Noname manuscript No.
(will be inserted by the editor)

On Measuring Similarity for Sequences of Itemsets

Elias Egho · Chedy Raïssi · Toon Calders · Nicolas Jay · Amedeo Napoli

the date of receipt and acceptance should be inserted later

Abstract Computing the similarity between sequences is a very important challenge for many different data mining tasks. There is a plethora of similarity measures for sequences in the literature, most of them being designed for sequences of items. In this work, we study the problem of measuring the similarity between sequences of itemsets. We focus on the notion of common subsequences as a way to measure similarity between a pair of sequences composed of a list of itemsets. We present new combinatorial results for efficiently counting distinct and common subsequences. These theoretical results are the cornerstone of an effective dynamic programming approach to deal with this problem. In addition, we propose an approximate method to speed up the computation process for long sequences. We have applied our method to various data sets: healthcare trajectories, online handwritten characters and synthetic data. Our results confirm that our measure of similarity produces competitive scores and indicate that our method is relevant for large scale sequential data analysis.

1 Introduction

Sequential data is widely present and used in many applications such as matching of time series in databases Faloutsos et al. [1994], DNA or amino-acids protein sequence analysis Sander and Schneider [1991]; Chothia and Gerstein

Elias Egho · Nicolas Jay · Amedeo Napoli
LORIA, Vandoeuvre-les-Nancy, France,
E-mail: {firstname.lastname}@loria.fr

Chedy Raïssi
INRIA, Nancy Grand Est, France,
E-mail: chedy.raissi@inria.fr

Toon Calders
University Libre Bruxelles,
E-mail: toon.calders@ulb.ac.be

[1997], web log analysis [Yang and Zhang, 2003], and music sequences matching [Serrà et al., 2012]. Consequently, analyzing sequential data has become an important data mining and machine learning task with a special focus on the examination of pairwise relationships between sequences. For example, some clustering and kernel-based learning methods depend on computing distances or similarity scores between sequences [Leslie et al., 2002; Xiong et al., 2011]. However, in many applications, similarity measures on sequential data remain limited to *simple sequences*, which are ordered lists of items (i.e., symbols) [Levenshtein, 1966; Herranz et al., 2011; Keogh, 2002; Wang and Lin, 2007]. By contrast, in modern life sciences [Wodak and Janin, 2002], sequential data sets are often represented as ordered lists of symbol sets (i.e., *itemsets*). This special feature is in itself a challenge and implies to carefully take into account complex combinatorial aspects to compute similarities between sequences.

In this study, we focus on the notion of common subsequences[1] as a means to define a distance or similarity score between a pair of sequences composed of a list of itemsets. The assumption that common subsequences can characterize similarity is not new. For instance, a very well known state-of-the-art algorithm, *longest common subsequence* [Hirschberg, 1975], uses the length of the longest common subsequence as a similarity measure between two sequences. However, as clearly stated by [Wang and Lin, 2007] for simple sequences: *"This measure [...] ignores information contained in the second, third, ..., longest subsequences"* . Additionally, this measure behaves erratically when the sequences contain itemsets. We justify this claim by considering three sequences $U = \langle\{c\}\{b\}\{a,b\}\{a,c\}\rangle$, $V = \langle\{b\}\{c\}\{a,b\}\{a,c\}\rangle$ and $W = \langle\{b,d\}\{a,b\}\{a,c\}\{d\}\rangle$. The longest common subsequence, denoted by LCS, between sequences U and V is $LCS(U,V) = \langle\{b\}\{a,b\}\{a,c\}\rangle$, and between U and W is $LCS(U,W) = \langle\{b\}\{a,b\}\{a,c\}\rangle$. This similarity measure is usually defined as $sim_{LCS}(S,T) = \frac{|LCS(S,T)|}{max(|S|,|T|)}$ and thus one may conclude that because $sim_{LCS}(U,V) = sim_{LCS}\ (U,W) = \frac{3}{4}$, then the sequence U is equidistant from sequence V and W. Clearly this is a wrong result as U is almost the same sequence as V, but with a slight inversion of the two first itemsets. *How can one maximize the information used to compute a similarity measure between two sequences?* Along with [Wang and Lin, 2007], we strongly believe that the *number of common subsequences* (and not only the length of the longest one) between two sequences is appealing in order to answer the previous question. We illustrate this intuition with the three previously considered sequences U, V and W. Let $ACS(S,T)$ be the cardinality of the set that contains *all common subsequences* between S and T. An example of a common subsequence between U and V is $\langle\{c\}\{a,b\}\rangle$. Notice, on the other hand, that $\langle\{c\}\{a,b\}\rangle$ is not a common subsequence between U and W. $ACS(U,V) = 40, ACS(U,W) = 26$ and $ACS(V,W) = 26$. Based on this computation, it is trivial to conclude that sequences U and V share stronger

[1] A subsequence is a sequence that can be derived from another sequence by deleting items without changing the order of itemsets. The notion of subsequence will be further developed in Section 3.

affinity than with W (a result that was not detected by the longest common subsequence measure). To date, there *does not exist* any approach that computes efficiently ACS and use it as a basis for a similarity measure for complex sequences. Accordingly, in this work, the main contributions are summarized as follows:

Theoretical analysis. We start by answering two fundamental theoretical open problems: (i) given a sequence of itemsets, can we count, *without enumerating*, the number of distinct subsequences? (ii) for a pair of sequences, can we *efficiently* count the number of common subsequences? We present two theorems that positively answer these questions.

Algorithmic and approximability results. We discuss and present a dynamic programming algorithm for counting all common subsequences (ACS) between two given sequences. This dynamic programming algorithm allows us to define in a simple and intuitive manner our similarity measure which is the ratio between the number of common subsequences from two sequences S and T divided by the maximal number of distinct subsequences. To cope with large data sets containing long input sequences, we present an *approximation technique* to compute efficiently ACS. The approach relies on approximating the size of a union of a family of sets in terms of the intersections of all subfamilies (i.e., *inclusion-exclusion principle*) based on the direct application of a result from [Linial and Nisan, 1990].

Experiments and Evaluations. The results reported in this work are a useful contribution with direct practical applications to different discriminative approaches, and in particular kernel methods, because new complex sequence kernels can be devised based on the theoretical results provided in this work. Moreover, the method is quite general in that it can be used (with slight modifications) for a broad spectrum of sequence-based classification or clustering problems. We report an extensive empirical study on synthetic datasets and qualitative experiments with datasets consisting of trajectories of cancer patients extracted from French healthcare organizations and online handwritten Assamese characters. We give empirical evidence showing that the proposed approximation method works well in practice. The different versions of the software source codes, data used for experiments and interactive data visualizations are publicly available from `http://www.loria.fr/~eegho/acs/`.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 briefly reviews the preliminaries needed in our development. Section 4 highlights the differences between our similarity measure and the well-known *"longest common subsequence"* along with a discussion on the motivations and the practical impact of our measure. Section 5 and 6 introduces our new combinatorial results. Section 7 presents a complete study of the complexity of the problem and an efficient approximation technique. Experimental studies are reported in Section 8 and we conclude our work in Section 9.

2 Related Work

[Levenshtein, 1966] proposed a measure to compute a distance between strings. Since then, many studies focused on developing efficient approaches for sequence similarities. The Levenshtein distance (or edit distance) between strings s and t is defined as the minimum number of edit operations needed to transform s into t. The edit operations are either an insertion, a deletion, or a substitution of a symbol. Many other approaches are built on this result but with notable differences like weighting the symbols and the edit operations [Herranz et al., 2011], or using stochastic processes [Oncina and Sebban, 2006]. For time series, a well-known approach is the Dynamic Time Warping (DTW) technique for finding an optimal alignment between two series [Berndt and Clifford, 1994]. Intuitively, the sequences are warped in a nonlinear fashion to match each other. DTW had a huge impact and has been used to compare multiple patterns in automatic speech recognition to cope with different speaking speeds [Muzaffar et al., 2005]. [Karlton Sequeira, 2002] and [Vlachos et al., 2003] followed a radically different approach by developing longest common subsequences approaches for the comparison and similarity measure. However, the common information shared between two sequences is more than the longest common subsequence. In fact counting all possible common information units between sequences provides a good idea about the similarity relationship between the sequences and their overall *complexity*. In addition, the common subsequences problem is related to the problem of counting the number of all distinct common subsequences between two sequences. [Wang and Lin, 2007] studied the usage of the count of all common subsequences (ACS) as a similarity measure between two sequences of items. [Elzinga et al., 2008] followed the same intuition and proposed a dynamic programming algorithm to count distinct common subsequences between two sequences of items. In this work, we extend and generalize the previous works of [Wang and Lin, 2007; Elzinga et al., 2008] for complex structures such as sequences of itemsets.

3 Preliminaries

Definition 1 (Sequence) Let $\mathcal{I}$ be a finite set of *items*. An *itemset* X is a non-empty subset of $\mathcal{I}$. A *sequence* S over $\mathcal{I}$ is an ordered list $\langle X_1 \cdots X_n \rangle$, where X_i $(1 \leq i \leq n, n \in \mathbb{N})$ is an itemset. n is called the size of the sequence S, denoted by $|S| = n$. The length, denoted by $\ell(S)$, is the total number of items occurring in the sequence, i.e., $\ell(S) = \sum_{i=1}^{n} |X_i|$. S^l denotes the l-prefix $\langle X_1 \dots X_l \rangle$ of sequence S with $1 \leq l \leq n$. The j-th itemset X_j of sequence S is denoted $S[j]$ with $1 \leq j \leq n$.

Definition 2 (Subsequence) A sequence $T = \langle Y_1 \cdots Y_m \rangle$ is a **subsequence** of $S = \langle X_1 \dots X_n \rangle$, denoted by $T \preceq S$, if there exist indices $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ such that $Y_j \subseteq X_{i_j}$ for all $j = 1 \dots m$ and $m \leq n$. S is said to be a **supersequence** of T. $\varphi(S)$ denotes the **set of all subsequences** of

$\mathcal{D}_{ex} =$

S_1	$\langle\{a\}\{a,b\}\{e\}\{c,d\}\{b,d\}\rangle$
S_2	$\langle\{a\}\{b,c,d\}\{a,d\}\rangle$
S_3	$\langle\{a\}\{b,d\}\{c\}\{a,d\}\rangle$
S_4	$\langle\{a\}\{a,b,d\}\{a,b,c\}\{b,d\}\rangle$

Table 1: The sequence database used as a running example

a given sequence S and $\phi(S) = |\varphi(S)|$. For two sequences S and T, $\varphi(S,T)$ denotes the set of **all common subsequences** between two sequences S and T: $\varphi(S,T) = \varphi(S) \cap \varphi(T)$ and $\phi(S,T) = |\varphi(S,T)|$.

We now define the following similarity measure between two sequences of itemsets S and T.

Definition 3 (Sequence Similarity) The similarity between two sequences S and T, denoted $sim_{ACS}(S,T)$ is defined as the number of common subsequences divided by the maximal number of subsequences of S and T; that is:

$$sim_{ACS}(S,T) = \frac{\phi(S,T)}{\max\{\phi(S),\phi(T)\}}$$

The similarity measure satisfies the following conditions: (i) Non-negativity, $sim_{ACS}(S,T) \geqslant 0$; (ii) Identity of indiscernibles, $sim_{ACS}(S,T) = 1$ if and only if $S = T$ and (iii) Symmetry, $sim_{ACS}(S,T) = sim_{ACS}(T,S)$. However, sim_{ACS} does not respect the *triangular inequality* condition.

Definition 4 (Concatenation) Let $S = \langle X_1 \cdots X_n\rangle$ be a sequence, and Y be an itemset. The **concatenation of the itemset Y with the sequence** S, denoted $S \circ Y$, is the sequence $\langle X_1 \cdots X_n\ Y\rangle$.

As usual, the powerset of an itemset Y will be denoted by $\mathcal{P}(Y)$, and $\mathcal{P}_{\geq 1}(Y)$ denotes all nonempty subsets of Y; that is, $\mathcal{P}_{\geq 1}(Y) = \mathcal{P}(Y) \setminus \{\emptyset\}$.

