Available online at www.sciencedirect.com
ScienceDirect
Information Processing and Management 43 (2007) 1005–1019
INFORMATION PROCESSING & MANAGEMENT
www.elsevier.com/locate/infoproman

s-grams: Defining generalized *n*-grams for information retrieval

Anni Järvelin [a,*], Antti Järvelin [b], Kalervo Järvelin [a]

[a] *University of Tampere, Department of Information Studies, FIN-33014 University of Tampere, Finland*
[b] *University of Tampere, Department of Computer Sciences, FIN-33014 University of Tampere, Finland*

Received 1 June 2006; received in revised form 24 September 2006; accepted 26 September 2006
Available online 22 November 2006

Abstract

n-grams have been used widely and successfully for approximate string matching in many areas. *s*-grams have been introduced recently as an *n*-gram based matching technique, where di-grams are formed of both adjacent and non-adjacent characters. *s*-grams have proved successful in approximate string matching across language boundaries in Information Retrieval (IR). *s*-grams however lack precise definitions. Also their similarity comparison lacks precise definition. In this paper, we give precise definitions for both. Our definitions are developed in a bottom-up manner, only assuming character strings and elementary mathematical concepts. Extending established practices, we provide novel definitions of *s*-gram profiles and the L_1 distance metric for them. This is a stronger string proximity measure than the popular Jaccard similarity measure because Jaccard is insensitive to the counts of each *n*-gram in the strings to be compared. However, due to the popularity of Jaccard in IR experiments, we define the reduction of *s*-gram profiles to binary profiles in order to precisely define the (extended) Jaccard similarity function for *s*-grams. We also show that *n*-gram similarity/distance computations are special cases of our generalized definitions.

Keywords: Information retrieval; Approximate string matching; *n*-grams; *s*-grams

1. Introduction

s-gram matching is an approximate string matching technique, where the text strings compared are decomposed into *s*-grams, i.e., into fixed length substrings. The degree of similarity between the strings can be computed by comparing their *s*-gram sets. The idea dates back to Shannon's Mathematical Theory of Communication (1948) and it has in earlier literature also been referred to as *n*-grams (e.g. Pfeiffer, Poersch, & Fuhr, 1996; Robertson & Willett, 1998) or *q*-grams (e.g. Ukkonen, 1992; Zobel & Dart, 1996). Typically *n*-grams consist of adjacent character pairs, triples etc. of the original strings. The name *s*-gram originates from a study by Pirkola, Keskustalo, Leppänen, Känsälä, and Järvelin (2002), who devised a novel classified *s*-gram matching technique. In this technique the *s*-grams are formed of both adjacent and non-adjacent characters of

* Corresponding author.
E-mail addresses: anni.jarvelin@uta.fi (A. Järvelin), antti.jarvelin@uta.fi (A. Järvelin), kalervo.jarvelin@uta.fi (K. Järvelin).

0306-4573/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2006.09.016

the text strings and classified into sets for computing the similarity. The name *s*-gram comes from the word skip and points to the idea that a number of characters are skipped when the substrings (*s*-grams) are formed. In the original study, Pirkola et al. (2002) found classified *s*-grams better than the adjacent *n*-grams in matching cross-lingual spelling and monolingual morphological variants.

In the literature, the terminology used for referring to *n*-grams and *s*-grams has varied. In this paper, the term *n-gram* will be used to refer to the adjacent *n*-grams only. The term *s-gram* is used when referring to both the adjacent and non-adjacent *s*-grams. Thus *n*-grams are a special case of *s*-grams, where zero characters are skipped when the substrings are formed. Di-grams are conventional *n*-grams where the substrings' length is two ($n = 2$), for tri-grams n is three. When referring to different length *s*-grams expressions such as *s*-di-grams and *s*-tri-grams will be used. Furthermore, if also the skip length of an *s*-gram needs to be specified, *s*-grams of length n and skip-length k will be referred as $s_{n,k}$-grams.

n-gram matching is a widely used technique both within IR and outside (e.g. Downie & Nelson, 2000; Grossman & Frieder, 2004; Keskustalo, Pirkola, Visala, Leppänen, & Järvelin, 2003; McNamee & Mayfield, 2003; O'Rourke, Robertson, & Willettt, 1997; Pirkola et al., 2002; Pfeiffer et al., 1996; Robertson & Willett, 1998; Toivonen, Pirkola, Keskustalo, Visala, & Järvelin, 2005; Uitdenbogerd & Zobel, 2002; Ukkonen, 1992; Ullman, 1977; Zobel & Dart, 1996). Ukkonen (1992) gives a formal definition to *n*-grams. However, the classified *s*-grams found superior to *n*-grams in information retrieval applications lack such defendable definitions. Also their similarity comparison (Pirkola et al., 2002) lacks a stringent definition. This paper provides stringent definitions both for *s*-grams and their classified similarity comparison. We do this firstly by formalizing the *similarity* comparison proposed in (Pirkola et al., 2002). Secondly, we propose the use of *distance* measures used earlier in the literature and extend them for classified *s*-grams. As these have not been tested empirically so far, our work suggests that new empirical work be done.

The rest of the paper is organized as follows. Approximate string matching techniques and the use of *s*-grams in IR are discussed in Section 2. In Section 3 *s*-grams are presented in more detail and an example application of the *s*-gram matching technique is given. Section 4 discusses prior formalizations of *n*-grams and Section 5 presents our formalization of *s*-grams with *n*-grams as a special case. Finally, Section 6 provides discussion and conclusions.

2. Approximate string matching techniques in IR

Approximate string matching techniques are based on the expectation that two strings that have similar strings of characters will have similar meaning and should therefore be regarded as being equivalent (Robertson & Willett, 1998). Recognizing and measuring similarity in strings is useful in several situations in IR as both natural morphological word form variation and variation due to e.g. typing errors or OCR (optical character recognition) errors occur in databases. Recognizing such word form variants as occurrences of the same string is essential as different forms of a word represent the same concept and are therefore equal from the standpoint of users' requests (Pirkola et al., 2002).

Cross-lingual spelling variation is a type of word form variation occurring between languages. Especially related languages often share a number of words that have the same origin (e.g. Latin based words) and only differ due to the orthographical differences between the languages. Proper names and technical terms are typical examples of words where cross-lingual spelling variation occurs (e.g. Brussels in Finnish is Bryssel). Approximate string matching can be used in cross-language information retrieval (CLIR) to recognize cross-lingual spelling variants as equivalents. In CLIR, a source language query is typically translated into the target language with machine-readable dictionaries. The general translation dictionaries often do not cover most proper names and technical terms, which therefore remain untranslatable in queries. They can nevertheless often be recognized as similar by approximate string matching, and therefore translated. This has a positive effect on query translation as proper names and technical terms often are important query keys (Pirkola et al., 2002).

The approximate similarity between strings can be measured with different methods. Here, Soundex and its variant Phonix, Edit distance (ED), Longest common subsequence (LCS) and the *s*-gram technique will be discussed. Soundex is an early phonetic matching scheme of Odell and Russell from 1918 (Hall & Dowling, 1980) and Phonix its newer variant developed by Gadd in late 1980's (Zobel & Dart, 1996). Matching schemes

based on comparing the phonetic similarity of strings are language dependent techniques. Soundex and Phonix were developed for the English language but can be modified for other languages (Pfeiffer et al., 1996). They use phonetic codes based on grouping similar sounding letters together. Strings sharing the same code are assumed to be similar (Zobel & Dart, 1996). Phonix also uses rules for letter-group transformations to provide context for the phonetic codes. Especially Soundex makes quite commonly the error of transforming dissimilar-sounding strings into the same code, and Pfeiffer et al. (1996) found both Soundex and Phonix clearly inferior to *n*-grams in proper name matching. Zobel and Dart (1996) tested various string similarity techniques for phonetic matching and found that an edit distance variant Editex that uses the letter-groupings of Soundex outperformed both Soundex and Phonix.

