Expert Systems With Applications 129 (2019) 169–185

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Framework for syntactic string similarity measures ☆

Najlah Gali*, Radu Mariescu-Istodor, Damien Hostettler, Pasi Fränti

Machine Learning Group, School of Computing, University of Eastern Finland, Joensuu FI-80101, Finland

ARTICLE INFO

Article history:
Received 17 September 2018
Revised 4 March 2019
Accepted 27 March 2019
Available online 2 April 2019

Keywords:
Similarity measure
String similarity
Information retrieval
Text processing

ABSTRACT

Similarity measure is an essential component of information retrieval, document clustering, text summarization, and question answering, among others. In this paper, we introduce a general framework of syntactic similarity measures for matching short text. We thoroughly analyze the measures by dividing them into three components: character-level similarity, string segmentation, and matching technique. Soft variants of the measures are also introduced. With the help of two existing toolkits (SecondString and SimMetric), we provide an open-source Java toolkit of the proposed framework, which integrates the individual components together so that completely new combinations can be created. Experimental results reveal that the performance of the similarity measures depends on the type of the dataset. For well-maintained dataset, using a token-level measure is important but the basic (crisp) variant is usually enough. For uncontrolled dataset where typing errors are expected, the soft variants of the token-level measures are necessary. Among all tested measures, a soft token-level measure that combines set matching and q-grams at the character level perform best. A gap between human perception and syntactic measures still remains due to lacking semantic analysis.

1. Introduction

Similarity measures are needed in several fields, including biomedical, signal processing, natural language processing, statistics, artificial intelligence, and information retrieval. For example, linking records requires a similarity measure to locate matches across pairs of lists not having unique identifiers (Agbehadji et al., 2018; Song, Batjargal, & Maeda, 2019). In information retrieval, a measure is needed to retrieve documents relevant to a user's query. A similarity measure is also needed for resolving range violations in a large knowledge graph extracted from structured data such as Wikipedia (Lertvittayakumjorn, Kertkeidkachorn, & Ichise, 2017) and for auto-correcting text where a misspelled word is replaced with a dictionary word with high similarity. Measuring similarity of text has further been used in detecting plagiarism. Since plagiarism usually happens in some parts of text, the text should be segmented into smaller fragments before measuring the similarity (Ehsan & Shakery, 2016). Modern techniques use *skip-grams* and *Word2Vec* for cross-language plagiarism detection as well as *q-grams* (Barrón-Cedeño, Gupta, & Rosso, 2013; Franco-Salvador, Rosso, & Montes-y-Gómez, 2016). Other types of data for which a similarity measure is needed include:

- Titles of web pages (Gali, Mariescu-Istodor, & Fränti, 2016), *V-café – Viet-Café*;
- Keywords and keyphrases (Rezaei & Fränti, 2014), *theater – theatre*;
- Named entities (Cohen, Ravikumar, & Fienberg, 2003b; Moreau, Yvon, & Cappé, 2008), *U.S State Department – US Department of State*;
- Personal names (Christen, 2006; Snae, 2007), *Gail Vest – Gayle Vesty*;
- Place names (Recchia & Louwerse, 2013), *Ting Tsi River – Tingtze River*;
- Ontology alignments (Cheatham & Hitzler, 2013; Sun, Ma, & Wang, 2015), *associate professor – senior lecturer*;
- Short segments of text (Metzler, Dumais, & Meek, 2007), *Apple computer – Apple pie*;
- Sentences (Achananuparp, Hu, & Shen, 2008), *I haven't watched television for ages – It's been a long time since I watched television*.
- Trace links in software development (Alenazi, Reddy, & Niu, 2018).

☆ A preliminary version of the paper was presented in ICPR conference (Gali et al., 2016). It covered twenty-four existing measures for matching title phrases. The new version thoroughly analyzes 143 similarity measures, of which 91 are novel. Additional data sets were analyzed, more experiments were conducted, and new conclusions were drawn. An open-source Java toolkit with adaptable components is provided to implement all the measures reviewed in this paper and is made publicly available on http://cs.uef.fi/sipu/soft/stringsim/.

* Corresponding author.
E-mail addresses: najlaa@cs.uef.fi (N. Gali), radum@cs.uef.fi (R. Mariescu-Istodor), damien@cs.uef.fi (D. Hostettler), franti@cs.uef.fi (P. Fränti).

https://doi.org/10.1016/j.eswa.2019.03.048
0957-4174/

Given two strings, the goal is to determine their similarity. Similarity can be semantic or syntactic. Strings are *semantically similar* if they have the same meaning such as *car* and *automobile*, and *syntactically similar* if they have the same character sequence. Existing semantic similarity measures can be classified into *corpus-based, knowledge-based*, or a combination of the two (Mihalcea, Corley, & Strapparava, 2006).

Corpus-based measures, such as *latent semantic analysis* (LSA) (Landauer & Dumais, 1997), *pairwise mutual information* (PMI) (Turney, 2001), and *Word2Vec* (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013), measure the similarity between two strings depending on information gained from large corpora. For example, LSA assumes that words of similar meaning will occur in related pieces of text. A word-paragraph matrix is created where each value represents how many times the given word appears in that paragraph. *Singular value decomposition* has been used to find reduced dimensional representation of the matrix so that only important words are retained. Word similarity is computed by taking the cosine of the angle between any two vectors (rows) corresponding to the words being compared.

Knowledge-based measures use semantic networks such as *WordNet* (Miller, 1995). For example, Wu and Palmer (1994) measure the similarity between two words using the depth of their *least common subsumer* (LSC) and the *word depth*, where *depth* is the number of links between the word and the root word in WordNet (see Fig. 1). While useful, semantic similarity depends on language, taxonomy, and corpora. It may also provide poor results. For example, the similarity between *staff* and *body* is 0.76 although the two words are far from being similar (Wu & Palmer, 1994).

Syntactic measures operate on the words and their characters without any assumption of the language or the meaning of the content. They are, therefore, more general than semantic measures. Usually, syntactic measures output a distance, which indicates how dissimilar two data elements are. The larger the distance between the two elements, the less similar they are. Distance and similarity can be used interchangeably, as they are inverse functions. In this paper, all distance measures have been converted to similarity measures, which return a score in a [0, 1] interval where 0 means *nothing in common* and 1 means *exact match*.

Existing measures have usually been created for a specific task or application in mind. Most researchers are not interested in the details of the similarity measures; they just need one single measure for their application to provide estimates of the similarities of two strings. For this reason, the existing measures have been widely adopted in other applications areas than they were originally designed for. Many ad hoc measures are also used to fulfill the need. It is, therefore, an open question which measure should be used in a certain type of application. If there is a measure that is universal, it should fit all conceivable applications, not only a wide set.

Previous comparative studies focus mainly on specific tasks such as names of people, places, institutions or companies. They point out that the performance of the similarity measures is affected by text length, spelling accuracy, abbreviations and the language. Another common observation is that measures that have good performance on one data type can perform poorly on another. As a result, there is no measure consistently outperforming the others in all tasks (Christen, 2006; Snae, 2007). For example, Levenshtein distance (Levenshtein, 1966) and the Jaro metric (Jaro, 1989) work well for matching names but perform poorly for matching acronyms such as *Western Canadian Place fitness* and *WCP fitness*. Despite them being used in many tasks, extensive review and evaluation of syntactic similarity measures are missing in the literature.