Example 1: We use the sequence database $\mathcal{D}_{ex}$ in Table 1 as a running example. It contains 4 data sequences over the set of items $\mathcal{I} = \{a,b,c,d,e\}$. Sequence $\langle\{a\}\{b\}\{c,d\}\rangle$ is a subsequence of $S_1 = \langle\{a\}\{a,b\}\{e\}\{c,d\}\{b,d\}\rangle$. $\ell(S_1) = 8$ and $|S_1| = 5$. The 3-prefix of S_1, denoted S_1^3, is $\langle\{a\}\{a,b\}\{e\}\rangle$ and $S_1[2]$, the second itemset in sequence S_1, is $\{a,b\}$. The set of all subsequences of S_4^2 is

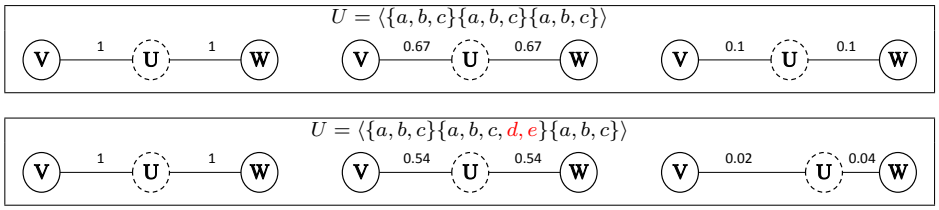
$$\begin{aligned}\varphi(S_4^2) = \{&\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{d\}\rangle, \langle\{a,b\}\rangle, \langle\{a,d\}\rangle, \langle\{b,d\}\rangle, \langle\{a,b,d\}\rangle,\\ &\langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{d\}\rangle, \langle\{a\}\{a,b\}\rangle, \langle\{a\}\{a,d\}\rangle, \langle\{a\}\\ &\{b,d\}\rangle, \langle\{a\}\{a,b,d\}\rangle\}\end{aligned}$$

Hence, $\phi(S_4^2) = 15$. The concatenation of the sequence S_4^2 with the itemset $\{a,b,c\}$, denoted as $S_4^2 \circ \{a,b,c\}$, is the sequence $\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle$. In addition,the set of all common subsequences of S_1^4 and S_2^3 is

$$\varphi(S_1^4, S_2^3) = \{\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{d\}\rangle, \langle\{c\}\rangle, \langle\{c,d\}\rangle, \langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{a\}\{d\}\rangle, \langle\{a\}\{c,d\}\rangle, \langle\{b\}\{d\}\rangle, \langle\{a\}\{b\}\{d\}\rangle\}$$

The similarity between S_1^4 *and* S_2^3 is

$$sim_{ACS}(S_1^4, S_2^3) = \frac{\phi(S_1^4, S_2^3)}{\max\{\phi(S_1^4), \phi(S_2^3)\}} = \frac{13}{\max\{56, 61\}} = \frac{13}{61} = 0.21$$



(a) $sim_{LCS_{size}}$ (b) $sim_{LCS_{length}}$ (c) sim_{ACS}

Fig. 1: Sensitivity of different sequence similarity measures

4 Longest And All Common Subsequences: A Comparison

In this section, we review the well-known similarity measure *longest common subsequence* and our novel similarity based on *all common subsequences*. We briefly compare and contrast the different advantages of the proposed measure.

Using *longest common subsequence* for computing the similarity between two *sequences of itemsets* can lead to *ambiguous* results. There are two ways for measuring the longest common sequence in this case: either by using the *size* or the *length* of the sequence. If the size is used, the similarity between two sequences is defined as: $sim_{LCS_{size}}(S,T) = \frac{|LCS(S,T)|}{max\{|S|,|T|\}}$. If the sequence length is preferred, the similarity between two sequences becomes: $sim_{LCS_{length}}(S,T) = \frac{\ell(LCS(S,T))}{max\{\ell(S),\ell(T)\}}$. For example, consider sequences $S = \langle\{a,b\}\{a,c\}\{a,f\}\rangle$ and $T = \langle\{a,b,f\}\{c\}\rangle$. The longest common subsequence between S and T is $\langle\{a,b\}\{c\}\rangle$, $sim_{LCS_{size}}(S,T) = \frac{2}{3}$ and $sim_{LCS_{length}}(S,T) = \frac{3}{6} = \frac{1}{2}$. One may conclude that the use of the length or the size is decided by the data analyst based on his data set and prior knowledge. However, this duality between the size and length holds deeper consequences as the measure $sim_{LCS_{size}}$ does not satisfy the constraint concerning the identity of indiscernibles (i.e., two sequences may have a similarity of 1 while being different as shown in Figure 1a and thus may hinder the following analysis processes. Our measure avoids the use of an extra parameter of length or size of a sequence. The definition of sim_{ACS} depends only on the number of common subsequences and the total number of subsequences.

Another important point to highlight is the *sensitivity*. Our measure is very sensitive to sequence modifications. Let us consider three sequences $U = \langle\{a,b,c\}\{a,b,c\}\{a,b,c\}\rangle$, $V = \langle\{a,b\}\{a,b\}\{a,b\}\rangle$ and $W = \langle\{a,b\}\{a,b\}\{a,b,d,e\}\rangle$. The similarity between U and V is equal to the similarity between U and W for all the three measures sim_{ACS}, $sim_{LCS_{length}}$ and $sim_{LCS_{size}}$. Thus, U is equidistant to V and W as shown in Figure 1a. When adding items d and e to the second itemset of U and recomputing the similarities, on can see that U remains equidistant to V and W, whether $sim_{LCS_{length}}$ or $sim_{LCS_{size}}$ is used. However, if sim_{ACS} is used, U appears to be more similar to W than to V, as illustrated in Figure 1c.

The rest of the paper, up to the experiments section, will be dedicated to devise efficient techniques for computing $\phi(S)$ and $\phi(S,T)$.

5 Counting All Distinct Subsequences

In this section, we present an efficient technique *to count* the number, $\phi(S)$, of all distinct subsequences for a given sequence S. First, we present the intuition behind the proposed counting scheme. Suppose that we extend a given sequence $S = \langle X_1 \cdots X_n\rangle$ with an itemset Y and we observe the relation between $\phi(S)$ and $\phi(S \circ Y)$. Two cases may appear:

1. Y is disjoint with any itemset in S; i.e., for all $i = 1 \ldots n$, $Y \cap S[i] = \emptyset$, then the number of distinct subsequences of $S \circ Y$ equals $|\varphi(S)| \cdot 2^{|Y|}$, since for all $T \in \phi(S)$ and $Y' \in \mathcal{P}_{\geq 1}(Y)$, $T \circ Y'$ is not in $\phi(S)$. For example, $\phi(\langle\{a,b\}\{c\}\rangle \circ \{d,e\}) = 8 \cdot 2^2 = 32$.
2. At least one item of Y appears in an itemset of S; i.e., $\exists i \in [1,n] : Y \cap S[i] \neq \emptyset$. In this case, $|\varphi(S \circ X)|$ is smaller than $|\varphi(S)| \cdot 2^{|Y|}$, because not every combination of a sequence in $\varphi(S)$ with an element from the power set of Y results in a unique subsequence. For example, if $S = \langle\{a,b\}\rangle$ and $Y = \{a,b\}$, the set of all subsequences of S is $\varphi(S) = \{\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{a,b\}\rangle\}$ and the power set of Y is $\mathcal{P}(Y) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$. The sequence $\langle\{a\}\rangle$ can be obtained by either extending the empty sequence $\langle\rangle \in \varphi(S)$ with the itemset $\{a\} \in \mathcal{P}(Y)$, or by extending $\langle\{a\}\rangle \in \varphi(S)$ with $\emptyset \in \mathcal{P}(Y)$.
   Therefore, we need to define a method to remove the repetitions from the count. Formally, $|\varphi(S \circ Y)| = |\varphi(S)| \cdot 2^{|Y|} - R(S,Y)$ where $R(S,Y)$ represents a *correction term* that equals the number of repetitions of subsequences that should be suppressed for a given S concatenated with the itemset Y.

We illustrate the second case with an example.

Example 2: Consider sequence S_4 from our toy data set. $S_4^2 = \langle\{a\}\{a,b,d\}\rangle$ is the 2-prefix of S_4. Recall from Example 1 that the total number of subsequences of S_4^2 is $\phi(S_4^2) = 15$. Now suppose that we extend this sequence S_4^2 with the itemset $Y = \{a,b,c\}$. Clearly, concatenating each sequence from $\varphi(S_4^2)$ with each element in the power set of $\{a,b,c\}$ will generate some subsequences multiple times. For instance, the subsequence $\langle\{a\}\{b\}\rangle$ is generated

twice: $\langle\{a\}\rangle \circ \{b\}$ and $\langle\{a\}\{b\}\rangle \circ \emptyset$. The same applies to other subsequences $\langle\{a\}\rangle$, $\langle\{b\}\rangle$, $\langle\{a,b\}\rangle$, $\langle\{a\}\{a\}\rangle$ and $\langle\{a\}\{ab\}\rangle$. Thus, making a total of 6 subsequences that are counted twice. In this case, the correct number of distinct subsequences for $S_4^2 \circ Y = \langle\{a\}\{a,b,d\}\{a,b,c\}\rangle$ is $|\varphi(S_4^2)| \cdot 2^{|Y|} - R(S_4^2, Y) = 15 \cdot 2^3 - 6 = 114$.

As illustrated by the above example, the difficulty lies in the computation of the value of the *correction term* $R(S,Y)$. The general idea is to compensate the repeated concatenation of subsequences from S by the power set of Y. The problem occurs with sequences in $\varphi(S) \circ \mathcal{P}_{\geq 1}(Y)$ that are already in $\varphi(S)$. Suppose T is such a sequence, then T must be decomposable as $T' \circ Y'$, where $T' \in \varphi(S^i)$ for some $i = 0 \ldots n-1$, and $Y' \subseteq Y \cap S[j]$, for some $j \in i+1 \ldots n$. The following definition introduces the *position set* that will index the itemset positions that generate duplicates in S.

Definition 5 (Position set) Given a sequence $S = \langle X_1 \cdots X_n\rangle$ and an itemset Y, $L(S,Y)$ is the set of all **maximal positions** where the itemset Y has a maximal intersection with the different itemsets $S[i]$, $i = 1 \ldots n$. Formally,

$$L(S,Y) = \{i \mid S[i] \cap Y \neq \emptyset \wedge \forall j; j > i \wedge S[i] \cap Y \not\subseteq S[j] \cap Y\}$$

Notice that if there are multiple positions that generate the same duplicates, we only consider the last one (right-most in the sequence).

Example 3: Consider $S_4^3 = \langle\{a\}\{a,b,d\}\{a,b,c\}\rangle$ from our running example. $L(\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle, \{b,d\}) = \{2,3\}$. The index 2 is chosen because the intersection between $S_4[2]$ and $\{b,d\}$ is not empty (i.e., $\{a,b,d\} \cap \{b,d\} = \{b,d\} \neq \emptyset$) and is maximal in the sense that $S_4[2] \cap \{b,d\} \not\subseteq S_4[3] \cap \{b,d\}$. Similarly, the index 3 is chosen because $S_4[3] \cap \{b,d\} = \{b\} \neq \emptyset$, and the index of $S_4[3]$ is greater than the index of $S_4[2]$ in the sequence.

The following lemma now formalizes the observation that we only need to consider the sets $S[i]$ for i in the position set.

Lemma 1 *Let S be a sequence and Y an itemset. Then $\phi(S \circ Y) = \phi(S) \cdot 2^{|Y|} - R(S,Y)$, with*

$$R(S,Y) \;=\; \left| \bigcup_{\ell \in L} \{\varphi(S^{\ell-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap Y)\} \right|$$

Proof See Appendix.

Notice, however, that the sets $\varphi(S^{\ell-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap Y)$ are not necessarily disjoint; consider, e.g., $S = \langle\{a,b\}\{b,c\}\rangle$ and $Y = \{a,b,c\}$. The position set is $L = \{1,2\}$, and $\langle\{b\}\rangle$ appear in both $\varphi(S^0) \circ \mathcal{P}_{\geq 1}(S[1] \cap Y)$ and $\varphi(S^1) \circ \mathcal{P}_{\geq 1}(S[2] \cap Y)$. To incorporate this overlap, we compute the cardinality of the union in Lemma 1 using the inclusion-exclusion principle, leading to the following theorem:

Theorem 1 *Let* $S = \langle X_1 \dots X_n \rangle$ *and* Y *be an itemset. Then,*

$$\phi(S \circ Y) = \phi(S) \cdot 2^{|Y|} - R(S, Y) \tag{1}$$

with

$$R(S,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\phi(S^{\min(K)-1}) \cdot \left(2^{|(\bigcap_{j \in K} S[j]) \cap Y|} - 1 \right) \right) \tag{2}$$

Proof See Appendix.