Edit distances (ED) are distance measures, which count the minimal number of single character instertions, deletions and replacements needed to transform one string to another. Different operations can be assigned different costs, depending on their likelihood (Navarro, 2001). ED is sometimes referred to as Damerau–Levenstein metric as Damerau and Levenstein developed the metric separately during the 1960's. Damerau developed an early edit distance measure for handling spelling errors, which accepts a difference of one insertion, deletion, replacement or transposition of a character in the strings compared (Damerau, 1964). Pfeiffer et al. (1996) and Zobel and Dart (1996) studied different approximate string matching techniques in name matching and ED was tested in both studies. While both found that combining evidence from different string matching techniques was the best solution, the results concerning the individual techniques and specially ED and *n*-grams diverged: Pfeiffer et al. (1996) found *n*-grams clearly a better technique than the ED, whereas Zobel and Dart (1996) reported that ED outperformed *n*-grams.

LCS is a string matching technique that measures the length of the longest pairing of characters that can be made between two strings, so that the pairings respect the order of the letters (Navarro, 2001). O'Rourke et al. (1997) found LCS to be the best method for matching historical word form variants (compared to di- and tri-grams). They nevertheless concluded that di-grams would be the method of choice in an operational system due to LCS's high demand on computational time. Keskustalo et al. (2003) compared *s*-grams to different length *n*-grams, ED and LCS in matching cross-lingual spelling variants and found that, where ED often outperforms the adjacent *n*-grams, the classified *s*-grams performed better than ED for all the six language pairs studied. LCS was always inferior to *s*-grams and ED.

The modern applications of the *n*-gram matching in IR are discussed in, e.g., Grossman and Frieder (2004) and in Robertson and Willett (1998). The *n*-gram matching, and its generalization *s*-gram matching, are language independent techniques and can therefore be easily applied to all languages in which the strings consist of space- or punctuation-delimited sequences of characters (Robertson & Willett, 1998). Grossman and Frieder (2004) point to *n*-gram applications in handling OCR errors, spelling error correction, text compression, authorship detection, and discuss applications in traditional text retrieval at some length. The present article has its focus on *s*-gram matching in information retrieval (IR) and above all in cross-language information retrieval (CLIR) and therefore mainly IR research is discussed. Results from several studies (Pfeiffer et al., 1996; Pirkola et al., 2002; Keskustalo et al., 2003) support the proposition that *s*-gram matching is the choice approximate string matching technique in IR and CLIR and therefore an extensive discussion of other approximate matching techniques is not provided here.

Biological and genetic IR are application areas of *s*-grams, where the strings compared can be thousands of characters long and therefore higher value of n may be used (Altschul, Gish, Miller, Myers, & Lipman, 1990; Bishop & Thompson, 1984; Miller, Gurd, & Brass, 1999). Califano and Rigoutsos (1993) proposed a $s_{n,k}$-gram method for matching molecule biological sequences. For large values of n and k (e.g., $n > 10$) the proposed method led to serious efficiency problems due to the size of the index needed for retrieval. Similar problems are faced when word-spanning *n*-grams are used for indexing document collections instead of word-based indexing (McNamee & Mayfield, 2004): When the strings decomposed to *n*-grams are long (queries, documents) and the number of unique *n*-grams in any collection is bounded by $|\Sigma|^n$, where $|\Sigma|$ denotes the size of an alphabet Σ, the index size grows rapidly with n. The value of n appropriate for *n*-gram based indexing varies with the language considered. The optimal value of n for European languages is between 4 and 5 (McNamee & Mayfield, 2004), as for Chinese, where word based indexing faces difficulties because of the short comings of the programs used for recognizing word boundaries, di-gram based indexing has been popular (Chen, He, Xu, Gey, & Meggs, 1997) and index size is not a problem.

The definitions given for n-grams in the present article hold for these applications. The approaches are nevertheless different: in n-gram based indexing the value of n needs to be set high enough to ensure the discriminating power of the indexing features (n-grams). In the present article the n-grams are seen as a word level string similarity measure used before a query is matched against a document collection. The problems with long text passages and high values of n are not addressed with length, as the average word length is well under ten characters and as the focus is on (CL)IR, where n can be limited to low values. The effect that the classified s-grams special features have for the index size is discussed in more detail in Section 6.

3. *s*-grams

3.1. *s*-gram basics

The classified s-gram technique was introduced as a solution for monolingual morphological and cross-lingual spelling variation problems in IR (Pirkola et al., 2002) and its performance has been tested with several language pairs (Pirkola et al., 2002; Keskustalo et al., 2003). In the classified s-gram matching technique s-grams are divided into categories (classes) on the basis of the number of the skipped characters and only the s-grams belonging to the same class are compared to each other. *Skip-gram class* indicates the skip length used when generating a set of s-grams belonging to a class. Two or more skip-gram classes may also be combined into more general skip-gram classes (Pirkola et al., 2002; Keskustalo et al., 2003). The *character combination index* (CCI) then indicates the set of all the skip-gram classes to be formed from a string. Different combinations of skipped characters can be used. For example the CCI $\{\{0\},\{1,2\}\}$ means that two skip-gram classes are formed from a string: one with conventional n-grams formed of adjacent characters and one with s-grams formed both by skipping one and two characters. An example of forming the skip-gram classes is given in Table 1. The largest value in a skip-gram class is called the spanning length of the skip-gram class (Keskustalo et al., 2003), e.g., for skip-gram class $\{0,1\}$, the spanning length is one.

Different skip-gram classes carry forward different evidence from their host string and s-grams can therefore be tuned to handle different phenomena by adjusting the skip-gram classes. Keskustalo et al. (2003) gives a good presentation on how the skip-gram classes relate to different variation in strings from the cross-lingual spelling variation point of view. Cross-lingual spelling variation typically involves single character insertions, deletions and substitutions, or their two-character combinations. For example transforming Swedish variant *heksaklorid* into the English *hexachloride* involves a single insertion (e) and combinations of deletion and substitution ($ks \rightarrow x$) and substitution and insertion ($k \rightarrow ch$) Keskustalo et al. (2003). Therefore it is reasonable to use only skip-gram classes with spanning length of two or less when matching cross-lingual spelling variants. Also spelling errors typically involve character substitution, insertion, deletion and reversal errors and their combinations (Ullman, 1977). The spanning length of s-grams can be restricted to two or less again, as Zamora, Pollock, and Zamora (1981) have reported that most of the misspelled strings in text databases only contain a single error.