Syntactic similarity measures can be classified into two broad classes: character-level and token-level (Gali et al., 2016). *Character-level measures,* such as *Levenshtein distance* (Levenshtein, 1966), treat the strings as sequences of characters. This type of measure is useful when the strings are single words involving only misspellings, typographical errors or slight morphological variations. However, strings can be split into tokens by whitespace and punctuation marks. In this case, it can be more useful to analyze the strings as a sequence or a set of tokens instead of just characters; such measures are called *token-level measures*. They are more suitable when less significant tokens are missing from one string or when they are in different order. The biggest deficiency of most token-level measures is that they only compare whether the tokens are exactly the same or not.

Only a few methods combine character- and token-level measures (Cohen et al., 2003b; Sidorov, Gelbukh, Gómez-Adorno, & Pinto, 2014; Vargas, 2008). These methods are called *soft measures*. The principle of a soft measure is to apply a character-level measure to all pairs of tokes between the strings and consider only tokens that satisfy a certain criterion (e.g. threshold) as input to a token-level measure. According to Jimenez, Gonzalez, and Gelbukh (2010), soft cosine, outperforms both character- and token-level measures for name matching. *Soft-cosine* combines the cosine for token matching and bigrams for character-level matching.

However, when applied as standalone measures, neither cosine nor bigrams is the best choice (Jimenez et al., 2010). It is expected that a better combination can be found from other character- and token-level measures by analyzing them extensively.

In this paper, we introduce a novel framework for a generic similarity measure. We provide a review of the existing syntactic similarity measures and show how they fit into this framework. We perform a comparative study of 143 similarity measures, in total, of which 52 exist in the literature and 91 are new

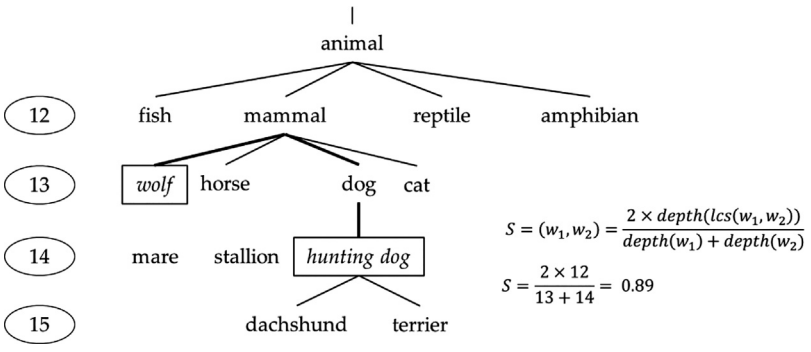

Fig. 1. Semantic similarity between wolf and hunting dog using WordNet taxonomy, with mammal being the least common subsumer; the circled numbers represent the depth.

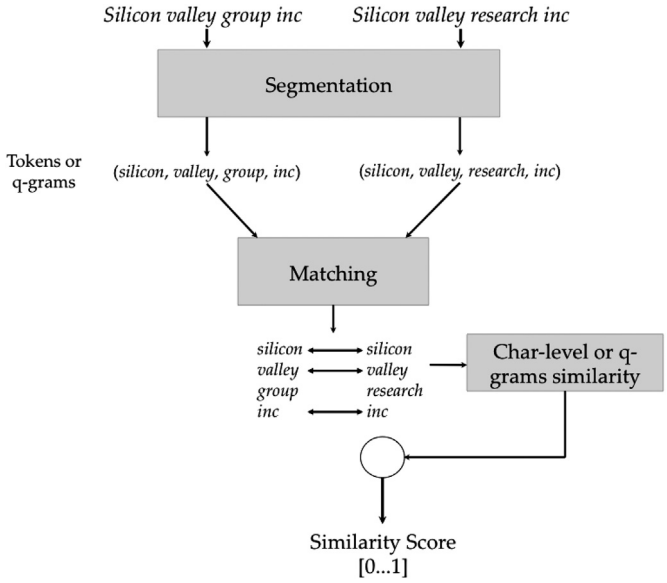

Fig. 2. String similarity workflow.

combinations formed from the existing components. We study the design choices for each of the building blocks including the character-level measure, segmentation strategy, and the token-level matching. We aim at answering the questions, which combination works best in a certain type of applications, and whether a single multi-purpose measure could be identified that would fit, if not perfectly, but sufficiently well for most applications. For the sake of completeness, we compare the obtained results with the semantic similarity measure *Word2Vec*.

To begin with, we divide the procedure of measuring the similarity into three components (see Fig. 2):

- Character-level measure,
- String segmentation,
- Matching technique.

String similarity can be computed directly by considering the entire string as one token using character-level or q-gram measures; a q-gram is a substring of length q of some string s. An alternative approach is to segment the string with language components such as words or character groups taken into account. We performed a systematic study of these components and found several new measures combining the best properties of various character- and token-level measures.

We also provide an open-source Java toolkit, StringSim that contains all the measures reviewed in this paper and allows combining different measures to produce their soft variants. To the best of our knowledge, there is no other public toolkit that goes beyond traditional measures by providing adaptable components and supports soft variants of the existing measures. Some combinations might not have a known application as of now; however, new applications appear every day. Therefore, investigating the different combinations for novel and non-trivial applications is worthwhile. This package also allows extensive cross-comparison of all known combinations. Several new meaningful measures are also introduced, of which q-grams and set-matching methods perform well in all text types used. Some combinations can also be found using the existing packages listed in Table 1.

2. Character-level measures

A string can be viewed as a unit composed of a sequence of characters. Existing character-level measures can be categorized into the following three classes:

- Exact match,
- Transformation,
- Longest common substring (LCS).

Exact match provides a simple binary result: 1 = *the strings are exactly the same*, 0 = *otherwise*. This is the classical way of comparing strings in information retrieval but is slowly being replaced by approximate matching. Nevertheless, most token-level measures still use this naive approach.

Transformation measures quantify the similarity of two strings by counting the number of operations needed to turn one string into the other. It can be achieved in several ways. Most common is *edit distance*, which measures the minimum number of edit operations needed to transform a string s_1 to string s_2. The edit operations include insertion, deletion, and substitution. The best match can be found by dynamic programming in $O(|s_1|\times|s_2|)$ time using $O(\min(|s_1|,|s_2|))$ space, where $|s_1|$ and $|s_2|$ are the lengths of the strings s_1 and s_2 to be compared in characters (Jimenez, Becerra, Gelbukh, & Gonzalez, 2009). Variations of edit distance have been proposed (depending on the number, type, and cost of operations), including *Levenshtein* (Levenshtein, 1966), *Damerau–Levenshtein* (Damerau, 1964), *Needleman–Wunsch* (Needleman & Wunsch, 1970), *Smith–Waterman* (Smith & Waterman, 1981), and *Smith–Waterman–Gotoh* (Gotoh, 1982) (see Table 2).