Example 4: We illustrate the complete counting process with sequence S_4^3. The position sets for this sequence are: $L(\langle\rangle, \{a\}) = \emptyset,\ L(\langle\{a\}\rangle, \{a,b,d\}) = \{1\},\ L(\langle\{a\}\{a,b,d\}\rangle, \{a,b,c\}) = \{2\}$.

$$\begin{aligned}
\phi(\langle\rangle) &= 1 \\
\phi(\langle\{a\}\rangle) &= \phi(\langle\rangle) \cdot 2^{|\{a\}|} \ = \ 2 \\
\phi(\langle\{a\}\{a,b,d\}\rangle) &= \phi(\langle\{a\}\rangle) \cdot 2^{|\{a,b,d\}|} - \phi(\langle\rangle) \cdot (2^{|\{a,b,d\} \cap \{a\}|} - 1) \\
&= 2 \cdot 2^3 - 1 \cdot (2^1 - 1) = 15 \\
\phi(\langle\{a\}\{a,b,d\}\{a,b,c\}\rangle) & \\
&= \phi(\langle\{a\}\{a,b,d\}\rangle) \cdot 2^{|\{a,b,c\}|} - \phi(\langle\{a\}\rangle) \cdot (2^{|\{a,b,d\} \cap \{a,b,c\}|} - 1) \\
&= 15 \cdot 2^3 - 2 \cdot (2^2 - 1) = 114
\end{aligned}$$

6 Counting All Common Subsequences

In this section, we will extend the previous results to count all common distinct subsequences between two sequences S and T. Again, we discuss the basic intuition and then present the main result. Suppose that we extend the sequence S with an itemset Y and we observe the relation between $\varphi(S,T)$ and $\varphi(S \circ Y, T)$, two cases may appear:

1. If no items in Y appear in any itemset of S and T then the concatenation of the itemset Y with the sequence S *has no effect* on the the set $\varphi(S,T)$.
2. If at least one item in Y appears in either one of the sequences S or T (or both) then it can be observed that new common subsequences may appear in $\varphi(S,T)$. As for the counting method of the distinct subsequences of a unique sequence S, repetitions may occur and a generalized correction term for both S and T needs to be defined. Formally,

$$|\varphi(S \circ Y, T)| = |\varphi(S,T)| + A(S,T,Y) - R(S,T,Y)$$

where $A(S,T,Y)$ represents the number of extra common subsequences that should be added and $R(S,T,Y)$ is the correction term.

Similarly to the distinct subsequence problem, the position set will index the positions that generate duplicate sequences. The following lemma formalizes this observation:

Lemma 2 *Let $S = \langle X_1 \dots X_n \rangle$, $T = \langle X'_1 \dots X'_m \rangle$ and Y an itemset.*

$$A(S,T,Y) \;=\; \left| \bigcup_{\ell \in L(T,Y)} \left\{ \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geq 1}(T[\ell] \cap Y) \right\} \right|$$

$$R(S,T,Y) = \left| \bigcup_{\ell \in L(S,Y)} \left\{ \bigcup_{\ell' \in L(T,Y)} \left\{ \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right\} \right|$$

Proof See Appendix.

Example 5: Let $S_1^4 = \langle \{a\}\{a,b\}\{e\}\{c,d\} \rangle$ be the 4-prefix of S_1, and let $S_2^3 = \langle \{a\}\{b,c,d\}\{a,d\} \rangle$ be the 3-prefix of S_2 from our running example. Suppose that we extend S_1^4 with the itemset $Y = \{b,d\}$ and count all distinct common subsequences between $S_1^4 \circ \{b,d\}$ and S_2^3. Notice that the intersections between itemset $\{b,d\}$ and the itemsets in S_2^3 are non-empty and maximal for the itemsets $\{b,c,d\}$ and $\{a,d\}$. Thus, $L(S_2^3, \{b,d\}) = \{2,3\}$ and $A(S_1^4, S_2^3, \{b,d\}) = |\{\varphi(S_1^4, S_2^1) \circ \mathcal{P}_{\geq 1}(\{b,d\} \cap \{b,c,d\})\} \cup \{\varphi(S_1^4, S_2^2) \circ \mathcal{P}_{\geq 1}(\{b,d\} \cap \{a,d\})\}| = 14$. In a similar way, the intersections between itemset $\{b,d\}$ and the itemsets in S_1^4 are non-empty and maximal for the itemsets in positions 2 and 4.Thus $L(S_1^4, \{b,d\}) = \{2,4\}$. In this case, adding the values $A(S_1^4, S_2^3, \{b,d\})$ to $\phi(S_1^4, S_2^3)$ will count erroneously some subsequences. For instance, the subsequences $\langle \{a\}\{b\}\{d\} \rangle$ and $\langle \{b\}\{d\} \rangle$ are counted twice: once in $\varphi(S_1^4, S_2^3)$ and the other when all sequences of the set $\varphi(S_1^4, S_2^3)$ are extended with $\{b,d\} \cap \{a,d\}$. The same remark applies to other subsequences: $\langle \{b\} \rangle$,$\langle \{d\} \rangle$, $\langle \{a\}\{b\} \rangle$ and $\langle \{a\}\{d\} \rangle$. In this case, the correct number of all common distinct subsequences between $S_1^4 \circ \{b,d\}$ and S_2^3 is $|\varphi(S_1^4, S_2^2)| + A(S_1^4, S_2^3, \{b,d\}) - R(S_1^4, S_2^3, \{b,d\})$ where:

$$\begin{aligned} R(S_1^4, S_2^3, \{b,d\}) = |&\{\varphi(S_1^1, S_2^1) \circ \mathcal{P}_{\geq 1}(\{a,b\} \cap \{b,c,d\} \cap \{b,d\})\} \\ &\cup \{\varphi(S_1^1, S_2^2) \circ \mathcal{P}_{\geq 1}(\{a,b\} \cap \{a,d\} \cap \{b,d\})\} \\ &\cup \{\varphi(S_1^3, S_2^1) \circ \mathcal{P}_{\geq 1}(\{c,d\} \cap \{b,c,d\} \cap \{b,d\})\} \\ &\cup \{\varphi(S_1^3, S_2^2) \circ \mathcal{P}_{\geq 1}(\{c,d\} \cap \{a,d\} \cap \{b,d\})\}| = 6 \end{aligned}$$

Thus,

$$\begin{aligned} \phi(S_1^4 \circ \{b,d\}, S_2^3) &= |\varphi(S_1^4, S_2^2)| + A(S_1^4, S_2^3, \{b,d\}) - R(S_1^4, S_2^3, \{b,d\}) \\ &= 13 + 14 - 6 = 21 \end{aligned}$$

Similarly to Lemma 1 and as illustrated in the above example, the computation of the cardinality of the unions in Lemma 2 implies the usage of the inclusion-exclusion principle. This remark leads to the second theorem:

Theorem 2 *Let $S = \langle X_1 \dots X_n \rangle$, $T = \langle X'_1 \dots X'_m \rangle$ and Y an itemset. Then,*

$$\phi(S \circ Y, T) = \phi(S,T) + A(S,T,Y) - R(S,T,Y) \tag{3}$$

with

$$A(S,T,Y) = \sum_{K \subseteq L(T,Y)} (-1)^{|K|+1} \left(\phi(S, T^{\min(K)-1}) \cdot \left(2^{|(\bigcap_{j \in K} T[j]) \cap Y|} - 1 \right) \right) \tag{4}$$

and

$$R(S,T,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\sum_{K' \subseteq L(T,Y)} (-1)^{|K'|+1} \cdot f(K,K') \right) \tag{5}$$

where

$$f(K,K') = \phi(S^{\min(K)-1}, T^{\min(K')-1}) \cdot \left(2^{|(\bigcap_{j \in K} S[j]) \cap \left(\bigcap_{j' \in K'} T[j']\right) \cap Y|} - 1 \right)$$

Proof See Appendix.

6.1 Dynamic Programming

Theorem 2 leads to a simple dynamic programming algorithm. For two given sequences S and T, such that $|S| = n$ and $|T| = m$, the program produces a $(n+1) \times (m+1)$ matrix (with indices starting at 0), denoted $\mathcal{M}$ such that:

$$\mathcal{M}_{i,j} = \begin{cases} 1 & i = 0 \\ 1 & j = 0 \\ \phi(S^i, T^j) & \text{otherwise.} \end{cases}$$

Example 6: Consider the two sequences S_1 and S_2 from our running example. $\phi(S_1, S_2) = 21$ and the set of all common subsequences of S_1 and S_2 is:

$$\begin{aligned} \varphi(S_1,S_2) = \{ &\langle\rangle, \langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle, \langle\{d\}\rangle, \langle\{c,d\}\rangle, \langle\{b,d\}\rangle, \langle\{a\}\{a\}\rangle, \\ &\langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{a\}\{d\}\rangle, \langle\{a\}\{b,d\}\rangle, \langle\{a\}\{c,d\}\rangle, \\ &\langle\{b\}\{d\}\rangle, \langle\{c\}\{d\}\rangle, \langle\{d\}\{d\}\rangle, \langle\{c,d\}\{d\}\rangle, \langle\{a\}\{d\}\{d\}\rangle, \\ &\langle\{a\}\{b\}\{d\}\rangle, \langle\{a\}\{c\}\{d\}\rangle, \langle\{a\}\{cd\}\{d\}\rangle\} \end{aligned}$$

We detail the computation of the cell $\mathcal{M}_{2,3}$ which represents the number of all common subsequence between $S_1^1 \circ \{a,b\}$ and S_2^3. The position sets are $L(S_1^1, \{a,b\}) = \{1\}$ and $L(S_2^3, \{a,b\}) = \{2,3\}$. Using the result in Theorem 2:

$$\begin{aligned} \mathcal{M}_{2,3} &= \phi(\langle\{a\}\{a,b\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle) \\ &= \phi(\langle\{a\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle) + A(S_1^1, S_2^3, \{a,b\}) - R(S_1^1, S_2^3, \{a,b\}) \end{aligned}$$

$\phi(\langle\{a\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle)$ was previously computed and stored in $\mathcal{M}_{1,3} = 2$. $A(S_1^1, S_2^3, \{a,b\})$ and $R(S_1^1, S_2^3, \{a,b\})$ are computed as follows:

$$\begin{aligned} A(S_1^1, S_2^3, \{a,b\}) &= \phi(S_1^1, S_2^1) \cdot (2^{|\{b,c,d\}\cap\{a,b\}|} - 1) + \phi(S_1^1, S_2^2) \cdot (2^{|\{a,d\}\cap\{a,b\}|} - 1) \\ &\quad -\phi(S_1^1, S_2^1) \cdot (2^{|\{b,c,d\}\cap\{a,d\}\cap\{a,b\}|} - 1) \\ &= \mathcal{M}_{1,1} \cdot (2-1) + \mathcal{M}_{1,2} \cdot (2-1) - \mathcal{M}_{1,1} \cdot (1-1) \\ &= 2 + 2 - 0 = 4 \end{aligned}$$

$$\begin{aligned} R(S_1^1, S_2^3, \{a,b\}) &= \phi(S_1^0, S_2^1) \cdot (2^{|\{a\}\cap\{b,c,d\}\cap\{a,b\}|} - 1) \\ &\quad +\phi(S_1^0, S_2^2) \cdot (2^{|\{a\}\cap\{a,d\}\cap\{a,b\}|} - 1) \\ &\quad -\phi(S_1^0, S_2^1) \cdot (2^{|\{a\}\cap\{b,c,d\}\cap\{a,d\}\cap\{a,b\}|} - 1) \\ &= \mathcal{M}_{0,1} \cdot (1-1) + \mathcal{M}_{0,2} \cdot (2-1) - \mathcal{M}_{0,1} \cdot (1-1) = 1 \end{aligned}$$

Finally, $\mathcal{M}_{2,3} = \phi(\langle\{a\}\{a,b\}\rangle, \langle\{a\}\{b,c,d\}\{a,d\}\rangle) = 2 + 4 - 1 = 5$.
The entire computation for $\phi(S_1, S_2)$ is illustrated in Table 6.