It is common to use padding spaces in the beginning and in the end of the strings when forming s-grams. The padding helps to get the characters at the beginning and at the end of a string properly presented in its s-gram set. For conventional n-grams it is common to use a padding of $n-1$ characters (Robertson & Willett, 1998). For s-grams a padding that varies together with the length of the substring (n) and the number of the skipped characters can be used. In accordance with these rules, the set of padded $s_{2,0}$-grams for the string *abra-*

Table 1
The skip-gram classes for forming the s-di-grams with different CCIs for the string *abradacabra*

Type	CCI	s-gram classes
$s_{2,0}$	$\{0\}$	$\{ab, br, ra, ad, da, ac, ca, ab, br, ra\}$
$s_{2,1}$	$\{1\}$	$\{ar, ba, rd, aa, dc, aa, cb, ar, ba\}$
$s_{2,2}$	$\{2\}$	$\{aa, bd, ra, ac, da, ab, cr, aa\}$
$s_{2,\{1,2\}}$	$\{1,2\}$	$\{ar, ba, rd, aa, dc, cb, ar, bd, ra, ac, da, ab, cr\}$
$s_{2,\{\{0\},\{1,2\}\}}$	$\{\{0\},\{1,2\}\}$	$\{\{ab, br, ra, ad, da, ac, ca, ab, br, ra\}, \{ar, ba, rd, aa, dc, cb, ar, bd, ra, ac, da, ab, cr\}\}$

dacabra is: {_*a*, *ab*, *br*, *ra*, *ad*, *da*, *ac*, *ca*, *ab*, *br*, *ra*, *a*_}. Keskustalo et al. (2003) tested with several languages different types of padding spaces for conventional di-grams, tri-grams and *s*-di-grams, and found that using padding spaces both in the beginning and the end of the words gave the best results. However, leaving out the padding spaces can help down-weighting the derivational suffixes and prefixes, when handling morphological variation or cross-lingual spelling variants in inflected forms. For example, the use of the beginning padding only has been found beneficial for Finnish, which is an inflectionally complex suffix language (Pirkola et al., 2002).

3.2. Motivating example

The typical dictionary-based query translation approach to CLIR has a downside in the constant need for updating the dictionaries and in that different dictionaries are needed for each language pair. These features can make the approach costly and replacing it by cheaper, language independent techniques would be desirable. Between closely related languages that share a high number of cross-lingual spelling variants, approximate string matching techniques can be used for a simpler, fuzzy query translation technique. A fuzzy *s*-gram query translation between the Scandinavian languages Norwegian and Swedish is explored here to give an example of a *s*-gram matching application.[1] Norwegian and Swedish are closely related languages that share a high number of cross-lingual spelling variants: around 90% of the vocabularies of the languages are similar having only some orthographical and inflectional differences (Baròdal, Jörgensen, Larsen, & Martinussen, 1997). This provides a good basis for fuzzy translation.

Our fuzzy query translation experiment was done using adjacent di-grams and classified *s*-di-grams for translating Norwegian search topics to Swedish. The goal was to, with fuzzy techniques, reach translation quality sufficient to enable effective searching of a Swedish document collection and competitive with the dictionary translation. *s*-grams were formed with two different character combination indices: *s*-gram1's with CCI $\{\{0\},\{0,1\},\{1,2\}\}$ and *s*-gram2's with CCI $\{\{0\},\{1,2\}\}$. Only the better performing *s*-gram1's are discussed in the following. The fuzzy translation was compared to dictionary-based query translation and to monolingual Swedish IR. Also a Norwegian baseline query was formed to see how much the fuzzy translation improves the results compared to no translation at all.

A typical CLIR test setting with the search topics and test collection from Cross-Language Evaluation Forum (CLEF) 2003 was used. The CLEF'03 environment includes document collections and a set of 60 search topics in several languages, including Swedish (Peters, 2003). The Swedish document collection contains 142819 news wire articles from the Swedish news agency TT published in 1994–1995. As the Norwegian search topics were not included in the CLEF test environment, the English test topics were translated to Norwegian by a native Norwegian-speaker. The document collection and the search topics were not morphologically preprocessed for the fuzzy translation—the words were used in the inflected word forms in which they appeared in text. The collection and the topics were nevertheless normalized for the dictionary-based translation and monolingual IR, to ensure hard baselines. The stop words, as well as the duplicates of words appearing in identical word forms, were removed from the search topics. Finally, six of the search topics did not have any relevant documents in the Swedish document collection and were therefore removed from the topics. The tests were run with the remaining 54 topics.

Test queries were formed from the words of the title and description fields of the test topics (on average 7.5 words after duplicates were removed). The fuzzy translation was done by translating the Norwegian topics into Swedish by matching the *n*- or *s*-grams of the topic words against the Swedish document collection's index words' *n*- or *s*-grams. The three best matches were selected as translations for each word. For the dictionary translation, the GlobalDix dictionary by Kielikone Ltd. was used. All the translations for each topic word were selected to the query and the queries were structured according to the Pirkola method (Pirkola, 1998). The Norwegian and Swedish baseline queries were formed directly from the Norwegian and Swedish topics' words. The performance of the translation techniques was measured by interpolated mean average precision

[1] Fuzzy translation in CLIR between related languages has received some attention before: McNamee and Mayfield (2004) studied a fuzzy query translation technique based on comparing the source language query keys' *n*-grams directly to document collection's *n*-gram index, the results being promising.

over recall levels 10–100 averaged over all queries and as average precision at the recall level 10. A precision-recall graph was created using ten recall levels (10–100). The statistical significance of the results was tested using the Wilcoxon signed ranks test.

The results are presented in Table 2 and in Fig. 1. The statistical significance levels are given in the table. The *s*-grams achieved on average 80% of the dictionary baseline's performance and 62% of the monolingual Swedish baselines performance. The differences in the techniques' average precisions are statistically significant (Table 2). Still, 80% of dictionary baselines performance is a result that shows that *s*-gram translation is a promising and interesting query translation technique in CLIR between related languages. Especially so, when the language dependent dictionary translation setting requires two morphological analyzers (Norwegian and Swedish) and a translation dictionary to function, while the *s*-gram translation only needs a (language independent) program for producing the *s*-grams. When comparing the precision at the higher ranks, i.e., among the 20 and the 50 first retrieved documents (The document cut-off value (DCV) at 20 and 50 retrieved documents) and at the 10% recall level, the differences between the translation techniques are not statistically significant. These documents placed at the top of the result list are the most important ones from the practical user perspective.

The *s*-grams clearly outperformed the Norwegian baseline, with an average precision statistically significantly better than that of the Norwegian baseline (Table 2). The high percentual difference in performance suggests that the difference also has practical significance. It can be seen from Fig. 1 that the *s*-gram technique's precision-recall curve settles clearly above the *n*-gram technique's curve at recall levels 10–70 and always clearly above the Norwegian baseline (nobase). The difference in the *s*- and *n*-gram techniques' average precision, 16.3%, is not statistically significant. The fuzzy *s*-gram translation can be further improved by combining it to other fuzzy techniques (see e.g. Toivonen et al. (2005) for a description of an efficient fuzzy translation technique for words missing from dictionaries that combines *n*-grams to statistical rewriting rules). Therefore, it can be concluded that the *s*-gram translation is a promising technique in query translation between closely related languages.

Table 2
Interpolated average precision values (over recall levels 10–100) and the differences between the techniques (%)

Technique	Nobase	Swebase	Dicbase	*n*-gram	*s*-gram1
Average precision	13.38	35.6	28.4	19.6	22.8
Difference to nobase (%)			+112.3**	+46.5*	+70.4**
Difference to swebase (%)			−20.2	−44.9**	−36**
Difference to dicbase (%)				−31**	−19.7*
Difference to *n*-gram (%)					+16.3

Statistical significance levels are indicated in the table: * = statistically significant difference at level 0.01, ** = statistically highly significant difference at level 0.001.

