

[3] Elizabeth Cuthill and James McKee. 1969. Reducing the bandwidth of sparse symmetric matrices. In *Proc. of the 1969 24th National Conference* (ACM '69). ACM, 157–172. DOI: 10.1145/800195.805928.

[4] Pinar Heggernes et al. 2001. The Computational Complexity of the Minimum Degree algorithm. Tech. rep. Institute for Computer Applications in Science and Engineering, Hampton VA, (Dec. 2001). https://www.cs.purdue.edu/homes/apothen/Papers/md-conf.pdf.

[5] Roger A. Horn and Charles R. Johnson. 2013. *Matrix Analysis*. (Second ed.). CUP, 662. ISBN: 978-0521548236.

[6] Sergio Jimenez et al. 2012. Soft Cardinality: A Parameterized Similarity Function for Text Comparison. In *Proc. of the 1st Joint Conference on Lexical and Computational Semantics – Volume 1: Proc. of the Main Conference and the Shared Task, and Volume 2: Proc. of the 6th Int. Workshop on Semantic Evaluation* (SemEval '12). ACL. Montreal, Canada, 449–453. http://dl.acm.org/citation.cfm?id=2387636.2387709.

[7] Jeff Johnson et al. 2017. Billion-scale similarity search with GPUs. *ArXiv e-prints*, (Feb. 2017). arXiv: 1702.08734 [cs.CV].

[8] Kenta Mikawa et al. 2011. A proposal of extended cosine measure for distance metric learning in text classification. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*. IEEE, 1741–1746.

[9] Tomáš Mikolov et al. 2013. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, (Jan. 2013). arXiv: 1301.3781 [cs.CL].

[10] Preslav Nakov et al. 2017. SemEval-2017 task 3: community question answering. In *Proc. of the 11th International Workshop on Semantic Evaluation* (SemEval '17). ACL, Vancouver, Canada, (Aug. 2017), 27–48.

[11] Behnam Neyshabur and Nathan Srebro. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In *Proc. of the 32nd Int. Conference on Machine Learning* (ICML'15). Vol. 37. JMLR.org, Lille, France, 1926–1934. http://dl.acm.org/citation.cfm?id=3045118.3045323.

[12] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. English. In *Proc. of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, (May 2010), 45–50. http://is.muni.cz/publication/884893/en.

[13] Gerard Salton and Chris Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Inform. Processing and Management*, 24, 513–523, 5.

[14] Grigori Sidorov et al. 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18, 3, 491–504.

[15] G.W. Stewart. 1998. *Matrix Algorithms: Volume 1: Basic Decompositions. Other Titles in Applied Mathematics*. SIAM, 458. ISBN: 9781611971408.

[16] Mihalis Yannakakis. 1981. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2, 1, 77–79.

[1] See https://github.com/RaRe-Technologies/gensim/, pull requests 1827, and 2016.