	$\{\emptyset\}$	$\{a\}$	$\{b,c,d\}$	$\{a,d\}$
$\{\emptyset\}$	1	1	1	1
$\{a\}$	1	2	2	2
$\{a,b\}$	1	2	4	5
$\{e\}$	1	2	4	5
$\{c,d\}$	1	2	10	13
$\{b,d\}$	1	2	12	21

Table 2: Matrix for counting all common subsequences between S_1 and S_2

7 Complexity and Linial-Nisan Approximation Results

7.1 Complexity

We will now discuss the complexity of computing the number of subsequences in a sequence of itemets and the number of common subsequences in two such sequences using the formulas in Theorems 1 and 2. Essential in this analysis is the size of the position set $L(S,Y)$, which will highly depend on the specific cases. It is important to notice that the size of $L(S,Y)$ is bounded by both $2^{|Y|}$ (every index corresponds to a unique subset of Y) and $|S|$ (every index corresponds to a unique position within S). Notice incidentally that the worst case $|L(S^{\ell-1}, S[\ell])| = \ell - 1$ is unlikely to happen for long sequences, as this implies

that if we construct the following sequence: $S[1] \cap S[\ell], \ldots, S[\ell-1] \cap S[\ell]$, none of the entries in the sequence is followed by a superset. This would only happen in pathological cases such as $\langle\{a,b,c\}\{a,b\}\{a,c\}\{b,c\}\{a\}\{b\}\{c\}\{a,b,c\}\rangle$. In this case $L(S^{\ell-1}, S[\ell]) = \{1,2,3,4,5,6,7,8\}$.

First we analyze the complexity of the brute-force method consisting of generating all subsequences followed by elimination of duplicates. For a sequence $\langle S_1 ... S_\ell \rangle$, there are at most $N = \Pi_{i=1}^{\ell} 2^{|S[i]|} = 2^{\ell(S)}$ subsequences we need to consider; for every position $i = 1 \ldots \ell$, any subset of S_i needs to be considered in combination with every subset of the other positions. Eliminating the duplicates can be done in time $N \log(N)$. The total complexity is hence $\mathcal{O}(N \log(N))$. Hence the total complexity of the brute-force method for finding all subsequences of S is bounded by $\ell(S)2^{\ell(S)}$. When computing the number of subsequences that two sequences S and T have in common, first their respective subsequences are listed, sorted and deduplicated. Then the two ordered lists can be intersected in time linear in the length of the longest list. Let M be the upper bound on the number of subsequences of T. The complexity of these operations comes down to $\mathcal{O}(N \log(N) + M \log(M)) = \mathcal{O}(\ell(S)2^{\ell(S)} + \ell(T)2^{\ell(T)})$.

Let us now analyze the complexity of the computation based upon the formula in Theorem 2. Suppose that we need to compute the number of subsequences of $\langle S_1 ... S_\ell \rangle$. Assume that we know the number of unique subsequences for all $S^k, k < \ell$. The number of computations we need to perform if we apply the formula of Theorem 1, to get the number of subsequences of $\langle S_1 ... S_k \rangle$ is proportional to the size of the powerset of $L(S^{\ell-1}, S[\ell])$; indeed, we need to compute a sum over all subsets of $L(S^{\ell-1}, S[\ell])$. It is easy to see that the size of this set $L(S^{\ell-1}, S[\ell])$ is bounded by $\min(\ell - 1, 2^{|S[\ell]|})$: every position can appear at most once, and for every subset in $S[\ell]$ there can be at most one position. The total complexity is therefore bounded by $\sum_{k=1}^{\ell} 2^{\min(k-1, 2^{|S[k]|})}$ and $\sum_{k=1}^{\ell} 2^{k-1} = 2^\ell - 1 = \mathcal{O}(2^\ell)$ is an upper bound, which is significantly better than the brute-force method listing all subsequences and removing the duplicates; even though $\mathcal{O}(2^\ell)$ is a very conservative upper bound (position sets will usually only contain a fraction of all possible indices!), the difference between 2^ℓ and $2^{\ell(S)}$ is enormous if the individual itemsets are large. For the pathological case described in the beginning of this section, for instance, the difference is as big as $2^{\ell(S)} = 2^{32}$ versus $2^\ell = 2^{15}$; that is: the number of steps by the method based on Theorem 1 is 131 072 times smaller than in the brute force algorithm.

The complexity for the computation of the number of common subsequences in Theorem 2 goes along the same lines. Again we will first assume that for two sequences S and T, the number of common subsequences $\phi(S^i, T^j)$ has been computed for all S^i and T^j with (i,j) smaller than $(|S|, |T|)$. Let Y be the last itemset of S; that is, $Y = S[|S|]$, and $S' = S^{|S|-1}$. The main complexity term in the formula in Theorem 2 is in $R(S', T, S[|S|])$; this term dominates the complete expression in terms of computational complexity. The complexity of the double sum is proportional to $2^{|L(S',Y)|}2^{|L(T,Y)|}$, which is bounded

by $2^{\min(|S'|,2^{|Y|})}2^{\min(|T|,2^{|Y|})} < 2^{|S'||T|}$. So, the total complexity, taking into account that we need to compute the number of common subsequences for all subsequences of S and T as well (cfr. the dynamic programming approach given in 6.1), leads to a total complexity of $\sum_{i=1}^{|S|}\sum_{j=1}^{|T|} 2^{ij} = \mathcal{O}(\min(|S|,|T|)\cdot 2^{|S||T|})$.

In conclusion, both the brute-force and the dynamic approach are, in worst case, exponential in the input. The brute-force method, however, is exponential in $\ell(S)$ and $\ell(T)$, whereas the dynamic approach is only exponential in the position sets L, which *can be* (but only in pathological cases *are*) as large as $|S| << |\ell(S)|$ and $|T| << \ell(T)$. So, even in the very unlikely worst case the dynamic approach is much better than the brute-force method. The brute-force algorithm, in contrast, takes exponential time in all cases. Although worst time polynomial complexity for similarity measures are highly desirable, we point out that in the experimental section it is clearly shown that the average runtime of the similarity computation is within acceptable ranges for the described applications. Furthermore, it is shown that in many applications the distance measure outperforms other, more efficiently computable measures. This is not an uncommon situation; another example of a distance measure that is hard to compute but very useful is for instance the graph edit distance [Gao et al., 2010].

7.2 Linial-Nisan Approximation

As stated by [Linial and Nisan, 1990]: *"Many computational problems may be viewed as asking for the size of a union of a collection of sets. [...] In these cases, the inclusion-exclusion formula may be used to find the size of the union"*. Our similarity measure relies heavily on the inclusion-exclusion principle: on the one hand, the exact computation of the number of all distinct subsequences of a sequence requires the computation of the *correction number*, $R(S,Y)$ in Equation 2, on the other hand the number of common subsequences needs the computation of the addition and correction terms, $A(S,T,Y)$ and $R(S,T,Y)$ in Equations 4 and 5. The computation drawback is the fact that the inclusion-exclusion formula has an exponential number of terms which, as mentioned previously in the complexity subsection, can become a problem with very long sequences and a position set L of large cardinality. This prompted our interest in approximating our similarity measure through the approximation of the inclusion-exclusion formula used in both ACS and ADS computations.

Theorem 3 (Linial-Nisan Approximation) *[Linial and Nisan, 1990, Theorem 2]. Let $A_1, A_2, \ldots, A_N$ be a collection of sets. Suppose that $|\bigcap_{i\in S} A_i|$ is given for every subset $S \subset \{1,\cdots,N\}$ of cardinality $|S| < K$. For any integers K, N there exist (explicitly given) constants $(\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N})$ such that for every collection of sets $A_1, A_2, \ldots, A_N$, the quantity*

$$\sum_{|S|\leq K} \alpha_{|S|}^{K,N} |\bigcap_{i\in S} A_i|$$

differs from $|\bigcup_{i=1}^{N} A_i|$ *by at most a factor of* $1 + O(e^{-\frac{2K}{\sqrt{N}}})$ *if* $K \geq \Omega(\sqrt{N})$ *or* $O(\frac{N}{K^2})$ *if* $K \leq O(\sqrt{N})$.

The real numbers $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N}$ are defined by Linial and Nisan to be the coefficients of the linearly transformed Chebyshev polynomials expressed in terms of the polynomials $\binom{x}{1}, \binom{x}{2}, \ldots, \binom{x}{K}$. $\overrightarrow{\alpha} = (\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N})$ is calculated efficiently by solving a set of linear equations. Consider the above polynomial identity for $x = 1, \ldots, K$. The vector of coefficients is calculated as follows: $\overrightarrow{\alpha} = \overrightarrow{t} \cdot \mathcal{A}^{-1}$, where $\mathcal{A}$ is the matrix whose (i,j) entry is $\binom{j}{i}$. The inverse matrix $\mathcal{A}^{-1}(i,j)$ is defined as $(-1)^{i+j}\binom{j}{i}$, $\overrightarrow{t} = (q_{K,N}(1), q_{K,N}(2), \ldots, q_{K,N}(K))$ is the linearly transformed Chebyshev polynomials, $q_{K,N}(x) = 1 - \frac{T_k(\frac{2x-(N+1)}{N-1})}{T_k(\frac{-(N+1)}{N-1})}$ and $T_K(x)$ is a polynomial of degree K and is given by

$$T_K(x) = \frac{(x + \sqrt{x^2-1})^K + (x - \sqrt{x^2-1})^K}{2}$$

In our approximation method, every time that the position set is too big (i.e., $|L| \geq \sigma$ where σ is a user provided size threshold) we compute $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_k^{K,N}$ with $K = \lceil \sqrt{|L|} \rceil$ and $N = |L|$ and then approximate the inclusion-exclusion formula through the following Theorems:

Theorem 4 *Let* $S = \langle X_1 \ldots X_n \rangle$ *and* Y *an itemset. Then,*

$$\phi_{LN}(S \circ Y) = 2^{|Y|} \cdot \phi_{LN}(S) - R_{LN}(S, Y) \tag{6}$$

with

$$R_{LN}(S,Y) = \sum_{k=1}^{K} \alpha_k^{K,N} \sum_{\substack{O \subseteq L(S,Y) \\ |O|=k}} \phi_{LN}(S^{\min(O)-1}) \cdot \left(2^{|(\bigcap_{j \in O} S[j]) \cap Y|} - 1\right)$$

where $N = |L(S,Y)|$, $K = \lceil \sqrt{|N|} \rceil$ *and* $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N}$ *are the Linial-Nisan coefficients.*

Theorem 5 *Let* $S = \langle X_1 \ldots X_n \rangle$, $T = \langle X'_1 \ldots X'_m \rangle$, Y *an itemset. Then,*

$$\phi_{LN}(S \circ Y, T) = \phi_{LN}(S,T) + A_{LN}(S,T,Y) - R_{LN}(S,T,Y) \tag{7}$$

with

$$A_{LN}(S,T,Y) = \sum_{k'=1}^{K'} \alpha_k^{'K',N'} \sum_{\substack{O' \subseteq L(T,Y) \\ |O'|=k'}} \phi_{LN}(S, T^{\min(O')-1})(2^{|(\bigcap_{j \in O'} T[j]) \cap Y|} - 1)$$

$$R(S,T,Y) = \sum_{k=1}^{K} \alpha_k^{K,N} \sum_{\substack{O \subseteq L(S,Y) \\ |O|=k}} \left(\sum_{k'=1}^{K'} \alpha_k^{'K',N'} \sum_{\substack{O' \subseteq L(S,Y) \\ |O'|=k'}} f(O,O') \right)$$

where
$f(O,O') = \phi_{LN}(S^{\min(O)-1}, T^{\min(O')-1}) \cdot \left(2^{|(\bigcap_{j\in O} S[j])\cap\left(\bigcap_{j'\in O'} T[j']\right)\cap Y|} - 1\right)$,
$N = |L(S,Y)|$, $N' = |L(T,Y)|$, $K = \lceil\sqrt{|L(S,Y)|}\rceil$, $K' = \lceil\sqrt{|L(T,Y)|}\rceil$ *and* $\alpha_1^{K,N}, \alpha_2^{K,N}, \ldots, \alpha_K^{K,N}$, $\alpha_1^{'K',N'}, \alpha_2^{'K',N'}, \ldots, \alpha_{K'}^{'K',N'}$ *are the Linial-Nisan coefficients.*