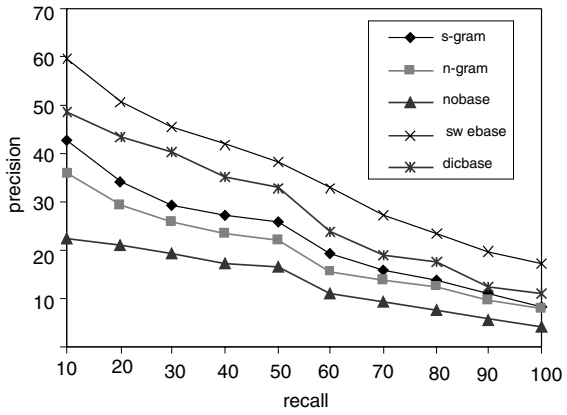

Fig. 1. Precision-recall graph over ten recall levels 10–100.

4. Formalizations of *n*-grams and their proximity functions

We begin by briefly reviewing the basic concepts of formal languages. We follow the conventions of Hopcroft, Motwani, and Ullman (2001). Then we define *n*-grams by following the conventions of Ukkonen (1992). Finally we give two proximity functions for strings based on their *n*-gram overlap. The first one is the L_1 metric originally defined for *n*-grams by Ukkonen (1992) and the second one is Jaccard's similarity function, which is very often used in IR experiments involving *n*-grams.

4.1. Basic concepts of formal languages

We define an *alphabet* Σ as a finite, non-empty set of symbols. A *string* is a finite sequence of symbols from given alphabet. For example, if we have an alphabet $\Sigma = \{a, b\}$, then a, b, ab and bba are strings over Σ. There is also the *empty string* ϵ which does not contain any symbols. Note that an empty string can be chosen from any alphabet. The *length* $|w|$ of a string w is the number of positions for symbols in the string. For instance $|abba| = 4$ and $|\epsilon| = 0$. A string $v = b_1 b_2 \ldots b_m$ is a *substring* of string $w = a_1 a_2 \ldots a_n$, $m \leqslant n$, if $b_1 b_2 \ldots b_m = a_i a_{i+1} \ldots a_{i+m-1}$ for some i, $1 \leqslant i \leqslant n$. If $w = abb$, then ϵ, a, ab, abb, bb and b are substrings of w.

If Σ is an alphabet we denote Σ^k a set of strings over Σ whose length is k. For example, if $\Sigma = \{a, b\}$, then $\Sigma^0 = \{\epsilon\}$, $\Sigma^1 = \{a, b\}$, $\Sigma^2 = \{aa, ab, ba, bb\}$ and $\Sigma^3 = \{aaa, aab, aba, baa, abb, bab, bba, bbb\}$. The set of all strings over an alphabet Σ is denoted Σ^*. In other words, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$.

For strings v and w, their *concatenation* vw is a string, which first has all the symbols of v in order of their appearance in v and then all the symbols in w in order of their appearance in w. Thus, the concatenation of strings $v = abba$ and $w = babba$ is $vw = abbababba$.

4.2. Definitions for n-grams

As we saw in previous sections, comparing two strings by calculating the overlap of their common substrings of certain length has a wide range of applications in IR. Now we are ready to give formal definitions for *n*-grams and their selected proximity functions.

Let Σ be a finite alphabet. An *n*-gram is any string $w \in \Sigma^n$. For example, if $\Sigma = \{a, b\}$ its di-grams are aa, ab, ba and bb. To be able to derive proximity functions between strings based on their *n*-gram overlap we need the following definition of string's *n*-gram profile (Ukkonen, 1992).

Definition 1. Let $w = a_1 a_2 \ldots a_m \in \Sigma^*$ and let $x \in \Sigma^n$ be a *n*-gram. If $a_i a_{i+1} \ldots a_{i+n-1} = x$ for some i, then w has an occurrence of x. Let $G(w)[x]$ denote the total number of occurrences of x in w. The *n*-gram profile of w is the vector $G_n(w) = (G(w)[x]), x \in \Sigma^n$.

Now, the distance of the strings can be defined as the L_1 norm (a.k.a Manhattan distance or city block distance) of the difference of their *n*-gram profiles (Ukkonen, 1992).

Definition 2. Let $v, w \in \Sigma^*$ and $n \in \mathbb{N}^+$. The *n*-gram distance between v and w is

$$D_n(v, w) = \sum_{x \in \Sigma^n} |G(v)[x] - G(w)[x]|. \tag{1}$$

Example 1. Let $\Sigma = \{a, b\}$ and $v = abba$, $w = babba \in \Sigma^*$. Their di-gram profiles, listed in lexicographical order of the di-grams, are $(0, 1, 1, 1)$ and $(0, 1, 2, 1)$. Thus, their di-gram distance $D_2(v, w) = 1$. Table 3 lists the di-grams of strings v and w and all di-grams over alphabet $\Sigma = \{a, b\}$.

The *n*-gram distance is pseudo metric (Ukkonen, 1992), i.e., for all $v, w, x \in \Sigma^*$:

(1) $D_n(v, w) \geqslant 0$ (non-negativity),
(2) $D_n(v, w) = D_n(w, v)$ (symmetry) and
(3) $D_n(v, w) \leqslant D_n(v, x) + D_n(x, w)$ (triangle inequality).

Table 3
The di-grams of strings $v = abba$, $w = babba$ over the alphabet $\Sigma = \{a,b\}^2$

	v	w	Σ^2
1	*ab*	*ba*	*aa*
2	*bb*	*ab*	*ab*
3	*ba*	*bb*	*ba*
4		*ba*	*bb*

Di-grams of v and w are listed in the order of their appearance in the strings and all occurrences of the di-grams are included in the list. Di-grams over Σ are listed in the lexicographical order. The di-gram profiles of v and w over the alphabet Σ are $G_2(v) = (0,1,1,1)$ and $G_2(w) = (0,1,2,1)$, when the di-grams are counted in lexicographical order.

It is not a metric since $D_n(v,w)$ can be 0 although $v \neq w$ (for example for di-grams of strings $v = abaa$ and $w = aaba$). Ukkonen (1992) lists also other properties of the n-gram distance which are here omitted.

Another approach to n-gram based string proximity is to calculate *similarity* between two strings instead of distance as in Eq. (1). Indeed, Keskustalo et al. (2003), Pirkola et al. (2002) and Toivonen et al. (2005) used Jaccard's formula based similarity function for s-grams. In order to formalize Jaccard's formula for s-gram similarity we first define it for ordinary n-grams—but based on n-gram profiles rather than n-gram sets.

In its basic form, Jaccard's formula for measuring the similarity between two sets A and B is written as

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{2}$$

where $|A|$ denotes the size of the set A. If $A = B$, then $A \cap B = A \cup B$, and thus $J(A,B) = 1$. On the other hand if A and B do not share common elements, then $A \cap B = \emptyset$ and $J(A,B) = 0$.

Since in set theory the duplicate occurrences of any element in the set are discarded, the Jaccard's formula based n-gram similarity function for strings ignores the multiple occurrences of n-grams. Therefore, instead of n-gram profile of Definition 1 we need a *binary n-gram profile*:

Definition 3. Let $w \in \Sigma^*$ and let $x \in \Sigma^n$ be a n-gram. Let $B(w)[x]$ denote the occurrences of string x in w as follows:

$$B(w)[x] = \begin{cases} 1 & \text{if } G(w)[x] > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The binary n-gram profile of w is the binary vector $B_n(w) = (B(w)[x])$, $x \in \Sigma^n$.