Levenshtein allows insertion, deletion, and substitution at a cost of one unit. *Damerau–Levenshtein* allows swapping of two adjacent characters ($ab \leftrightarrow ba$) at a cost of one unit. *Needleman–Wunsch* was originally developed in the area of bioinformatics to align protein or nucleotide sequence. It uses a cost of two units for insertion and deletion, and one for substitution. These types of edit distances are suitable for matching strings with typographical errors (*king sitric* and *kingsitric*), but not for other types of mismatch such as truncated or shortened strings (*Southville Running Club* and *Southville RC*). *Smith–Waterman* and *Smith–Waterman–Gotoh* offer solutions to this problem.

Table 1
Existing similarity measure packages.

Year	Package	Language	Type	No. of measures	Source
2003	SecondString[a]	Java	Character, Token, Soft	38	Cohen et al. (2003)
2005	SimMetric[b]	Java	Character, Q-gram, Token	23	—
2013	DKPro[c]	Java	Character, Q-gram, Token, Soft	20	Bär et al. (2013)
2014	Stringdist[d]	C	Character, Q-gram	10	Van der Loo (2014)
2016	Harry[e]	C	Character, Token	21	Rieck and Wressnegger (2016)
2017	StringSim[f]	Java	Character, Q-gram, Token, Soft	143	This paper

[a] https://sourceforge.net/projects/secondstring.
[b] https://sourceforge.net/projects/simmetrics.
[c] https://dkpro.github.io/dkpro-similarity.
[d] http://www.markvanderloo.eu/yaRb/category/string-metrics.
[e] http://www.mlsec.org/harry.
[f] http://cs.uef.fi/sipu/soft/stringsim.

Table 2
Character-level measures. *Edit* is the cost of operations according to the particular corresponding measure. The symbols s_1 and s_2 refer to the input strings. In Jaro, m is the number of matching characters and x is the number of transposed characters divided by 2. Symbol p in Jaro–Winkler is a scaling factor of 0.1, and l is the length of the common prefix up to four characters between the strings.

Similarity measure	Equation	Edit operation costs			
		Insert	Delete	Substitute	Swap
Levenshtein (1966)	$1-\frac{edit(s_1,s_2)}{max(\lvert s_1\rvert,\lvert s_2\rvert)}$	1	1	1	–
Damerau–Levenshtein (Damerau, 1964)	$1-\frac{edit(s_1,s_2)}{max(\lvert s_1\rvert,\lvert s_2\rvert)}$	1	1	1	1
Needleman and Wunsch (1970)	$1-\frac{edit(s_1,s_2)}{2\times max(\lvert s_1\rvert,\lvert s_2\rvert)}$	Variable	Variable	1	–
Smith and Waterman (1981)	$\frac{edit(s_1,s_2)}{min(\lvert s_1\rvert,\lvert s_2\rvert)}$	Variable	Variable	−2	–
Smith–Waterman–Gotoh (Gotoh, 1982)	$\frac{edit(s_1,s_2)}{min(\lvert s_1\rvert,\lvert s_2\rvert)}$	Variable	Variable	−3 + 3	–
Hamming (1950)	$1-\frac{edit(s_1,s_2)}{max(\lvert s_1\rvert,\lvert s_2\rvert)}$	–	–	1	–
Jaro (1989)	$\frac{1}{3}\times\left(\frac{m}{\lvert s_1\rvert}+\frac{m}{\lvert s_2\rvert}+\frac{m-x}{m}\right)$	–	–	–	–
Jaro–Winkler (Winkler, 1990)	$J(s_1,s_2)+(l\times p(1-J(s_1,s_2))$	–	–	–	–
Longest common substring (Friedman & Sideli, 1992)	$\frac{\lvert s\hat{u}b(s_1,s_2)\rvert}{max(\lvert s_1\rvert,\lvert s_2\rvert)}$	–	–	–	–

Smith–Waterman performs local alignment by finding similar regions in the two strings. It assigns a lower cost when the mismatch happens at the beginning or at the end of the strings than when it happens in the middle ([Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003; Elmagarmid, Ipeirotis, & Verykios, 2007). For example, the measure provides higher similarity value for strings such as *Prof. Mohammed A. Gali, University of Baghdad* and *Mohammed A. Gali, Prof.* than do Levenshtein or Needleman–Wunsch. The result is obtained in $O(\min(|s_1|,|s_2|) \times |s_1| \times |s_2|)$ using $O(|s_1| \times |s_2|)$ space (Christen, 2006).

Smith–Waterman–Gotoh (SWG) improves the scaling of Smith–Waterman by adding a so-called *affine gap cost* allowing better local alignment of the strings. It introduces two costs for insertion: *gap open* (a penalty of unmatched characters in the beginning of a string) and *gap extension* (a penalty for its continuation). In addition, substitution by a similar-sounding character ({d, t}, {g, j}) is given a higher score than by other mismatch characters. For example, a cost of +5 is assigned to matching characters, +3 to similar-sounding, and −3 to a mismatch. SWG requires $O(|s_1| \times |s_2|)$ in time and space.

Other examples of transformation measures are *Hamming* (Hamming, 1950), *Jaro* (Jaro, 1989), and *Jaro–Winkler* (Winkler, 1990). *Hamming* allows only substitutions, and the length of the strings must be equal. *Jaro* was originally developed for linking records having inaccurate text fields. It calculates the number of matching and transposed characters. Characters are matched if they are the same and located no farther than $[\max(|s_1|,|s_2|)/2] - 1$ within the string, and transposed if they are the same but in reverse order (a-u, u-a). For example, in comparing *CRATE* with *TRACE*, only '*R*' '*A*' '*E*' are the matching characters. Although '*C*' and '*T*' appear in both strings, they are farther than 1 unit (the result of $[5/2] - 1$). *Jaro–Winkler* modifies Jaro to provide higher weight to prefix matches. Winkler (1990) observed that typing errors usually occur in the middle or at the end of the string, but rarely at the beginning. As a result, Winkler adds a prefix weight ($l \times p\ (1 - d_j)$) which returns higher similarity scores when the strings match from the beginning, where l is the length of the common prefix up to four characters, p is a scaling factor of 0.1, and d_j is the Jaro similarity (see Table 2).

The LCS measure (Friedman & Sideli, 1992) was designed for applications such as matching patient records in a clinical setting and text summarization, but it can also be applied for comparing short text. Therefore, we study LCS as well. It finds the longest contiguous sequence of characters that co-occur in the two strings. The result is normalized by dividing the length of this sequence by the length of the longer string.

To sum up, character-level measures are useful for matching strings that contain only a few typographical errors but not for detecting the ordering of entire tokens. For example, they fail to capture the similarity between *Café Manta* and *Manta café*.

3. String segmentation

Segmentation divides the strings into units such as q-grams or words. It utilizes information at a higher level than characters alone. Two approaches exist to segment the string:

- Q-grams,
- Tokenization.