Example 7: Consider $S = \langle\{a,c,d,e,f,g,h,i,j,k\}\{a,b,d,e,f,g,h,i,j,k\}\{a,b,c,e,f,g,h,i,j,k\}\{a,b,c,d,f,g,h,i,j,k\}\{a,b,c,d,e,g,h,i,j,k\}\{a,b,c,d,e,f,h,i,j,k\}\{a,b,c,d,e,f,g,i,j,k\}\{a,b,c,d,e,f,g,h,j,k\}\{a,b,c,d,e,f,g,h,i,k\}\{a,b,c,d,e,f,g,h,i,j,k\}\rangle$. The number of distinct subsequences for S^9 is $\phi(S^9) = 1{,}233\,117\,9 \cdot 10^{27}$ and the position set for the last itemset is $L(S^9, S[10]) = \{1,2,3,4,5,6,7,8,9\}$. With the normal exact computation the final number of distinct subsequences is: $\phi(S^{10}) = 2^{|\{a,b,c,d,e,f,g,h,i,j,,k\}|}.\phi(S^9) - R(S^9, S[10]) = 2{,}524\,192\,3 \cdot 10^{30}$. Notice that the inclusion-exclusion formula used for the computation of $R(S^9, S[10])$ contains $\sum_{i=1}^{9}\binom{9}{i}$ terms. To do the Linial-Nisan approximation, remark that $N = |L| = 9$ and $K = \lceil\sqrt{N}\rceil = 3$. After solving the set of linear equations (detailed in the Appendix), $\alpha_1^{3,9} = 0.75$; $\alpha_2^{3,9} = -0.36$; $\alpha_3^{3,9} = 0.1$. Thus, the approximated number of distinct subsequences for S is $\phi_{LN}(S^{10}) = 2^{|\{a,b,c,d,e,f,g,h,i,j,,k\}|}.\phi_{LN}(S^9) - R_{LN}(S^9, S[10]) = 2{,}524\,495\,6 \cdot 10^{30}$ which is very close to the above number.

8 Experiments

In this section we empirically evaluate our similarity measure on synthetic and real-world datasets. Our approach is implemented in both C++ and Java languages. The goal of these experiments is to show the usefulness of the proposed similarity measure. The whole analysis is run over a MacBook Pro with a 2.5GHz Intel Core i5, 4GB of RAM running OS X 10.6.8. A dedicated web page to visualize data sets and interact with the experimental results is available at `http://www.loria.fr/~eegho/acs/`.

8.1 Studying Scalability on Synthetic Data Sets

In the following, we study the scalability of our measure computation. We assess the different runtimes with respect to three different parameters: (*i*) the average number of itemsets in a sequence; (*ii*) the average number of items in each itemset of a sequence and (*iii*) the total number of sequences that are processed through the similarity computation.We generated our datasets by IBM Quest synthetic data generator. Table 3 describes different synthetic datasets used in our experiments.

Figures 2 and 3 show the evolution of the runtime of 499 500 ($\frac{n\times(n-1)}{2}$) comparisons over 1 000 sequences w.r.t the average number of items in each

Dataset	Itemset Length	Sequences Length	Nb-Sequence
$data_1$	5	5	1000
$data_2$		10	
$data_3$		15	
$data_4$		20	
$data_5$		25	
$data_6$	10	5	1000
$data_7$		10	
$data_8$		15	
$data_9$		20	
$data_{10}$		25	
⋮			
$data_{26}$	30	5	1000
$data_{27}$		10	
$data_{28}$		15	
$data_{29}$		20	
$data_{30}$		25	
$data_{31}$	10	10	2000
$data_{32}$			3000
$data_{33}$			4000
$data_{34}$			5000

Table 3: Synthetic Datasets

itemset and the average number of itemsets in each sequence. We run this test on several types of sequences: sequences with itemsets of cardinality 5, 10, 15, 20, 25, 30 (i.e, the number of items in each itemset) and with several lengths: 5, 10, 15, 20 and 25 itemsets (i.e, the number of itemsets in each sequence). As expected, the plots on Figure 2 show that the execution time for calculating the similarity matrix without any approximation increases with the size of the sequences. In all the experiments, applying Linial-Nisan approximation depends on a parameter k. When the length of the position set is greater than k, then the approximation method is applied. With the Linial-Nissan approximation, the execution time is greatly reduced as seen on Figure 3. The time to compute the similarities with the approximation for 1000 sequences of 25 itemsets is less than 1000 seconds in comparison with more than 150000 seconds (i.e., there is a 150× runtime gain). The next experiment aims at comparing the runtime performances of sim_{LCS} with our proposed similarity measure. Figure 4 presents a comparison of the similarity matrix computation runtimes based on sim_{ACS}, sim_{LCS} and sim_{ACS} with the Linial-Nisan approximation with $k = 5$. In this experiment, the length of the sequence is fixed to 15 (i.e., 15 items per sequence). The plot illustrates the fact that our similarity measure, along with the Linial-Nissan approximation, takes almost the same runtime as the longest common subsequence computations. The plot on Figure 5 highlights the impact of the parameter k on the runtime for the Linial-Nissan approximation.

The final experiment focuses on the similarity matrix computation runtime when the size of the input sequences increases. We run these tests over sequences with 10 itemsets on average and with 10 items for each itemset. For

5 000 sequences (i.e. 12 497 500 similarity comparisons), the execution time for our similarity measure is about 30 minutes (1 832 seconds) and about 11 minutes (660 seconds) for the Linial-Nissan approximation. Figure 6 reports the different runtime observations. These experiments highlight the fact that our measure is efficient in terms of runtime for a large panel of sequences with different varying parameters.

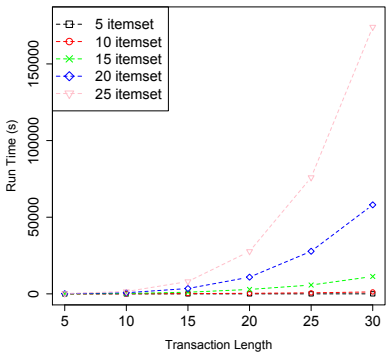


Fig. 2: 1 000 sequences similarity matrix computation runtime. Various lengths of itemsets

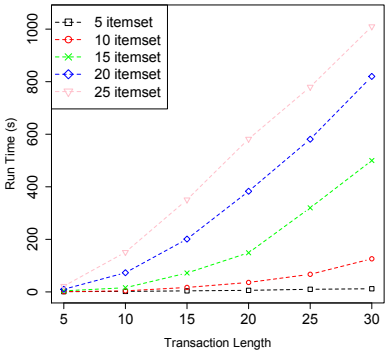


Fig. 3: 1 000 sequences similarity matrix computation with approximation runtime. Various lengths of itemsets

8.2 Analyzing Healthcare Trajectories

This batch of experiments was conducted with healthcare data from PMSI[2], a French nationwide hospital information system. In this system, each hospital stay leads to the collection of a standardized set of administrative and medical data. Although they are essentially used for payment purposes, data from the PMSI can also support the exploration of patients' journeys through several hospitalizations and feed a decision support system, helping healthcare managers for strategic planning and organization of the healthcare system. Such a goal cannot be reached without a recomposition and a better understanding of the so called healthcare trajectories.

In a healthcare trajectory, every hospitalization can be described by the healthcare institution where it takes place, its main cause (diagnosis) and a set of medical and surgical procedures underwent by the patient. For example $\{Moselle, Metz\ regional\ hospital,\ lung\ cancer,\ chest\ radiography\}$ represents a stay in the regional hospital of the city of Metz, in the administative

[2] Programme de Médicalisation des Systèmes d'Information

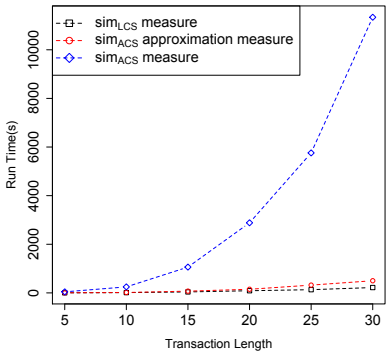


Fig. 4: Multiple 1 000 sequences similarity matrix computations. Sequences of fixed length: 15 items

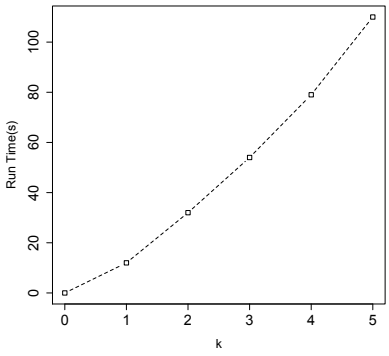


Fig. 5: 1 000 sequences similarity matrix computation with approximation runtime. Various values of k

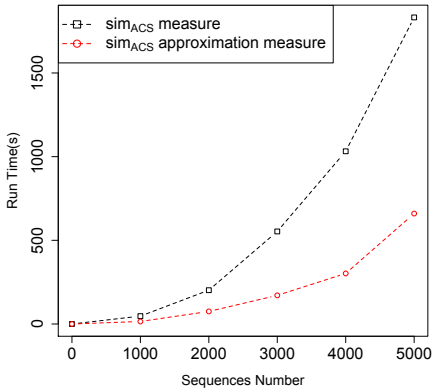


Fig. 6: Similarity matrix calculation run time w.r.t the number of sequences

area of Moselle[3], for a lung cancer, where the patient underwent a chest radiography. A patient trajectory is modeled as a sequence of itemsets, each itemset representing one hospitalization. Computing similarity between patient care trajectories will open the way to patient clustering.

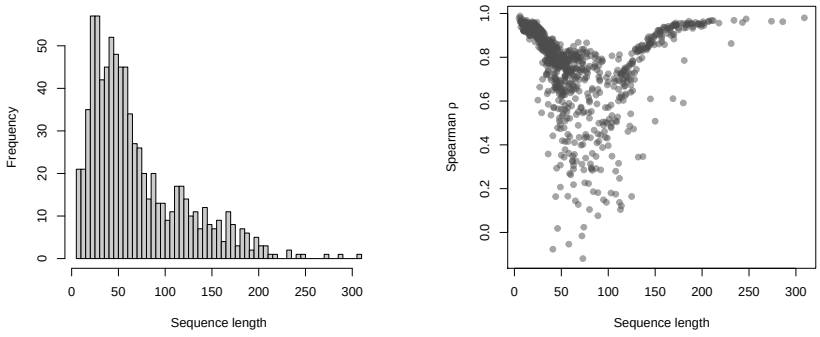
Our dataset contains 828 patients suffering from lung cancer who live in the Lorraine region, of Eastern France. In PMSI, information is coded using controlled vocabularies. In particular, diagnoses are coded with the International

[3] Moselle is one of the 101 departments of France

Patients	*Trajectories*
P_1	$\langle\{54, CHU_{nancy}, C34, ZBQK\}\{57, CL_{metz}, Z51, ZBQK\}\rangle$
P_2	$\langle\{75, CH_{paris}, C34, ZBQK\}\{57, CL_{metz}, Z51, GFFA, GLLD\}\rangle$

Table 4: Healthcare trajectories of 2 patients

Classification of Diseases (ICD10)[4]and medical procedures with the French nomenclature for procedures (CCAM)[5]. Table 4 shows an example of healthcare trajectories for two patients. For example, P_1 has two hospitalizations. He was admitted in the University Hospital of Nancy (encoded as CHU_{nancy}), in Meurthe-et-Moselle (department number 54) for a lung cancer ($C34$), and underwent a chest radiography ($ZBQK$). Then, he was hospitalized in a private clinic in the city of Metz (CL_{metz}), Moselle (department number 57), for a chemotherapy session ($Z51$) where he also had a chest radiography. Figure 7a shows the distribution of the length of sequences of care in our dataset, the median length being 54 (median size is equal to 11 stays).