With the binary n-gram profile of w the Jaccard's formula based n-gram similarity function takes the following form.

Definition 4. Let v and w be non-empty strings from Σ^* and $n \in \mathbb{N}^+$. The Jaccard's n-gram similarity between v and w is

$$J_n(v,w) = \frac{\sum_{x \in \Sigma^n} B(v)[x]B(w)[x]}{\sum_{x \in \Sigma^n} (B(v)[x] + B(w)[x] - B(v)[x]B(w)[x])}. \tag{3}$$

Example 2. Let $\Sigma = \{a,b\}$ and $v = abba$, $w = babba \in \Sigma^*$. Now binary di-gram profiles of v and w, listed in lexicographical order of the di-grams, are $(0,1,1,1)$ and $(0,1,1,1)$. Thus their Jaccard's di-gram similarity $J_2(v,w) = 1$ and therefore the strings v and w are treated as equal (cf. Example 1).

As we mentioned in the previous section, it is common in IR applications to use padding spaces around the strings to get the symbols in the beginning and the end of the strings properly represented in the string comparison. It was also mentioned that it is common to use $n - 1$ padding spaces around the strings. Next we show how the padded n-gram comparison of the strings can be performed. For this purpose we assume a special padding symbol ♯. For example, in text retrieval applications ♯ could be thought as a regular space character.

Let Σ be a finite alphabet including the padding symbol $\sharp$, n an integer > 0 and $v, w \in (\Sigma \setminus \{\sharp\})^*$. Comparing strings v and w based on their n-gram overlap with, say, $n-1$ padding spaces in both the beginning and the end of the strings is performed as follows. Let p be a special string consisting only $n-1$ padding symbols, i.e., $p \in \{\sharp\}^{n-1}$ (clearly p is also in Σ^*). The padded comparison of the strings v and w is now trivial since operations $D_n(pvp, pwp)$ and $J_n(pvp, pwp)$ do the job. This follows from the fact that Σ^n contains (trivially) all n-grams of both v and w plus those n-grams which begin or end $n-1$ or less padding spaces (because $\sharp \in \Sigma$). If padding is used only in the beginning of the strings, the comparison is performed with $D_n(pv, pw)$ and $J_n(pv, pw)$. The situation where padding is used only in the end of the strings is handled correspondingly.

5. Formalizations of *s*-grams and their proximity functions

We shall now generalize the definition of n-grams by allowing skips between the symbols of the string w from which the n-gram is formed, i.e., the definition is generalized for s-grams. While Keskustalo et al. (2003) only consider s-grams of length of two, our definitions are for s-grams of any length. We also show how we get n-grams as special a case of our s-gram definition.

5.1. Definitions for s-grams

For simplicity we require that the skips in the s-grams are *systematical*, i.e., (1) each skip has equal length and (2) the skips are performed in each character position. With gram length of 2 and skip of 1 the $s_{2,1}$-grams of $w = abbababba$ are ab, ba, bb, aa, bb, ab, and ba.

Definition 5. Let $w = a_1 a_2 \ldots a_m \in \Sigma^*$, $n \in \mathbb{N}^+$ be a gram length, $k \in \mathbb{N}$ a skip length and let $x \in \Sigma^n$ be an n-gram. If $a_i a_{i+k+1} \ldots a_{i+(k+1)(n-1)} = x$ for some i, then w has a $s_{n,k}$-gram occurrence of x. Let $G_k(w)[x]$ denote the total number of $s_{n,k}$-gram occurrences of x in w. The $s_{n,k}$-gram profile of w is the vector $G_{n,k}(w) = (G_k(w)[x])$, $x \in \Sigma^n$.

Now, the $s_{n,k}$-gram based L_1 norm is defined as for n-grams:

Definition 6. Let v, w be strings from Σ^*, $n \in \mathbb{N}^+$ be a gram length and $k \in \mathbb{N}$ a skip length. The $s_{n,k}$-gram distance between v and w is

$$D_{n,k}(v, w) = \sum_{x \in \Sigma^n} |G_k(v)[x] - G_k(w)[x]|. \tag{4}$$

Example 3. Let $\Sigma = \{a, b\}$ be an alphabet and $v = aabab$, $w = babab$ strings from Σ^*. Their $s_{2,1}$-gram profiles, listed in lexicographical order of the $s_{2,1}$-grams, are $(1, 1, 0, 1)$ and $(1, 0, 0, 2)$. Thus, their $s_{2,1}$-gram distance $D_{2,1}(v, w) = 2$. Note that their di-gram distance defined in Eq. (1) would be 1. Table 4 lists the $s_{2,1}$-grams of strings v and w and all $s_{2,1}$-grams over alphabet $\Sigma = \{a, b\}$.

Note that Eq. (1) is a special case of Eq. (4), because $D_n(v, w) = D_{n,0}(v, w)$ for all strings $v, w \in \Sigma^*$. Furthermore, the following theorem shows that distance $D_{n,k}$ is a pseudo metric. The theorem is easy to prove and the proof is thus omitted.

Table 4
The $s_{2,1}$-grams of strings $v = aabab$ and $w = babab$ from the alphabet $\Sigma = \{a, b\}$

	v	w	Σ^2
1	*ab*	*bb*	*aa*
2	*aa*	*aa*	*ab*
3	*bb*	*bb*	*ba*
4			*bb*

$s_{2,1}$-grams of v and w are listed in the order of their appearance in the strings and all occurrences of the $s_{2,1}$-grams are included in the list. Di-grams over Σ are listed in the lexicographical order. The $s_{2,1}$-gram profiles of v and w over the alphabet Σ are $G_{2,1}(v) = (1, 1, 0, 1)$ and $G_{2,1}(w) = (1, 0, 0, 2)$, when the $s_{2,1}$-grams are counted in lexicographical order.

Theorem 1. *For all* $v, w, x \in \Sigma^*$,

(1) $D_{n,k}(v,w) \geqslant 0$ (*non-negativity*),
(2) $D_{n,k}(v,w) = D_{n,k}(w,v)$ (*symmetry*) *and*
(3) $D_{n,k}(v,w) \leqslant D_{n,k}(v,x) + D_{n,k}(x,w)$ (*triangle inequality*).

Thus also distance $D_{n,k}$ is pseudo metric. It is not metric, because $D_n(v,w)$ can be 0 although $v \neq w$ (for example for $s_{2,1}$-grams of strings $v = aaba$ and $w = aaab$).

Because, Keskustalo et al. (2003), Pirkola et al. (2002) and Toivonen et al. (2005) used Jaccard's similarity function for *s*-grams it is reasonable to formalize Jaccard's formula also for *s*-grams. This gives us also another point of view to *s*-gram based string proximity using the concept of similarity instead of distance.

The definition of Jaccard's similarity function for *s*-gram based string matching is analogous to that of *n*-grams. First, we need to define the *binary s-gram profile*:

Definition 7. Let $w \in \Sigma^*$, $n \in \mathbb{N}^+$ be a gram length and $k \in \mathbb{N}$ a skip length of the *s*-grams. Let $B_k(w)[x]$ denote the occurrences of string x in w as follows:

$$B_k(w)[x] = \begin{cases} 1 & \text{if } G_k(w)[x] > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The binary $s_{n,k}$-gram profile of w is the binary vector $B_{n,k}(w) = (B_k(w)[x])$, $x \in \Sigma^n$.