The *q-grams approach* (Shannon, 1948) divides a string into substrings of length q. The q-grams were first used for string matching in Ukkonen (1992). The segmentation is overlapping, as the same character belongs to several q-grams (except when $q = 1$). Substrings of length 2 are called *bigrams (or 2-grams)* and length 3 *trigrams (or 3-grams)*. The rationale behind q-grams is that the sequence of characters is more important than the characters alone. The q-grams for a string s are obtained by sliding a window of length q over the string (see Table 3). To consider also substrings of length $q - 1$ and to recognize prefixes and suffixes of the string, so-called *padding characters* (#% $) are appended to the beginning and end of the string. The similarity is calculated as follows:

$$QGramsSim(s_1, s_2) = 1 - \frac{\sum_{i=1}^{n} |match(q_i, Q_{s_1}) - match(q_i, Q_{s_2})|}{|Q_{s_1}| + |Q_{s_2}|} \quad (1)$$

where Q_{s1} and Q_{s2} are the multi-sets of q-grams from s_1 and s_2, respectively, $n = |Q_{s1} \cup Q_{s2}|$, and *match* (q_i,Q_{s1}) is the number of times the q-gram q_i appears in Q_{s1}. In this paper, we use q-grams with paddings to consider tokens having fewer than q characters, such as the determiners *a* and *an*.

Variants of q-grams are positional q-grams and skip-grams. *Positional q-grams* (Christen, 2006) preserve the position of the grams in the string and match only q-grams with a distance of less than a predefined threshold. For example, *club* contains the positional bigrams (*cl*, 0), (*lu*, 1), (*ub*, 2). If the threshold is set to 1, then the bigram (*lu*, 1) will only match to bigrams in the second string in positions 0, 1 or 2. *Skip-grams* (Keskustalo, Pirkola, Visala, Leppänen, & Järvelin, 2003) are bigrams that skip one or more character in the middle (see Table 3).

Tokenization breaks a string into units called *tokens* using whitespaces and punctuation characters (see Table 3). The rationale behind tokenization is to utilize information at the token level

Table 3
The segmentation of string **The club at the Ivy**, symbol _ refers to space.

Segmentation method	Output
None (character sequence)	the club at the ivy
q-grams ($q=3$)	the, he_, e_c, _cl, clu, lub, ub_, b_a, _at, at_, t_t, _th, the, he_, e_i, _iv, ivy
q-grams with padding characters	##t, #th, the, he_, e_c, _cl, clu, lub, ub_, b_a, _at, at_, t_t, _th, the, he_, e_i, _iv, ivy, vy%, y%%
1-skip-grams	t*e, h*c, e*l, c*u, l*b, u*a, b*t, a*t, t*h, t*e, h*i, e*v, i*y
Tokenization	the, club, at, the, ivy

and to overcome problems of token swap and missing tokens. In Christen (2006), two solutions to solve the token ordering problem were introduced: *sorting heuristic* and *permuting heuristic*. In sorting heuristic, each string is tokenized, tokens alphabetically ordered, re-joined again, and then edit distance is applied to the modified strings. In permuting heuristic, all token permutations are obtained from the first string and a comparison between all the permuted strings and the second string is then performed; the highest similarity value is chosen. However, these heuristic solutions are inefficient for other types of mismatching such as missing tokens, especially when the length of the absent token is considerable such as *Rosso* and *Rosso restaurant*. A better solution is therefore needed in such cases.

4. Matching techniques

Methods for token matching involve two challenges: which tokens to match and how to compute the similarity between the matched tokens. These challenges will be discussed next in greater detail. It should be noted that q-grams could also be used as the matching units even though we use the term token for simplicity. Matching depends on how the strings are represented. Three possibilities exist (see Fig. 3):

- Sequence,
- Set,
- Bag-of-tokens.

4.1. Sequence

The idea of *sequence* matching is to generalize Levenshtein or some other character-level measure to the token level. Instead of characters, tokens are used as comparative units. The cost of the edit operations insertion, deletion, and substitution is a function

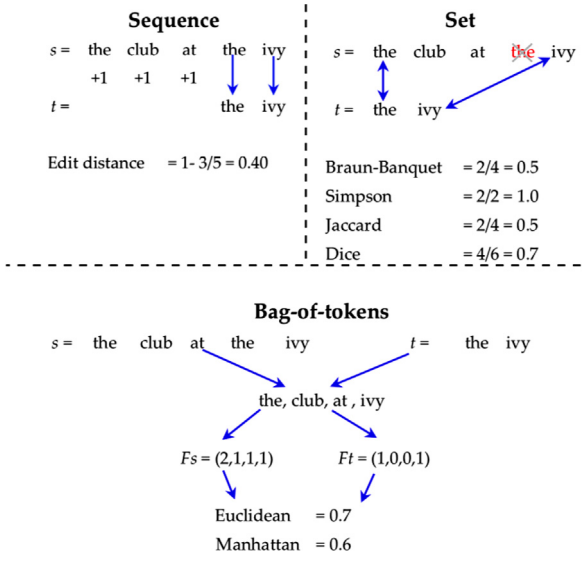

Fig. 3. Examples of exact string matching at token level.

of the tokens being compared. In Chaudhuri, Ganjam, Ganti, and Motwani (2003), the cost of substitution is calculated as the Levenshtein distance of the two tokens weighted by the inverse document frequency calculated from the two strings. Some character-level measures such as Smith–Waterman and SWG cannot be used at the token level because they are based on properties of characters such as their similar sounds.

4.2. Set

The idea behind *set* matching is to make the matching independent of the order. Sets, which are a collection of non-repeating tokens, are first generated from the input strings. Any set-matching method then can be applied to measure the overlap between the sets. Most of them calculate the intersection and differ only on how they are normalized. *Braun-Banquet* (Choi, Cha, & Tappert, 2010), *Simpson coefficient* (Choi et al., 2010), *Jaccard index* (Rezaei & Fränti, 2016), and *Dice coefficient* (Brew & McKelvie, 1996) divide the cardinality of the intersection by the cardinality of the largest set, the smallest set, the cardinality of the union of the two sets, or the average cardinality of the two sets; where cardinality is the number of tokens in the set (see Table 7 on page 11). *Multi-set* allows the same token to appear multiple times as in *Rouge-N* (Lin, 2004). It computes the similarity by using the *F*-score, which combines the precision and the recall.

Although useful, these measures fail when the tokens have different spelling or have minor typographical errors. For instance, *Jaccard* ('*gray color*', '*color gray*') = 1, but when the string is written with different spelling, then *Jaccard* ('*gr**a**y color*', '*colo**u**r gr**e**y*') = 0 because both words have different forms despite having the same meaning.

To overcome this problem, two approaches (also called *soft measures*) to generalize set matching have been introduced. In one approach, a character-level measure is used to estimate the similarity between tokens consisting from different strings. Only token pairs that are similar enough are considered. For example, in Monge and Elkan (1996), only pairs that have highest similarity scores according to a character-level measure are used to calculate the overall similarity (see Table 4 and Fig. 4). Michelson and Knoblock (2007) consider two tokens to match if their similarity score according to a character-level measure is above a predefined threshold. A drawback of this approach is that the threshold should be empirically chosen depending on the characteristics of the data set.