(a) Distribution of the length of patient trajectories

(b) Agreement between sim_{ACS} versus sim_{LCS} according to sequence length

Fig. 7: Agreement measures and patient trajectories length distributions

A Comparison Between sim_{ACS} and sim_{LCS}

We compare sim_{ACS} with the two sim_{LCS} similarity measures (i.e., using either sequence length or size as previously defined in Section 4). For a given

[4] http://apps.who.int/classifications/apps/icd/icd10online/

[5] http://www.ameli.fr/accueil-de-la-ccam/index.php

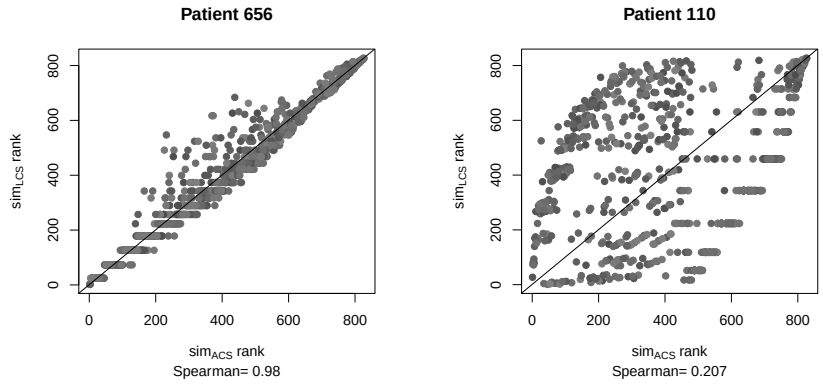
Patient	sim_{ACS} ranking	$sim_{LCS_{length}}$ ranking
P_{40}	291	468
P_{502}	126	126
P_{209}	484	491
⋮	⋮	⋮
P_{827}	129	126

Table 5: Example of rank correlation for patient P_{656}

sequence S, we ranked the 827 other sequences according to their similarity with S. Agreement between sim_{ACS} and sim_{LCS} rankings was then analyzed using Spearman's rank correlation coefficient. Figure 7b shows the link between a sequence length and its related Spearman coefficient [Myers and Well, 2003]. The agreement is high (Spearman $\rho > 0.8$) for short and long sequences. For medium length sequences, sim_{ACS} and sim_{LCS} have a very different behavior. In Figure 7b, sim_{LCS} has been computed using sequence length, but the same comments apply to the measure based on sequence size. A deeper comparison was performed by selecting a random sample of patient trajectories. For each sequence, rank correlation between similarity measures was analyzed with a scatter plot. An example of the similarity rankings is given in Table 5 for patient P_{656}. The scatter plots uses the sim_{ACS} ranking as an X-axis and the $sim_{LCS_{length}}$ ranking values for the Y-axis. Two different distributions of similarity values appear as shown in Figures 8a and 8b.

Strong correlation between sim_{ACS} and sim_{LCS} results. This case corresponds in our data set to the shortest and longest sequences, which have similar results for sim_{ACS} and sim_{LCS} (the optimal scatter plot in this case is a perfect diagonal). Their similarity with sequences of a much different length is very small and close to zero. This is the case, for example on Figure 8a, with patient 656 who went through 99 hospitalizations (one of the longest trajectories). Actually, whether similarity is measured with sim_{ACS} or sim_{LCS}, it decreases monotically when the size or the length difference between the two sequences increases. It can also be noticed that there is a high number of *ties* for sim_{LCS} (i.e., a repetition of the same similarity value for different sequences) as it can be seen on the plot in Figure 8a with different strata (i.e., bands).

Weak correlation between sim_{ACS} and sim_{LCS} results. This case corresponds to medium length trajectories, with two possible situations, as shown in Figure 8b. First situation: sim_{ACS} rankings are smaller than sim_{LCS} rankings (upper left part of the scatter plot). This happens when two patients have a long common subsequence but still have differences in some itemsets (i.e., hospitalizations). As sim_{ACS} is more sensitive to items variability it tends to output smaller similarity values than sim_{LCS}. Second situation: sim_{ACS} rankings are higher than sim_{LCS} rankings (lower right part of the scatter plot). This can be observed when two patients have similar hospitalizations but differ in their order (i.e., permutations). On the one hand, these differences



(a) Strong correlation (b) Weak correlation

Fig. 8: sim_{ACS} versus sim_{LCS}

tend to break long common subsequences and produce a small sim_{LCS} value. On the other hand, there is still a high number of common subsequences which is reflected through the higher sim_{ACS} similarity.

Clustering Healthcare Trajectories

Sim_{ACS} was used to build a similarity matrix between patient trajectories. A hierarchical clustering procedure was then applied using the *hclust* method in R software [R Core Team, 2012]. The number of clusters was set to 4 based on a priori knowledge from our experts. To assess the quality of our similarity measure, we describe each cluster with *"representative"* trajectories. To do so, we first extract frequent closed sequential patterns from our dataset by applying CloSpan [Yan et al., 2003] with a minimal support of 10%. Then, the support of the obtained patterns is computed in each of the 4 different clusters. Patterns having the highest variation of support between clusters were detected using a chi-squared measure (χ^2). Patterns with a high χ^2 and a high support in a given cluster are considered as distinguishing features of that cluster. After discussing these results with our medical expert, two criteria appeared to be related with the results of the clustering process, the *place of hospitalization* and the *length of the care trajectories*. We describe in the following the different clusters built with our similarity measure and its associated medical explanations.

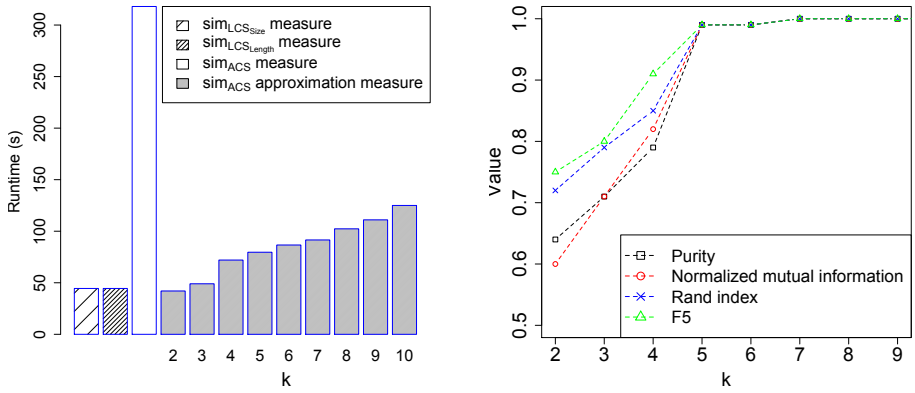
1. The pattern $\langle\{54, C34, GFFA\}\rangle$ has a high χ^2 and a high support in Cluster 1. Patients in that cluster underwent a pneumonectomy ($GFFA$) in a hospital from Meurthe-et-Moselle (department number 54). Patients usually have a short trajectory (median is 6 stays).

2. The pattern $\langle\{57, C34, GFFA\}\rangle$ is highly frequent in the second cluster (support is around 80%) but not in the others. It contains patients having underwent a pneumonectomy ($GFFA$) in a hospital from Moselle (department number 57). This cluster is characterized by longer patterns with repeated stays in the departement of Moselle, such as $\langle\{57\}\{57\}\{57\}\{57\}\{57\}\{57\}\rangle$. Patients in that cluster have a median trajectory length of 13.
3. The pattern $\langle\{54, Z51\}\{54, Z51\}\{54, Z51\}\{54, Z51\}\{54, Z51\}\{54, Z51\}\rangle$ is over represented in the third cluster (support is approximately 95%) than in any other cluster. It represents patients who have repeated chemotherapy sessions in Meurthe-et-Moselle department. The median trajectory length in that cluster is 37.
4. This cluster is similar to the third cluster (chemotherapy sessions) but with stays occurring in various places, especially in the border region of Alsace.

The clustering is based on a combination of different trajectory lengths and precise diagnoses or procedures such as pneumonectomy or chemotherapies. Our similarity measure is only based on the number of common subsequences and we were able to build clusters that were close to the knowledge that doctors and experts have on patients trajectories in the Lorraine Region. Furthermore, for our experts, these results are very encouraging as they correspond to the two main modalities in care for lung cancer: (i) surgery only or (ii) chemothery with or without surgery. They also highlight some important geographical characteristics in care trajectories and suggest that variability in the organization of cancer care is related to local factors.

Linial-Nisan Approximation

The runtime to build the similarity matrix for 828 patient trajectories without approximation is about 5 minutes. We applied the Linial-Nisan approximation on the PMSI dataset and built the similarity matrix with several values of $k \in [2, 10]$. Figure 9a shows the computation time to build the similarity matrix with varying values for k. To assess the quality of the approximations, we compare the clusters obtained using the Linial-Nisan approximation with the clusters obtained using sim_{ACS}. We use several criteria of clustering quality (i.e, Purity, Normalized mutual information, Rand index and F measure) proposed in [Manning et al., 2008] to evaluate the quality of the clusters obtained using the Linial-Nisan approximation. Figure 9b shows the results with several values of k. The higher the value of a cluster quality is, the more homogeneous the cluster is (i.e., it contains similar objects to the previously computed cluster with sim_{ACS}). This highlights the fact that approximating our similarity measure still yields good and competitive conclusions, with fast computation times (less than 120 seconds).



(a) Runtime for building the similarity matrix for 828 patient trajectories

(b) Clusters matching values obtained with the Linial-Nissan approximation

Fig. 9: Linial-Nisan cluster matching and runtime

8.3 Clustering Handwritten Assamese Symbols

Assamese Symbols

Our second batch of experiments was conducted with handwritten Assamese symbols from the UCI Machine Learning Repository[6]. Assamese is an official language used in the state of Assam in North East of India. It contains 183 symbols including 52 characters, 10 numerals and 121 conjunct consonants. The dataset is collected from 45 users who were instructed to write the symbols on a graphic tablet. The tablet program records the handwriting as a stream of (x, y) coordinate points using the appropriate pen position sensor along with the pen-up and pen-down switching. Figure 10 presents a sample of users writing the অ character. In the data set, each symbol for each user is modeled as a sequence of strokes. Each stroke consists of a set of points (x, y) representing the pen position on the tablet (i.e., a stroke represents an itemset). Each sequence representing a symbol was preprocessed by cropping and scaling the resolution of the symbol. Figure 11 illustrates the two-steps preprocessing applied on the character অ for a given user. The total number of sequences in the data set is 8235 (45 users drawing each 183 symbols). Figure 12 shows the distribution of the number of itemsets (i.e., strokes) for each symbol in the dataset. Our objective is to study the different Assamese handwriting techniques and build user clusters based on their handwriting style.