With the binary $s_{n,k}$-gram profiles of strings v and w, the *s*-gram similarity function based on Jaccard's formula takes the following form.

Definition 8. Let v and w be non-empty strings from Σ^*, $n \in \mathbb{N}^+$ a gram length and $k \in \mathbb{N}$ a skip length. The Jaccard's *s*-gram similarity between v and w is

$$J_{n,k}(v,w) = \frac{\sum_{x \in \Sigma^n} B_k(v)[x] B_k(w)[x]}{\sum_{x \in \Sigma^n} (B_k(v)[x] + B_k(w)[x] - B_k(v)[x] B_k(w)[x])}. \tag{5}$$

Example 4. Let $\Sigma = \{a, b\}$ be an alphabet and $v = aabab$, $w = babab$ strings from Σ^*. Their binary $s_{2,1}$-gram profiles, listed in lexicographical order of the $s_{2,1}$-grams, are $(1,1,0,1)$ and $(1,0,0,1)$. Thus, their Jaccard's $s_{2,1}$-gram similarity $J_{2,1}(v,w) = 2/3$ (cf. Example 3). Note that their Jaccard's di-gram distance defined in Eq. (3) would also be 2/3.

Note that Eq. (3) is a special case of Eq. (5), because $J_n(v,w) = J_{n,0}(v,w)$ for all strings $v, w \in \Sigma^*$.

As with *n*-grams, it is also common with *s*-grams to use padding spaces around the strings to get the symbols in the beginning and the end properly represented in string comparison. The approach illustrated in the previous section for *n*-grams also works with *s*-grams, and therefore it is not repeated here. However, it should be noted that with *s*-grams it is common to use $(k+1)(n-1)$ padding spaces around the strings instead of $n-1$ (i.e, the length of the padding string p is $(k+1)(n-1)$).

5.2. Skip-gram classes and their proximity functions

Pirkola et al. (2002) and Keskustalo et al. (2003) found that the *s*-gram matching performance is improved when the *s*-gram sets of two strings are produced in several different ways and classified so that the similarity in each skip-gram class is computed separately. The evidence from the different skip-gram classes is then combined for the comparison of the strings. Therefore, we will now begin the formulation skip-gram classes and their proximity functions.

Definition 9. The skip-gram class of skip lengths, or shortly skip-gram class, is a set $C \in \mathscr{P}(\mathbb{N})$. Character Combination Index (CCI) is a set of skip-gram classes, i.e., a set $\mathscr{C} \in \mathscr{P}(\mathscr{P}(\mathbb{N}))$. If we want to refer to *s*-grams of length n in certain skip-gram class C we will simply write $s_{n,C}$-gram.

Definition 10. Let $w \in \Sigma^*$, $C \in \mathscr{P}(\mathbb{N})$ a skip-gram class and $x \in \Sigma^n$. Let $G_C(w)[x] = \sum_{k \in C} G_k(w)[x]$. The skip-gram class profile of w is the vector $G_C(w) = (G_C(w)[x])$, $x \in \Sigma^n$. In other words, $G_{n,C}(w) = \sum_{k \in C} G_{n,k}(w)$.

The next definition gives the L_1 norm for skip-gram classes:

Definition 11. Let v, w be strings in Σ^*, $n \in \mathbb{N}^+$ be a gram length and $C \in \mathscr{P}(\mathbb{N})$ a skip-gram class. The skip-gram class distance between v and w is

$$D_{n,C}(v,w) = \sum_{x \in \Sigma^n} |G_C(v)[x] - G_C(w)[x]|. \tag{6}$$

Example 5. Let $\Sigma = \{a,b\}$ be an alphabet and $v = aabab$, $w = babab$ strings from Σ^*. Their $s_{2,\{0,1\}}$-skip-gram class profiles, listed in lexicographical order of the $s_{2,0}$-grams and $s_{2,1}$-grams, are $(2,3,1,1)$ and $(1,2,2,2)$. Thus, their $s_{2,\{0,1\}}$-skip-gram class distance $D_{2,\{0,1\}}(v,w) = 4$.

Note that we can calculate Eq. (4) with Eq. (6), because $D_{n,k}(v,w) = D_{n,\{k\}}(v,w)$ for all strings $v, w \in \Sigma^*$. Especially n-gram distance of Eq. (1) between strings v and w is given by $D_{n,\{0\}}(v,w)$.

According to the following theorem also skip-gram class distance is a pseudo metric. Again, the proof is obvious and thus omitted.

Theorem 2. *For all* $v, w, x \in \Sigma^*$,

(1) $D_{n,C}(v,w) \geqslant 0$ (*non-negativity*),
(2) $D_{n,C}(v,w) = D_{n,C}(w,v)$ (*symmetry*) *and*
(3) $D_{n,C}(v,w) \leqslant D_{n,C}(v,x) + D_{n,C}(x,w)$ (*triangle inequality*).

Next we want to derive the Jaccard's similarity function for gram class based string matching analogous to previous Jaccard's formulas. Again, this requires us to define a *binary skip-gram class profile*.

Definition 12. Let $w \in \Sigma^*$, and $C \in \mathscr{P}(\mathbb{N})$ a skip-gram class and $x \in \Sigma^n$. Let

$$B_C(w)[x] = \begin{cases} 1 & \text{if } G_C(w)[x] \geqslant 0, \\ 0 & \text{otherwise.} \end{cases}$$

The binary skip-gram class profile of w is the binary vector $B_{n,C}(w) = (B_C(w)[x])$, $x \in \Sigma^n$.

The Jaccard's formula based s-gram similarity using binary gram-class profiles of strings can now be defined like Jaccard's similarities earlier.

Definition 13. Let v and w be non-empty strings from Σ^*, $n \in \mathbb{N}^+$ be a gram length and $C \in \mathscr{P}(\mathbb{N})$ a skip-gram class. The Jaccard's skip-gram class similarity between v and w is

$$J_{n,C}(v,w) = \frac{\sum_{x \in \Sigma^n} B_C(v)[x] B_C(w)[x]}{\sum_{x \in \Sigma^n} (B_C(v)[x] + B_C(w)[x] - B_C(v)[x] B_C(w)[x])}. \tag{7}$$

Example 6. Let $\Sigma = \{a,b\}$ be an alphabet and $v = aabab$, $w = babab$ strings from Σ^*. Their binary $s_{2,\{0,1\}}$-skip-gram class profiles, listed in lexicographical order of the $s_{2,0}$-grams and $s_{2,1}$-grams, are $(1,1,1,1)$ and $(1,1,1,1)$. Thus, their $s_{2,\{0,1\}}$-skip-gram class similarity is $J_{2,\{0,1\}}(v,w) = 1$. Therefore, according to the Jaccard's skip-gram class similarity, strings v and w are equal. This is a notable difference to the L_1 norm between v and w calculated in Example 5.

Note again that we can calculate Eq. (5) with Eq. (7), because $J_{n,k}(v,w) = J_{n,\{k\}}(v,w)$ for all strings $v, w \in \Sigma^*$. Especially n-gram similarity of Eq. (3) between strings v and w is given by $J_{n,\{0\}}(v,w)$.