The second approach is a soft cardinality of set introduced by Vargas (2008). The idea for soft cardinality is that tokens similar

Table 4
Character-level similarities between *the grey colour* and *gray color* using SWG.

	the	grey	colour
gray	0.20	0.90	0.30
color	0.20	0.30	0.80
Max.	**0.20**	**0.90**	**0.80**

$$Monge-Elkan(the\ grey\ colour, gray\ color) = \frac{1}{3}(0.20 + 0.90 + 0.80) = 0.63$$

Fig. 4. Example of a soft matching measure.

Table 5
Soft cardinality of sets {gray, grey}, {gray, color} using SWG as the character-level measure.

	gray	grey	**Sum**	**1/sum**		gray	color	**Sum**	**1/sum**
gray	1.00	0.90	1.90	0.53	gray	1.00	0.30	1.30	0.77
grey	0.90	1.00	1.90	0.53	color	0.30	1.00	1.30	0.77
$\|T\|_{soft}$				1.06	$\|T\|_{soft}$				1.54

Table 6
Soft cardinality of the union of the sets {gray, color} and {the, grey, colour} using SWG as a character-level measure.

	gray	color	the	grey	colour	**Sum**	**1/sum**
gray	1.00	0.30	0.20	0.90	0.30	2.70	0.37
color	0.30	1.00	0.20	0.30	0.80	2.60	0.38
the	0.20	0.20	1.00	0.33	0.20	1.93	0.52
grey	0.90	0.30	0.33	1.00	0.30	2.83	0.35
colour	0.30	0.80	0.20	0.30	1.00	2.60	0.38
$\|T_1 \cup T_2\|_{soft}$							2.01

to others in the same set count less than tokens that are unique. Therefore, the soft cardinality of a set containing similar tokens should be less than that of a set containing the same number of tokens but significantly different.

For example, consider the two sets {*gray, grey*} and {*gray, color*}. The soft cardinality of the former set using SWG as the character-level measure is 1.06 (a bit more than one object) and the latter is 1.54 (more than one but less than two objects) (see Table 5). Therefore, soft cardinality provides a better measure of the unique concepts represented by the string.

In this paper, we use the function presented in Vargas (2008) to estimate the soft cardinality of a set. Let T be a set of n tokens: $T = \{T^1, T^2 \ldots T^n\}$, and $d(T^i, T^j)$ is a character-level similarity measure scaled in the range [0,1]. The soft cardinality of T is computed as:

$$|T|_{soft} = \sum_{i=1}^{n} \left[\frac{1}{\sum_{j=1}^{n} d\left(T^i, T^j\right)} \right] \quad (2)$$

After soft cardinality has been defined, we use it to calculate the cardinality of union, as illustrated in Table 6. The size of the intersection is computed through cardinalities of the two sets and their union (Vargas, 2008), as follows:

$$|T_1 \cap T_2|_{soft} = |T_1|_{soft} + |T_2|_{soft} - |T_1 \cup T_2|_{soft} \quad (3)$$

Finally, any set-matching method can be applied to compute the overall similarity by replacing the classic cardinality with soft cardinality. For example, soft Jaccard is computed as follows:

$$Jaccard(T_1, T_2) = \frac{|T_1 \cap T_2|_{soft}}{|T_1 \cup T_2|_{soft}} \quad (4)$$

It should be noted that set-matching methods using soft cardinality may give similarity scores greater than 1, because soft cardinality does not guarantee the traditional set inequalities $|T_1 \cap T_2| \leq \min(|T_1|, |T_2|)$ and $\max(|T_1|, |T_2|) \leq |T_1 \cup T_2|$ (Vargas, 2008).

4.3. Bag-of-tokens

The *bag-of-tokens* method combines the unique tokens from the two input strings into a single set (the "bag of tokens"). *Feature vectors* are then generated for both strings where each feature is the number of times the particular token in the bag appears in the string; this is denoted as *term frequency* (TF). However, this is not necessarily the best way to represent the strings, as common tokens like *a, to*, and *the* often have high frequencies. In Song, Zhu, and Chen (2014), it was observed that due to the short length of the phrases, most words appear only once in a text record and term frequency is therefore not efficient. *Term frequency-inverse document frequency* (TF-IDF) is therefore introduced to address this problem. It is the product of two statistics: the term frequency and its *inverse document frequency* (IDF_w). The latter is the total number of compared strings divided by the number of strings that contain the specific token. Metrics such as *cosine* (Cohen et al., 2003b), *Euclidean distance*, and *Manhattan distance* (Malakasiotis & Androutsopoulos, 2007) have been applied to compute the similarity between the two feature vectors (see Table 7). In Noh, Jo, and Lee (2015), TF-IDF and a set of 130 keywords were found to be the most promising components to estimate the similarity between patent documents. The length of patent documents is still much longer than typical text phrases that we consider.

Analogously to the set-matching techniques, these metrics compare tokens using exact match and ignore the degree of similarity between the tokens when generating the feature vectors. For instance, suppose that we have two strings $s_1 = play\ game$ and $s_2 = player\ gamer$. Our bag-of-tokens consists of four tokens {*play, game, player, gamer*} and the feature vectors corresponding to the two strings are $v_1 = [1, 1, 0, 0]$ and $v_2 = [0, 0, 1, 1]$. According to cosine measure, the similarity of these two vectors is 0 although they are quite similar.

To overcome this limitation, the *soft-cosine* measure has been introduced (Sidorov et al., 2014). It computes the similarity of each pair of tokens using a character-level measure; in (Sidorov et al., 2014), Levenshtein distance has been used. In the aforementioned example, if soft cosine (see Table 7) were applied with Jaro–Winkler as the character-level measure, the similarity of these two vectors would be 0.84. The same approach can also be applied to Euclidean and Manhattan metrics to produce their soft variants, although these have not been considered in literature so far (see Table 7).

Although generalized approaches suggest using a character-level measure to compare tokens, only a few combinations have been studied in the literature and were often tested only with one type of character-level measure. In addition, to best of our knowledge, there have not been unified tasks, data sets, and experimental setups in which all measures have been tested and evaluated for their usefulness. All these issues are addressed in this paper.

5. Experimental evaluations

We use the following experimental setup to analyze the performance of all the measures in relation to their properties, human intuition, clustering, and matching task. We aimed to find which measures

Table 7
Sequence, set and bag-of-tokens matching measures. For two strings s_1 and s_2, symbols $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are their vector representation and T_1 and T_2 denote their token sets. T_1^i is the i:th token in the set of tokens T_1 generated from string s_1. Symbol [] is used when referring to a multiset. Symbol sim_{ij} is the distance at the token level. Function d calculates the character-level similarity score between two tokens. Symbol n is the length of the feature vector.