[6] http://archive.ics.uci.edu/ml/datasets/Online+Handwritten+Assamese+Characters+Dataset

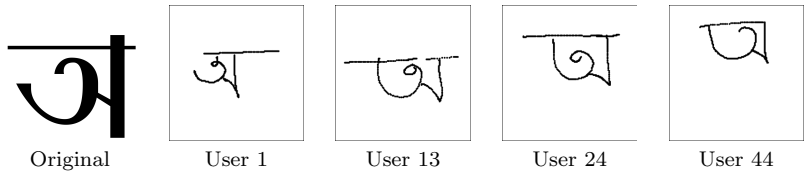


Fig. 10: Handwritten character অ

Fig. 11: Preprocessing steps on Character অ

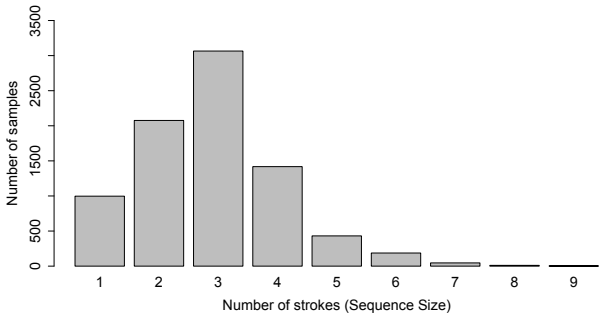
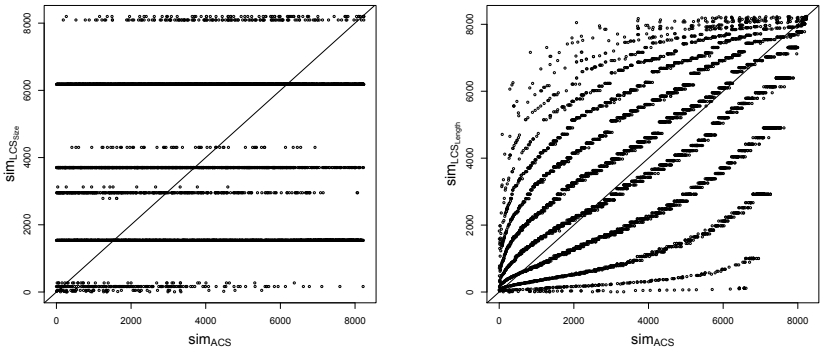


Fig. 12: Distribution of the number of strokes in the data set

A Comparison Between sim_{ACS} and sim_{LCS}

Similar to the healthcare trajectory data set experiments, we compared our similarity measure with the two versions of sim_{LCS}. The similarity between symbols was first computed using sim_{ACS} then with $sim_{LCS_{size}}$ and $sim_{LCS_{length}}$. For a given symbol X, we ranked the other symbols according to their similarity with X. The rank correlation was then analyzed with a scatter plot. Figure 13 represents the rank correlation between sim_{ACS} and sim_{LCS} when all the symbols are ranked according to the symbol ঝ. The non-correlation of the two measures is clear in Figures 13a and 13b. The remarkable strata displayed in Figure 13a are the result of the low variance over the similarity measures from $sim_{LCS_{size}}$. For example, character ঝ needs 3 strokes and so $sim_{LCS_{size}}$



(a) sim_{ACS} versus $sim_{LCS_{Size}}$ (b) sim_{ACS} versus $sim_{LCS_{Length}}$

Fig. 13: sim_{ACS} and sim_{LCS} rank correlation for character ঝ written by user 3

between ঝ and all the symbols which needs less than 3 strokes will always be limited to 3 possible values: $\frac{1}{3}$, $\frac{2}{3}$ or 1. The same reasoning applies to symbols with more than 3 strokes as $sim_{LCS_{Size}}$ will have at most 16 possible different values (the maximal number of strokes to write a symbol in the data set is 9). This is a limitation that is not present for the sim_{ACS} measure. To further illustrate this point, notice that for the user 3, $Sim_{LCS_{Size}}$ similarity value between ঝ and ক is $\frac{2}{3}$ and is exactly the same similarity value between ঝ and ল. Yet, sim_{ACS} returns a similarity value for ঝ and ক that is much lager than the similarity between ঝ and ল. The same drawback can be observed with the $sim_{LCS_{length}}$ measure, albeit to a lesser extent, and is illustrated in Figure 13b.

Clustering Assamese Handwritten Symbols

The goal of this experiment is to cluster the Assamese characters using sim_{ACS}. To interpret this experiment in a simple manner we will first detail the clustering process for 2 symbols ক্ক and ক্ত over a small sample or 3 randomly selected users, illustrated in Figure 14. The whole process over the 45 users and the 8235 symbols is discussed later.

A trivial case such as 6 sequences selected over 3 random users returns contrasting conclusions. Different hierarchical clustering results with sim_{ACS}, $sim_{LCS_{size}}$ and $sim_{LCS_{length}}$ similarity measures are pointed out in Figures 15a, 15b and 15c. Clustering with sim_{ACS} returns good results with a clear partition of the sequences in 2 groups respectively representing the 2 studied symbols. Clustering with sim_{LCS} on the other side yields more subtle results

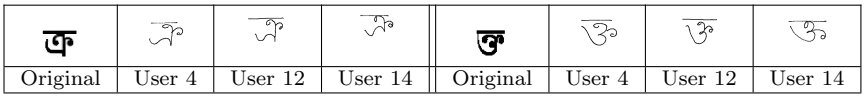
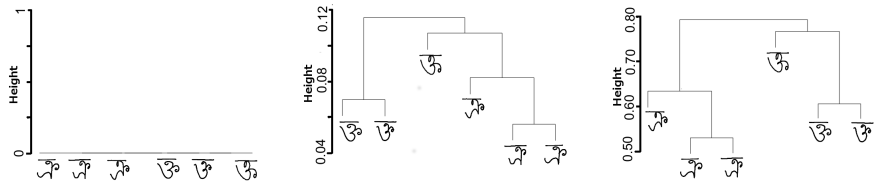
							
Original	User 4	User 12	User 14	Original	User 4	User 12	User 14

Fig. 14: How the users 4, 12 and 14 draw the characters ক and ঙ



(a) $sim_{LCS_{Size}}$ (b) $sim_{LCS_{Length}}$ (c) sim_{ACS}

Fig. 15: Hierarchical clustering results for the users 4, 12 and 14 draw the characters ক and ঙ

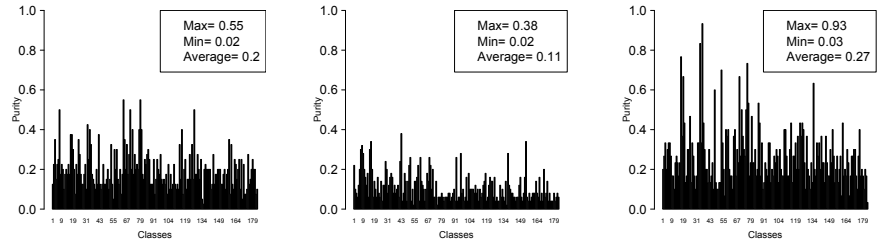
with $sim_{LCS_{size}}$ clustering all the sequences in the same group with equidistant similarities. $sim_{LCS_{length}}$ gives better results but still incorrectly clusters one of the characters.

	Purity	NMI	RI	F_5
$sim_{LCS_{Size}}$	0.45	0.11	0.5	0.43
$sim_{LCS_{Length}}$	0.63	0.35	0.67	0.52
sim_{ACS}	**0.86**	**0.6**	**0.85**	**0.85**

Table 6: Clustering evaluation of 90 users draw two characters ক and ঙ

In the following experiment, we extended the 2 symbols clustering to the 45 users with a total of 90 sequences. We generated the similarity matrix, applied hierarchical clustering and cut the hierarchical clustering tree at the second level from the top. We computed the Purity, Normalized Mutual Information (NMI), Rand Index (RI) and F measure to evaluate how well the clusters, obtained using the three measures, matched with the original clusters (i.e, the 2 original clusters contain the 45 different handwritten characters ঙ and ক). The results are reported in Table 6. Notice that sim_{ACS} returns for this experiment more homogeneous clusters.

In the final experiment, we built a similarity matrix using sim_{ACS} for the 45 users and the 8235 symbols. A hierarchical clustering procedure is also applied. We cut the tree at the 183^{th} level (the number of clusters corresponding to the number of symbols in the alphabet). Our goal is to de-



(a) $sim_{LCS_{Size}}$ (b) $sim_{LCS_{Length}}$ (c) sim_{ACS}

Fig. 16: Purity value for each class of handwritten symbols

tect whether we can group all the users who draw the same character in one class. The results with the Purity measure are reported in Figure 16. Other detailed results for this experiment are described at the following url: `http://www.loria.fr/~eegho/acs/`.

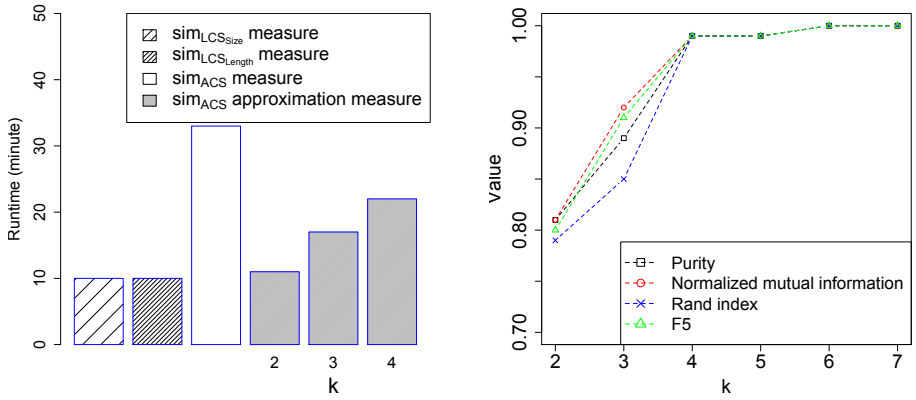
Linial-Nisan approximation

We built the similarity matrix with several value of $k \in [2, 9]$. Our goal is twofold: (i) compare the clustering quality based on the approximation method and (ii) compare the runtime. We compared the clusters obtained using the Linial-Nisan approximation with the clusters obtained using sim_{ACS}. Figure 17b reports the values of each cluster quality measure with several different values of k. We noticed that starting from $k = 4$, the clusters are identical to the clusters built using sim_{ACS}. In terms of runtime, Figure 17a reports the different time gains when building the similarity matrix with sim_{ACS}.

9 Conclusion

In this paper, we study the problem of counting all common subsequences between two sequences of itemsets. We present theoretical results and an efficient dynamic programming algorithm (ACS) to count the number of common subsequences between two sequences. This solution allows us to define in a simple and intuitive manner a similarity measure, denoted sim_{ACS}, between two sequences S and T. In addition, we propose an approximation method to speed up the computation for long sequences. This similarity has been successfully applied for the analysis of real-world healthcare, handwritten symbols and synthetic data sets.

For future work, we plan to apply the measure on various sequence data sets (more precisely trajectory mining, molecular bioinformatics and text classification). We also intend to compare our measure with kernel methods for sequence classification.



(a) Runtime for building the similarity matrix for 8325 symbols

(b) Clusters matching values obtained with the Linial-Nissan approximation

Fig. 17: Linial-Nisan cluster matching and runtime

References

Donald J. Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD Workshop*, pages 359–370, 1994.

C. Chothia and M. Gerstein. Protein evolution. how far can sequences diverge? *Nature*, 6617(385):579–581, 1997.

Cees Elzinga, Sven Rahmann, and Hui Wang. Algorithms for subsequence combinatorics. *Theor. Comput. Sci.*, 409(3):394–404, 2008.

Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, SIGMOD '94, pages 419–429, New York, NY, USA, 1994. ACM.

Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

Javier Herranz, Jordi Nin, and Marc Sole. Optimal symbol alignment distance: A new distance for sequences of symbols. *IEEE Transactions on Knowledge and Data Engineering*, 23:1541–1554, 2011.

D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, June 1975.

Mohammed J. Zaki Karlton Sequeira. Admit: Anomaly-base data mining for intrusions. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2002.

Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 406–417. VLDB Endowment, 2002.

Christina Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for svm protein classification. *Pacific Symposium On Biocomputing*, 575(50):564–575, 2002.

V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

Nathan Linial and Noam Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Fariha Muzaffar, Bushra Mohsin, Farah Naz, and Lecturer Farooq Jawed. Dsp implementation of voice recognition using dynamic time warping algorithm. *IEEE Explore*, pages 1–7, 2005.

J. L. Myers and A. D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum Associates, New Jersey, 2003.

Jose Oncina and Marc Sebban. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recogn.*, 39(9):1575–1587, September 2006.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, 1(9):56–68, 1991.

Joan Serrà, Holger Kantz, Xavier Serra, and Ralph G. Andrzejak. Predictability of music descriptor time series and its application to cover song detection. *IEEE Transactions on Audio, Speech & Language Processing*, 20(2): 514–525, 2012.

Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn J. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors, *KDD*, pages 216–225. ACM, 2003.

Hui Wang and Zhiwei Lin. A novel algorithm for counting all common subsequences. In *Proceedings of the 2007 IEEE International Conference on Granular Computing*, GRC '07, pages 502–, Washington, DC, USA, 2007. IEEE Computer Society.

S.J. Wodak and J. Janin. Structural basis of macromolecular recognition. *Adv Protein Chem*, 61:9–73, 2002.

Tengke Xiong, Shengrui Wang, Qingshan Jiang, and Joshua Zhexue Huang. A new markov model for clustering categorical sequences. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 854–863, Washington, DC, USA, 2011. IEEE Computer Society.

Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large datasets. In *In SDM*, pages 166–177, 2003.

Qiang Yang and Haining Henry Zhang. Web-log mining for predictive web caching. *IEEE Trans. on Knowl. and Data Eng.*, 15(4):1050–1053, July 2003. ISSN 1041-4347.