Finally, we define a proximity functions for strings based on their total s-gram overlap in given set of skip-gram classes specified by a CCI.

5.3. CCI-based string proximity

In Pirkola et al. (2002) and Keskustalo et al. (2003), the final similarity function for *s*-grams was based on the Character Combination Index (CCI) and an extension of Jaccard's formula. The idea was to calculate the Jaccard similarity in each skip-gram class and then to combine the similaries in a novel way to an overall similarity function. In this section, we shall continue our approach from preceeding sections by, firstly, defining a *distance* function for two strings based on their *s*-gram profiles and the given CCI, and secondly, binarizing these profiles and defining (precisely) the Jaccard style similarity function for the same situation. As we shall see though some examples, the two functions may give quite different results. This is due to the distance function counting each occurrence of any *s*-gram and the similarity function only counting one, because it is set oriented. Only the latter has so far been experimentally tested in the case of *s*-grams.

Definition 14. Let v and w be strings in Σ^* and $\mathscr{C} \in \mathscr{P}(\mathscr{P}(\mathbb{N}))$ a character combination index. The distance $D_{n,\mathscr{C}}(v,w)$ of v and w with regard to $\mathscr{C}$ is defined as the average distance of their skip-gram class distances, i.e.,

$$D_{n,\mathscr{C}}(v,w) = \frac{1}{|\mathscr{C}|}\sum_{C\in\mathscr{C}} D_{n,C}(v,w). \tag{8}$$

Example 7. Let $\Sigma = \{a,b\}$ be an alphabet, $v = abbababba$, $w = baabaaba$ strings from Σ^* and $\mathscr{C} = \{\{0,1\},\{2\}\}$ a CCI. To calculate the distance $D_{2,\mathscr{C}}(v,w)$, we first need to calculate the skip-gram class distances $D_{2,\{0,1\}}(v,w)$ and $D_{2,\{2\}}(v,w)$. The skip-gram class profiles listed in lexicographical order of the *s*-grams are for string v $G_{2,\{0,1\}}(v) = (1,5,5,4)$ and $G_{2,\{2\}}(v) = (2,1,1,2)$, and for string w $G_{2,\{0,1\}}(w) = (4,4,5,0)$ and $G_{2,\{2\}}(w) = (3,0,0,2)$. Thus,

$$D_{2,\mathscr{C}}(v,w) = \frac{1}{|\mathscr{C}|}\sum_{C\in\mathscr{C}} D_{2,C}(v,w) = \frac{D_{2,\{0,1\}}(v,w) + D_{2,\{2\}}(v,w)}{|\{\{0,1\},\{2\}\}|} = \frac{8+3}{2} = 5.5.$$

The advantage of using average distance in the above definition is that the CCI distance remains as a pseudo metric. However, the CCI distance $D_{n,\mathscr{C}}(v,w)$ has never been empirically tested. Instead, the following definition gives a proximity function loosely on based Jaccard's formula that is used by Pirkola et al. (2002) and Keskustalo et al. (2003) for skip-gram class based string comparison.

Definition 15. Let v and w be non-empty strings over an alphabet Σ, $n \in \mathbb{N}$ and $\mathscr{C} \in \mathscr{P}(\mathscr{P}(\mathbb{N}))$ a CCI. The similarity $S_{n,\mathscr{C}}(v,w)$ of v and w with regard to CCI $\mathscr{C}$ is

$$S_{n,\mathscr{C}}(v,w) = \frac{\sum_{C\in\mathscr{C}}\sum_{x\in\Sigma^n} B_C(v)[x]B_C(w)[x]}{\sum_{C\in\mathscr{C}}\sum_{x\in\Sigma^n}(B_C(v)[x] + B_C(w)[x] - B_C(v)[x]B_C(w)[x])}. \tag{9}$$

Example 8. Let $\Sigma = \{a,b\}$ be an alphabet, $v = abbababba$, $w = baabaaba$ strings from Σ^* and $\mathscr{C} = \{\{0,1\},\{2\}\}$ a CCI. To calculate the similarity $S_{2,\mathscr{C}}(v,w)$, we need to calculate the binary skip-gram class profiles. The profiles, listed in lexicographical order of the *s*-grams, for string v are $B_{2,\{0,1\}}(v) = (1,1,1,1)$, $B_{2,\{2\}}(v) = (1,1,1,1)$, and for string w are $B_{2,\{0,1\}}(w) = (1,1,1,0)$, $B_{2,\{2\}}(w) = (1,0,0,1)$ Thus,

$$S_{2,\mathscr{C}}(v,w) = \frac{\sum_{C\in\mathscr{C}}\sum_{x\in\Sigma^2} B_C(v)[x]B_C(w)[x]}{\sum_{C\in\mathscr{C}}\sum_{x\in\Sigma^2}(B_C(v)[x] + B_C(w)[x] - B_C(v)[x]B_C(w)[x])} = \frac{(1+1+1+0)+(1+0+0+1)}{(1+1+1+1)+(1+1+1+1)} = \frac{5}{8} = 0.625.$$

We gave the similarity function of Eq. (9) by convention, following Pirkola et al. (2002) and Keskustalo et al. (2003). However, this choice for similarity function for CCI based string matching is not the most intuitive one. Therefore, we propose a new, simpler similarity function calculated as an average of the skip-gram classes of the CCI, in the spirit of Eq. (8).

Definition 16. Let v and w be non-empty strings from Σ^* and $\mathscr{C} \in \mathscr{P}(\mathscr{P}(\mathbb{N}))$ a character combination index. The similarity $S'_{n,\mathscr{C}}(v,w)$ of v and w with regard to CCI $\mathscr{C}$ is

$$S'_{n,\mathscr{C}}(v,w) = \frac{1}{|\mathscr{C}|}\sum_{C\in\mathscr{C}} J_{n,C}(v,w). \tag{10}$$

The similarities given by the Eqs. (9) and (10) are not far apart but nevertheless different as the reader may find through the simple example constructed with strings *aabba* and *bbab* and a CCI $\mathscr{C} = \{\{0\},\{1\}\}$.

We end this section by noting that following equalities hold between the CCI based string proximity functions and s-gram proximity functions. Let v and w be strings over an alphabet Σ, $n \in \mathbb{N}^+$ a gram length and $k \in \mathbb{N}$ a skip length. Then

(1) $D_{n,k}(v,w) = D_{n,\{\{k\}\}}(v,w)$ and
(2) $J_{n,k}(v,w) = S_{n,\{\{k\}\}}(v,w)$,

where in (2) $S_{n,\{\{k\}\}}(v,w)$ denotes either Eq. (9) or Eq. (10). Thus proximities between n- and s-grams can be evaluated by using Eqs. (8)–(10) only.

6. Discussion and conclusions

n-grams have been used widely and successfully for approximate string matching in many areas. Recently, Pirkola et al. (2002) devised a novel classified s-gram matching technique, where di-grams are formed of both adjacent and non-adjacent characters. s-grams have proved successful in approximate string matching across language boundaries—especially in matching proper names and technical terminology. While n-grams have been precisely defined, s-grams so far lack such precise definitions. Also their similarity comparison in (Keskustalo et al., 2003) lacks a stringent definition. In this paper, we have given precise definitions both for s-grams and their distance/similarity comparison.