Matching measures		
Sequence		Soft variant
Chaudhuri et al. (2003)	$\text{sim}_{ij} = \begin{cases} \text{sim}_{i-1,\ j-1} & if\ T_1^i = T_2^j \\ min\begin{cases} \text{sim}_{i-1,j}+1 \\ \text{sim}_{i,j-1}+1 \\ \text{sim}_{i-1,j-1}+1 \end{cases} & otherwise \end{cases}$	$\text{sim}_{ij} = \begin{cases} \text{sim}_{i-1,\ j-1} & if\ T_1^i = T_2^j \\ min\begin{cases} \text{sim}_{i-1,j}+(1-d(T_1^i,T_2^j)) \\ \text{sim}_{i,j-1}+(1-d(T_1^i,T_2^j)) \\ \text{sim}_{i-1,j-1}+(1-d(T_1^i,T_2^j)) \end{cases} & otherwise \end{cases}$
	Set	Soft variant
Braun- Banquet (Choi et al., 2010)	$\frac{\lvert T_1 \cap T_2\rvert}{\max(\lvert T_1\rvert,\lvert T_2\rvert)}$	$\frac{\lvert T_1 \cap T_2\rvert_{soft}}{\max(\lvert T_1\rvert_{soft},\lvert T_2\rvert_{soft})}$
Simpson (Choi et al., 2010)	$\frac{\lvert T_1 \cap T_2\rvert}{\min(\lvert T_1\rvert,\lvert T_2\rvert)}$	$\frac{\lvert T_1 \cap T_2\rvert_{soft}}{\min(\lvert T_1\rvert_{soft},\ \lvert T_2\rvert_{soft})}$
Jaccard (Rezaei & Fränti, 2016)	$\frac{\lvert T_1 \cap T_2\rvert}{\lvert T_1 \cup T_2\rvert}$	$\frac{\lvert T_1 \cap T_2\rvert_{soft}}{\lvert T_1 \cup T_2\rvert_{soft}}$
Dice (Brew & McKelvie, 1996)	$\frac{2\times\lvert T_1 \cap T_2\rvert}{\lvert T_1\rvert+\lvert T_2\rvert}$	$\frac{2\times\lvert T_1 \cap T_2\rvert_{soft}}{\lvert T_1\rvert_{soft}+\ \lvert T_2\rvert_{soft}}$
Rouge-N (Lin, 2004)	$((\frac{1}{p}\ )+(\frac{1}{r}))^{-1}$ $p=\ \frac{\lvert\lvert T_1\rvert \cap \lvert T_2\rvert\rvert}{\lvert\lvert T_1\rvert\rvert},\ r=\ \frac{\lvert\lvert T_1\rvert \cap \lvert T_2\rvert\rvert}{\lvert\lvert T_2\rvert\rvert}$	$((\frac{1}{p}\ )+(\frac{1}{r}))^{-1}$ $p=\ \frac{\lvert\lvert T_1\rvert \cap \lvert T_2\rvert\rvert_{soft}}{\lvert\lvert T_1\rvert\rvert_{soft}}, r=\ \frac{\lvert\lvert T_1\rvert \cap \lvert T_2\rvert\rvert_{soft}}{\lvert\lvert T_2\rvert\rvert_{soft}}$
Monge-Elkan (1996)		$\frac{1}{\lvert\lvert T_1\rvert\rvert}\sum_{i=0}^{\lvert\lvert T_1\rvert\rvert}\max_{1\le j\le\lvert\lvert T_2\rvert\rvert} d(T_1^i,T_2^j)$
	Bag-of-tokens	Soft variant
Cosine (Cohen et al., 2003b)	$\frac{\sum_{i=1}^{n}\boldsymbol{v}_1^i\boldsymbol{v}_2^i}{\sqrt{\sum_{i=1}^{n}(\boldsymbol{v}_1^i)^2}\ \sqrt{\sum_{i=1}^{n}(\boldsymbol{v}_2^i)^2}}$	$\frac{\sum_{i,j=1}^{n} d(T_1^i,T_2^j)\boldsymbol{v}_1^i\boldsymbol{v}_2^j}{\sqrt{\sum_{i,j=1}^{n} d(T_1^i,T_1^j)\boldsymbol{v}_1^i\boldsymbol{v}_1^j}\ \sqrt{\sum_{i,j=1}^{n} d(T_2^i,T_2^j)\boldsymbol{v}_2^i\boldsymbol{v}_2^j}}$
Euclidean (Malakasiotis & Androutsopoulos, 2007)	$1-\frac{\sqrt{\sum_{i=1}^{n}(\boldsymbol{v}_1^i-\boldsymbol{v}_2^i)^2}}{\sqrt{\lvert\boldsymbol{v}_1^i\rvert^2+\lvert\boldsymbol{v}_2^i\rvert^2}}$	$1-\frac{\sqrt{\sum_{i,j=1}^{n} d(T_1^i,T_2^j)(\boldsymbol{v}_1^i-\boldsymbol{v}_2^j)^2}}{\sqrt{(\sum_{i,j=1}^{n} d(T_1^i,T_1^j)\boldsymbol{v}_1^i\boldsymbol{v}_1^j)^2+(\sum_{i,j=1}^{n} d(T_2^i,T_2^j)\boldsymbol{v}_2^i\boldsymbol{v}_2^j)^2}}$
Manhattan (Malakasiotis & Androutsopoulos, 2007)	$1-\frac{\sum_{i=1}^{n}\lvert\boldsymbol{v}_1^i-\boldsymbol{v}_2^i\rvert}{\lvert\boldsymbol{v}_1^i\rvert+\lvert\boldsymbol{v}_2^i\rvert}$	$1-\frac{\sum_{i,j=1}^{n} d(T_1^i,T_2^j)\lvert\boldsymbol{v}_1^i-\boldsymbol{v}_2^j\rvert}{\sum_{i,j=1}^{n} d(T_1^i,T_1^j)\boldsymbol{v}_1^i\boldsymbol{v}_1^j+\sum_{i,j=1}^{n} d(T_2^i,T_2^j)\boldsymbol{v}_2^i\boldsymbol{v}_2^j}$

- are robust to text manipulation such as typographical errors and token change,
- correlate to human intuition,
- have higher similarity to such character strings that refer to the same entity;
- can be applied in clustering,
- can be applied to find entries in different databases.

We selected 10 matching techniques, 9 character-level measures, and 2 q-gram measures. This gave us $(10\times(9+2))=110$ soft token-level measures in total. We also considered exact match at the character level and the semantic measure Word2Vec as references. We therefore had 143 different combinations in our experiments, of which 52 exist and 91 are novel (see Table 8).

Word2Vec model, provided by Google,[1] has been trained using the English language (US). We slightly modified the model so that it outputs value 0 (instead of infinite) if a word cannot be recognized and value 1 if two words are equal, even if they are not recognized. Without this modification, infinite values would appear during the calculation of soft similarity.

All string characters were converted to lowercase as a preprocessing step in all tests because it can have positive effect on the accuracy (Uysal & Gunal, 2014). We also suppressed the spaces in the beginning and at the end of the strings if there were any.

5.1. Data sets

We used three publicly available data sets containing mostly English text (see Table 9):

- Titler[2] (Gali, Mariescu-Istodor, & Fränti, 2017),
- The Mopsi photo collection[3,4],
- Match sets[5] (Cohen, Ravikumar, & Fienberg, 2003a).

The *Titler* data set contains 4968 candidate title phrases extracted from 1002 English websites. The ground truth titles were manually annotated by two people independently on each other, and in the case of disagreement, a third person made a judgment between these two. The candidate title phrases were extracted automatically from the pages using the method in Gali et al. (2017); therefore, different typographical representations exist, such as *Hotspring* and *Hot spring, Park hotel & spa* and *Park hotel and spa*. The phrases were evaluated for their relevance so that the user rates them from 0 (*irrelevant*) to 5 (*excellent match*). For example, 12 candidate phrases were extracted for the restaurant *the Apollo*; six of them were rated 5, two were rated 4, and four were rated 3. The minimum number of phrases extracted for a web page was one and the maximum was 30 (Gali et al., 2017).