Appendix

Proof of Lemma 1

Let $T = \langle T_1, \ldots, T_m \rangle$ be a sequence that is counted multiple times; i.e., $T \in (\varphi(S) \circ \mathcal{P}_{\geq 1}(Y)) \cap \varphi(S)$. Clearly $T_m \in \mathcal{P}_{\geq 1}(Y)$ as otherwise T would not have been in $\varphi(S) \circ \mathcal{P}_{\geq 1}(Y)$. Let ℓ denote $\max\{j | T_m \subseteq S[j]\}$. Since $T \in \varphi(S)$, such ℓ must exist. Then, $\ell \in L(S, Y)$, since ℓ is the largest index for which $S[\ell] \cap Y$ includes T_m. Therefore, $T \in S^{\ell-1} \circ (\mathcal{P}_{\geq 1}(S[\ell] \cap Y))$ for a $\ell \in L(S, Y)$. □

Proof of Theorem 1

The proof is a simple application of the inclusion-exclusion principle to compute the cardinality of the union of Lemma 1:

$$R(S,Y) = \left| \bigcup_{\ell \in L} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap Y) \right\} \right| = \sum_{K \subseteq L} (-1)^{|K|+1} \left| \bigcap_{\ell \in K} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap Y) \right\} \right|$$

The proof is completed by the following two observations:

$$set_K := \bigcap_{\ell \in K} \left\{ \varphi(S^{\ell-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap Y) \right\} = \varphi(S^{\min(K)-1}) \circ \mathcal{P}_{\geq 1}((\cap_{k \in K} S[k]) \cap Y)$$

Indeed; any sequence of length m in set_K has $T^{m-1} \in S^{\min(K)-1}$, and $T_m \in \mathcal{P}_{\geq 1}(S[k] \cap Y)$, for all $k \in K$ and

$$\left| \varphi(S^{\min(K)-1}) \circ \mathcal{P}_{\geq 1}((\cap_{k \in K} S[k]) \cap Y) \right| = \left| \phi(S^{\min(K)-1}) \right| \cdot \left(2^{|(\cap_{k \in K} S[k]) \cap Y|} - 1 \right)$$

□

Proof of Lemma 2

Let $Z = \langle Z_1, \ldots, Z_m \rangle$ be a new subsequence that is added to $\varphi(S, T)$ after concatenating sequence S with itemset Y; i.e., $Z \in \varphi(S \circ Y, T) \setminus \varphi(S, T)$. Clearly $Z_m \in \mathcal{P}_{\geq 1}(Y)$ as otherwise Z would not have been added to $\varphi(S \circ Y, T)$. Let $\ell' = \max\{j | Z_m \subseteq T[j]\}$. Since $Z \in \varphi(S \circ Y, T)$, with $Z \preceq T$, then such $\ell' \in L(T, Y)$ must exist. ℓ' is the largest index for which $T[\ell'] \cap Y$ includes Z_m. Therefore, $Z \in \varphi(S, T^{\ell'-1}) \circ (\mathcal{P}_{\geq 1}(T[\ell'] \cap Y))$ for a $\ell' \in L(T, Y)$.

Let $W = \langle W_1, \ldots, W_m \rangle$ be a sequence that is counted multiple times; i.e., $W \in (\varphi(S, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(T[\ell'] \cap Y)) \cap \varphi(S, T)$ where $\ell' \in L(T, Y)$. Clearly $W_m \in \mathcal{P}_{\geq 1}(T[\ell'] \cap Y)$ as otherwise W would not have been in $\varphi(S, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(T[\ell'] \cap Y)$. Let $\ell = \max\{j | W_m \subseteq S[j]\}$. Since $W \in \varphi(S, T)$, such $\ell \in L(S, Y)$ must exist, since ℓ is the largest index for which $S[\ell] \cap Y$ includes Z_m. Therefore, $Z \in \varphi(S^{\ell-1}, T^{\ell'-1}) \circ (\mathcal{P}_{\geq 1}(S[\ell] \cap T[\ell'] \cap Y))$ for $\ell \in L(S, Y)$ and $\ell' \in L(T, Y)$. □

Proof of Theorem 2

Case 1: No items in Y appear in any itemset of S and T, in this case the set of all common distinct subsequences between $S \circ Y$ and T is exactly the same set of all common distinct subsequences between S and T. Hence, $\phi(S \circ Y, T) = \phi(S, T)$.

Case 2: If at least an item in Y appears in either one of the sequences S or T (or both), then $\varphi(S \circ Y, T)$ is expressed as the union of the set of all common distinct subsequences between S and T with the set of added sequences $\mathcal{A}$ *without* the set of repeated sequences $\mathcal{R}$. Formally,

$$\varphi(S \circ Y, T) = \varphi(S, T) \cup \mathcal{A} \backslash \mathcal{R} \tag{8}$$

with

$$\mathcal{A} = \left\{ \bigcup_{\ell' \in L(T,Y)} \varphi(S, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(T[\ell'] \cap Y) \right\}$$

$$\mathcal{R} = \left\{ \bigcup_{\ell \in L(S,Y)} \left\{ \bigcup_{\ell' \in L(T,Y)} \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right\}$$

Notice that because these three sets are disjoint, the cardinality of $\varphi(S \circ Y, T)$ can be simply expressed as $|\varphi(S \circ Y, T)| = |\varphi(S, T)| + |\mathcal{A}| - |\mathcal{R}|$. Using the inclusion-exclusion principle, $|\mathcal{A}|$, denoted as $A(S, T, Y)$ can be written as,

$$\begin{aligned} A(S,T,Y) &= \left| \left\{ \bigcup_{\ell \in L(T,Y)} \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geq 1}(T[\ell] \cap Y) \right\} \right| \\ &= \sum_{K \subseteq L(T,Y)} (-1)^{|K|+1} \left| \bigcap_{\ell \in K} \left\{ \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geq 1}(T[\ell] \cap Y) \right\} \right| \end{aligned} \tag{9}$$

$A(S, T, Y)$ is completed by the following two observations:

$$\begin{aligned} set_K &:= \bigcap_{\ell \in K} \left\{ \varphi(S, T^{\ell-1}) \circ \mathcal{P}_{\geq 1}(T[\ell] \cap Y) \right\} \\ &= \varphi(S, T^{\min(K)-1}) \circ \mathcal{P}_{\geq 1}((\cap_{k \in K} T[k]) \cap Y) \end{aligned}$$

And, the second observation:

$$|set_K| = \phi(S, T^{\min(K)-1}) \cdot \left(2^{|(\cap_{k \in K} T[k]) \cap Y|} - 1\right)$$

$A(S, T, Y)$ can be written as,

$$A(S,T,Y) = \sum_{K \subseteq L(T,Y)} (-1)^{|K|+1} \left(\phi(S, T^{\min(K)-1}) \cdot \left(2^{|\left(\bigcap_{j \in K} T[j]\right) \cap Y|} - 1 \right) \right)$$

The same inclusion-exclusion reasoning applies to the cardinality of $\mathcal{R}$, denoted $R(S, T, Y)$

$$R(S,T,Y) = \left| \left\{ \bigcup_{\ell \in L(S,Y)} \left\{ \bigcup_{\ell' \in L(T,Y)} \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right\} \right|$$

$$= \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\sum_{K' \subseteq L(T,Y)} (-1)^{|K'|+1} \left| \bigcap_{\ell \in K} \bigcap_{\ell' \in K'} \left\{ \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap T[\ell'] \cap Y) \right\} \right| \right)$$

The final result follows after noticing that,

$$set_{K,K'} = \bigcap_{\ell \in K} \bigcap_{\ell' \in K'} \varphi(S^{\ell-1}, T^{\ell'-1}) \circ \mathcal{P}_{\geq 1}(S[\ell] \cap T[\ell'] \cap Y)$$

$$set_{K,K'} = \varphi(S^{\min(K)-1}, T^{\min(K')-1}) \circ \mathcal{P}_{\geq 1}((\cap_{k \in K} S[k]) \cap \left(\cap_{k' \in K'} T[k']\right) \cap Y)$$

$R(S,T,Y)$ can be written as,

$$R(S,T,Y) = \sum_{K \subseteq L(S,Y)} (-1)^{|K|+1} \left(\sum_{K' \subseteq L(T,Y)} (-1)^{|K'|+1} \cdot f(K,K') \right)$$

where:

$$f(K,K') = \phi(S^{\min(K)-1}, T^{\min(K')-1}) \cdot \left(2^{|\left(\bigcap_{j \in K} S[j]\right) \cap \left(\bigcap_{j' \in K'} T[j']\right) \cap Y|} - 1 \right)$$

□

Details of Linial-Nisan Approximation for Example 7

To do the Linial-Nisan approximation, remark that $N = |L| = 9$ and $K = \lceil \sqrt{N} \rceil = 3$. The vector of Linial-Nisan coefficients is defined as $\overrightarrow{\alpha} = (\alpha_1^{3,9}, \alpha_2^{3,9}, \alpha_3^{3,9}) = \overrightarrow{t} \cdot \mathcal{M}^{-1}$ where $\mathcal{A}$ is the matrix whose (i,j) entry is $\binom{j}{i}$. The inverse matrix $\mathcal{A}^{-1}(i,j)$ is defined as $(-1)^{i+j}\binom{j}{i}$. In our example,

$$\mathcal{M}^{-1} = \begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}$$

$\overrightarrow{t} = (q_{K,N}(1), q_{K,N}(2), \ldots, q_{K,N}(K))$ is the vector of linearly transformed Chebyshev polynomials and is computed using the polynomial $T_K(x)$ as follows:

$$q_{3,9}(1) = 1 - \frac{T_3(\frac{2-(9+1)}{9-1})}{T_3(\frac{-(9+1)}{9-1})} = 1 - \frac{T_3(-1)}{T_3(-\frac{10}{8})} = 1 - \frac{-1}{-4,06} = 0.75$$

$$q_{3,9}(2) = 1 - \frac{T_3(\frac{4-(9+1)}{9-1})}{T_3(\frac{-(9+1)}{9-1})} = 1 - \frac{T_3(-\frac{6}{8})}{T_3(-\frac{10}{8})} = 1 - \frac{0.56}{-4,06} = 1.13$$

$$q_{3,9}(3) = 1 - \frac{T_3(\frac{6-(9+1)}{9-1})}{T_3(\frac{-(9+1)}{9-1})} = 1 - \frac{T_3(-\frac{4}{10})}{T_3(-\frac{10}{8})} = 1 - \frac{1}{-4,06} = 1.24$$

This vector is necessary to solve the system of linear equations:

$$\begin{aligned} \overrightarrow{\alpha} &= (\alpha_1^{3,9}, \alpha_2^{3,9}, \alpha_3^{3,9}) = \overrightarrow{t} \cdot \mathcal{M}^{-1} \\ &= \begin{pmatrix} 0.75 & 1.13 & 1.24 \end{pmatrix} \cdot \begin{pmatrix} 1 & -2 & 3 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0.75 & -0.36 & 0.1 \end{pmatrix} \end{aligned}$$

A solution to the system above is given by $\alpha_1^{3,9} = 0.75$; $\alpha_2^{3,9} = -0.36$; $\alpha_3^{3,9} = 0.1$. The real numbers $\alpha_k^{3,9}$ can now be used to approximate the inclusion-exclusion formula in the correction term as follows:

$$R_{LN}(S^9, S[10]) = \sum_{k=1}^{3} \alpha_k^{3,9} \sum_{\substack{O \subseteq L(S,Y) \\ |O|=k}} \phi_{LN}(S^{\min(O)-1}) \cdot \left(2^{|\left(\bigcap_{j \in O} S[j]\right) \cap S[10]|} - 1\right)$$
$$= 9{,}298\,127\,9 \cdot 10^{26}$$

Notice here that the formula contains only $\sum_{i=1}^{3} \binom{9}{i}$ terms, which is already a significant computation gain in comparison with the $\sum_{i=1}^{9} \binom{9}{i}$ terms in the classical approach. Finally, the approximated number of distinct subsequences for sequence S is $\phi_{LN}(S^{10}) = 2^{|\{a,b,c,d,e,f,g,h,i,j,,k\}|} \cdot \phi_{LN}(S^9) - R_{LN}(S^9, S[10]) = 2{,}524\,495\,6 \cdot 10^{30}$.