Following established practices in the literature (Ukkonen, 1992) we first presented the n-gram profiles of strings and then their *distance* measure, the L_1 metric. This is a well established distance measure, and takes into account both the kinds of n-grams two strings contain, and their number. Based on this we also defined binary n-gram profiles and the Jaccard *similarity* measure, which has been popular in IR experiments. As pointed out, this similarity measure is weaker than the L_1 distance metric because Jaccard is insensitive to the counts of each n-gram in the strings to be compared.

Turning to s-grams, we provided novel definitions of s-gram profiles and extended the L_1 distance metric for them. We gave definitions for simple s-gram profiles and distances and progressively extended them to skip-gram classes and collections of skip-gram classes (as specified by the CCI). The s-gram profiles were also reduced in each case to binary profiles in order to precisely define the (extended) Jaccard similarity functions for s-grams in each case. Again, as pointed out, the extended L_1 distance metrics may be claimed stronger than the corresponding Jaccard similarity measures because the Jaccard measures are insensitive to the counts of each n-gram in the strings to be compared. Interestingly, the L_1 distance metrics have never been employed in IR experiments based on s-grams. Their greater strength in measuring string proximity suggests their potential for IR experiments, which are planned for in our forthcoming research.

Pirkola et al. (2002) only considered s-di-grams, which entails one skipping possibility between the two characters. Our definitions were for general s-grams with multiple skipping positions and constant skip length. Unfortunately, at the sub-word character string level, we do not know of text retrieval applications of $s_{n,k}$-grams for $n > 2$.

Finally, we showed that n-gram similarity/distance computations are special cases of our generalized definitions. In fact, all s-gram and n-gram distance computations may be carried out by Eq. (8) and their Jaccard similarities by Eq. (9) or Eq. (10).

The size of the n-gram index has been pointed out as a critical factor in their application (Califano & Rigoutsos, 1993; McNamee & Mayfield, 2004), where the determining factors are the size of the symbol set, and n-gram length, which determine the length of the profile vectors per object (i.e. word, text, biological sequence). In our case, a further complication arises from the skip-gram classes as each class requires its own profile. However, in the present problem area, focussing on s-grams for mono-lingual and cross-language word matching, the symbol set Σ typically has less than 30 symbols, the gram length typically is 2 and the CCI $\mathscr{C}$ contains 2–3 classes. With such values the s-gram profile size per keyword ($= |\mathscr{C}||\Sigma^n|$) remains under 2 KB per keyword which is very reasonable. Recall that the keyword index takes, for each keyword, the keyword length and the address list length, which is governed by address specificity, address length, and the database size (word count).

At a more general level, our strong belief is that popular proximity measures need precise definitions. This fosters an understanding of their relative strengths (e.g. the L_1 distance metric being stronger than the Jaccard measure) and their consistent implementation and application. Our definitions were developed in a bottom-up manner, only assuming character strings (based on some alphabet) and elementary mathematical concepts.

Acknowledgements

This study was funded in part by Department of Information Studies, University of Tampere, TISE (Tampere Graduate School for Information Science and Engineering), and Academy of Finland under the Grant # 1209960.

References

Altschul, S., Gish, W., Miller, W., Myers, E., & Lipman, D. (1990). A basic local alignment search tool. *Journal of Molecular Biology, 215*, 403–410.

Baròdal, J., Jörgensen, N., Larsen, G., & Martinussen, B. (1997). Nordiska: Våra språk förr och nu. Studentlitteratur.

Bishop, M., & Thompson, E. (1984). Fast computer search for similar DNA sequences. *Nucleic Acid Research, 12*(13), 5471–5474.

Califano, A., & Rigoutsos, I. (1993). FLASH: A fast look-up algorithm for string homology. In *Proceedings of the 1st international conference on intelligent systems for molecular biology* (pp. 56–64). AAAI Press.

Chen, A., He, J., Xu, L., Gey, F., & Meggs, J. (1997). Chinese text retrieval without using a dictionary. In: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 42–49.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM, 7*(3), 171–176.

Downie, S., & Nelson, M. (2000). Evaluation of a simple and effective music information retrieval method. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 73–80). New York, NY USA: ACM Press.

Grossman, D. A., & Frieder, O. (2004). *Information retrieval: Algorithms and heuristics* (2nd ed.). Springer.

Hall, P. A. V., & Dowling, G. R. (1980). Approximate string matching. *ACM Computing Surveys, 12*(4), 381–402.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). *Introduction to automata theory languages and computation* (2nd ed.). Addison Wesley.

Keskustalo, H., Pirkola, A., Visala, K., Leppänen, E., & Järvelin, K. (2003). Non-adjacent digrams improve matching of cross-lingual spelling variants. In M. A. Nascimento, E. de Moura, & A. Oliveira (Eds.), *Proceedings of the 10th international symposium on string processing and information retrieval SPIRE. Lecture Notes in Computer Science* (2857, pp. 252–265). Berlin: Springer.

McNamee, P., & Mayfield, J. (2003). JHU/APL experiments in tokenization and non-words translation. In: CLEF 2003 working notes. Available from http://clef.iei.pi.cnr.it/.

McNamee, P., & Mayfield, J. (2004). Character n-gram tokenization for European language text retrieval. *Information Retrieval, 7*, 73–97.

Miller, C., Gurd, J., & Brass, A. (1999). A rapid algorithm for sequence database comparisons: application to the identification of vector contamination in the EMBL databases. *Bioinformatics, 15*(2), 111–121.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys, 33*(1), 31–88.

O'Rourke, A., Robertson, A., & Willettt, P. (1997). Word variant identification in old french. *Information Research, 2*(4).

Peters, C. (2003). Introduction to the CLEF 2003 working notes. Available from http://clef.iei.pi.cnr.it/.

Pfeiffer, U., Poersch, T., & Fuhr, N. (1996). Retrieval effectiveness of proper name search methods. *Information Processing and Management, 32*(6), 667–679.

Pirkola, A. (1998). The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 55–63). New York, NY, USA: ACM Press.

Pirkola, A., Keskustalo, H., Leppänen, E., Känsälä, A. P., & Järvelin, K. (2002). Targeted *s*-gram matching: a novel *n*-gram maching technique for cross- and monolingual word form variants. *Information Research, 7*(2). Available from http://InformationR.net/ir/7-2/paper126.html.

Robertson, A. M., & Willett, P. (1998). Applications of *n*-grams in textual information systems. *Journal of Documentation, 54*(1), 48–69.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal, 27*(8), 379–423.

Toivonen, J., Pirkola, A., Keskustalo, H., Visala, K., & Järvelin, K. (2005). Translating cross-lingual spelling variants using transformation rules. *Information Processing and Management, 41*, 859–872.

Uitdenbogerd, A. L., & Zobel, J. (2002). Music ranking techniques evaluated. In *ACSC '02 Proceedings of the twenty-fifth Australasian conference on computer science* (pp. 275–283). Darlinghurst, Australia: Australian Computer Society, Inc.

Ukkonen, E. (1992). Approximate string-matching with *q*-grams and maximal matches. *Theoretical Computer Science, 92*(1), 191–211.

Ullman, J. R. (1977). A binary *n*-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. *The Computer Journal, 20*(2), 141–147.

Zamora, E. M., Pollock, J. J., & Zamora, A. (1981). The use of trigram analysis for spelling error detection. *Information Processing and Management, 17*(8), 305–316.

Zobel, J., & Dart, P. (1996). Phonetic string matching: lessons from information retrieval. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 166–172). New York, NY, USA: ACM Press.