The *Mopsi photo collection* contains 42,739 geo-tagged photos collected since April 2016. Each photo has a short description (English or Finnish), time stamp, and the location where it was taken. Mopsi users can write a description immediately after taking a photo. Then, the Mopsi app will offer pre-written descriptions that the user can simply tap to use. The pre-written descriptions are obtained from photos near to the user. Therefore, photos taken at the same location tend to have similar descriptions when they describe the same object. The descriptions may contain typing errors.

Match sets are publicly available data sets that have been used to test similarity measures on matching tasks (Cohen et al., 2003a; Jimenez et al., 2009; Vargas, 2008). The data sets comprise seven domains, such as birds, business names, and games (see Table 9). Each domain consists of two or three databases collected from dif-

[1] https://code.google.com/archive/p/word2vec.
[2] http://cs.uef.fi/mopsi/titler.
[3] http://cs.uef.fi/mopsi/tools/photoclusters.php.
[4] http://cs.uef.fi/mopsi/PhotoDescriptionsClusters/dataset.zip.
[5] http://www.cs.cmu.edu/~wcohen/match.tar.gz.

Table 8
The tested 143 similarity measures. Blue cell refers to an existing and √ to a novel measure. The first row contains measures that use exact matching at the character level. The first column contains all character-level, q-grams, and semantic measures.

			Token-level									
			Set-matching						Bag-of-tokens			Seq.
		Ch/Q	Bra-Ban	Simpson	Jacc	Dice	Rouge	Mon-Elk	Cos	Eucl	Manh	Edit
	Exact match											
Character-level	Hamming		√	√	√	√	√	√	√	√	√	√
	Levenshtein						√			√	√	
	Dam-Levenshtein		√	√	√	√	√	√	√	√	√	√
	Needle-Wunsch		√	√	√	√	√	√	√	√	√	√
	SW		√	√	√	√	√	√	√	√	√	√
	SWG		√	√	√	√	√		√	√	√	√
	Jaro						√			√	√	
	Jaro-Winkler		√	√			√	√	√	√	√	√
	LCS		√	√	√	√	√	√	√	√	√	√
Grams	2-Grams						√			√	√	
	3-Grams		√	√	√	√	√	√	√	√	√	√
Semantic	Word2Vec		√	√			√	√				√

Table 9
Summary of the data sets.

Source	Data set	Size	Language	String length					
				Token			Character		
				Min	Av.	Max	Min	Av.	Max
Gali et al. (2017)	Titler	4968	English	1	3	8	4	14	39
This paper	Mopsi photos	1000	English Finnish	1	3	26	6	17	65
Cohen et al. (2003)	Bird Nybird	982	English	1	3	69	4	21	321
	Bird Scott1	38		2	3	8	7	20	58
	Bird Scott2	719		3	4	9	15	35	83
	Business	2139		1	3	8	4	19	51
	Game	855		1	5	55	4	27	255
	Park	654		2	3	12	6	16	58
	Restaurant	863		7	11	21	40	59	102

ferent sources. For example, the parks data set contains 396 national parks names from one listing and 258 from a second listing. Of these, 241 names describe the same park. In each data set, the entries contain different types of information that are joined by a tabulation character as follows:

- Ny Bird: scientific name, common name,
- Bird Scott1: web page link, common name, scientific name,
- Bird Scott2: web page link, common name, scientific name,
- Business: web page link, company name,
- Game: ID, name,
- Park: web page link, name,
- Restaurant: name, address, phone number, and brief description of the cuisine served.

The underlined fields are the identification keys (ID keys). Entries from different listings match if their ID keys are identical. These data sets are controlled; therefore, typing errors do not exist.

5.2. Text manipulation

We first examined how each measure performs under text manipulation: character change and token change. We selected 18 strings of different lengths[6] (see Fig. 5) as a baseline and applied several systematic changes.

We first made k random character changes and then l random token changes and report the average results. The expected result is that a syntactic measure has a linear correspondence to the number of character or token changes (k and l). This is denoted as *expected* in Figs 6 and 8. This assumption might not hold in all cases. Humans are known to be able to recognize the content even with the presence of very severe spelling errors, but in some other applications humans can be very sensitive to even small amount of character changes.

In testing, we computed the similarity between the original string and the manipulated string. On average, we observed the performance of the measures to be mostly invariant with regard to string length. We therefore report only the results for one selected string: *awesome animated monster maker: ultra-edition* (6 tokens and 40 characters). Due to the large number of tested measures, we only plot selected measures in Figs. 6 and 8. The rest

[6] http://cs.uef.fi/mopsi/TextManipulation/dataset.zip.

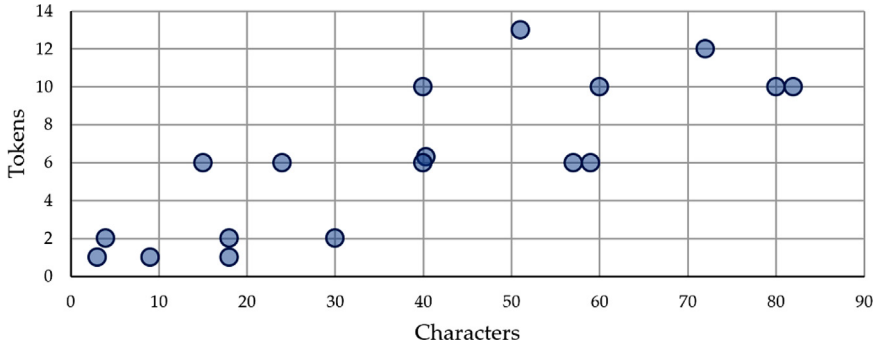

Fig. 5. Eighteen strings plotted based on how many characters and tokens they contain.

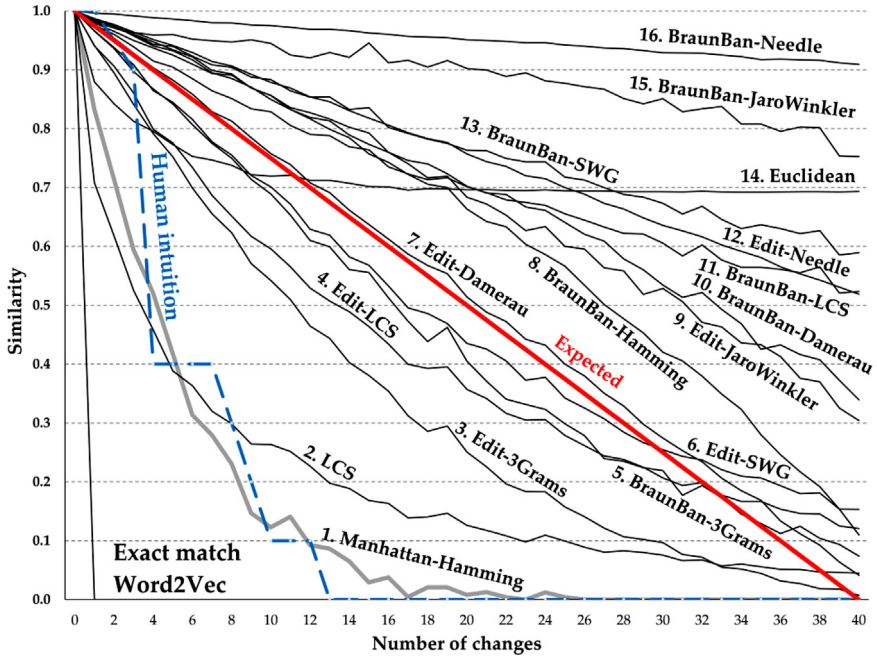

Fig. 6. Effect of character changes on the similarity measures.

of the measures have been clustered in a color pattern graph in Figs. 7 and 9 such that one line in Figs. 6 and 8 represents not only the named measure but also another with the same number in Figs. 7 and 9.

In Fig. 6, we observe that most character-level measures have a constant decrease of similarity when the number of changes increases, but that the amount of decrease varies. Levenshtein and its modified versions such as Damerau–Levenshtein correlate best with the number of character changes. LCS is one of the mostly affected measures because changes in the middle of the string produce significantly shorter common substrings. For example, when changing two characters in our example *awesome aRimated mXnster Maker: Ultra Edition,* the similarity drops to 0.58.

Q-gram measures show a uniform decrease with the number of characters being changed. They are slightly more sensitive to character changes because one change will destroy two or more bi- and trigrams. The token-level measures with exact match generally drop faster than the character-level measures and the q-grams because they discard the entire token even if only minor difference exists.

The performance of the soft measures is more stable than their corresponding crisp variants. In Fig. 6, we see that majority of the soft measures are less sensitive to the character change than expected. Indeed, because soft variants consider strings with minor differences as being similar, they also consider strings with major differences as having some similarity. One exception is the combination between Monge–Elkan and Smith–Waterman or SWG, which drops faster than expected. The combinations between Monge–Elkan and Levenshtein or Damerau–Levenshtein, together with the combination between set-matching methods, cosine and q-grams, correlate best with the expected result.

The semantic measure Word2Vec behaves very similar to exact match. Even a single character change usually brings the word out of the vocabulary, and therefore, it will not anymore match the original text. Exceptional cases appear only when a word changes to another word existing in the dictionary by luck, but the effect of this is insignificant. To sum up, semantic measures are not suitable for this kind of application.

To compare the measures with human intuition, we generated a ground truth depending on human understanding of the text. Eleven users were asked to perform the experiment as follows: one manipulated string was displayed to the user at a time (in the same order to all users) through an interface. The user had to write the correct string if able to recognize it and leave the input field empty otherwise. It should be noted that the same list of manipulated strings was given to all users and to all similarity measures. The process continued until all manipulations were tested. The average results were taken.

	Ch/Q	Edit	MongeElkan	Set matching methods and cosine	Manhattan	Euclidean
Exact match						
Hamming	7			8		
Levenshtein &Damerau-Levenshtein				10		
Needleman Wunch	12			16	1	14
Smith Waterman & SWG	6			13		
Jaro & Jaro Winkler	9		12	15		
LCS	2	4		11		
2Grams & 3Grams	3			5		
Word2Vec						

Fig. 7. Clusters of performance of similarity measures corresponding to Fig. 6. Exact match refers to standard measures and black cells are the unique patterns.

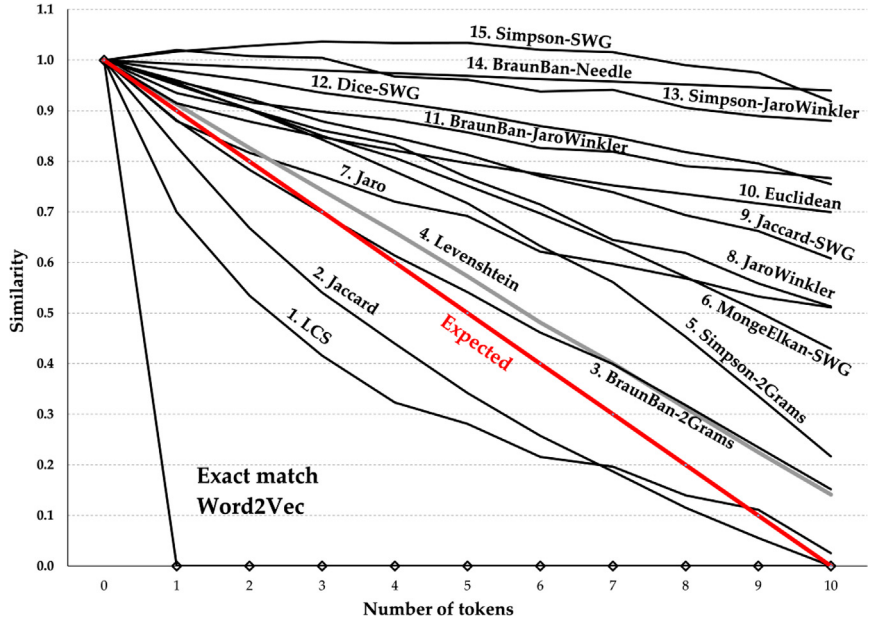

Fig. 8. Effect of token changes on the similarity measures.

	Ch/ Q	Edit	MongeElkan	Brau-Ban	Simpson	Jaccard	Dice & Rouge	Cosine	Manhattan	Euclidean
Exact match						2				
Hamming										
Levenshtein & Damerau-Levenshtein				6	12	6	9	6		
Needleman Wunch					14					
Smith Waterman & SWG	2		6	10	15	9	12	15	4	10
Jaro	7			11	13		12			
Jaro Winkler	8									
LCS	1			9	12	6	9	6		
2Grams					5		3			
3Grams				3		2				
Word2Vec										

Fig. 9. Clusters of performance of similarity measures corresponding to Fig. 8. Exact match refers to standard measures and black cells are the unique patterns.

As can be observed in Fig. 6, human intuition provides very different behavior compared to the expected linear behavior. Humans can perfectly recognize the text with up to 20% of character changes, depending on the text. One reason is that if a token was broken because of character changes, a human can still guess the correct word. In addition, the position of the characters being changed has an effect on human understanding. For example, a human can still read *tihs* correctly although the middle characters are swapped.

In conclusion, the majority of the measures performed well in this experiment, with the exception of exact match, LCS, token-level measures, and Manhattan soft variants, which dropped fast, as well as soft variants that combine set-matching methods or cosine with Needleman–Wunsch, Jaro and Jaro–Winkler, which dropped slowly. Euclidean soft variants may also be considered to have performed poorly because they did not see the differences between half-changed and totally changed strings. Levenshtein and Damerau–Levenshtein alone or combined with Monge–Elkan, as well as set-matching methods and the cosine combined with q-grams, and Monge–Elkan combined with Smith–Waterman, SWG, or LCS were closer to the expectation than other measures.

Fig. 8 illustrates that the majority of the measures had a uniform decrease with respect to the token change. Number 4 in Fig. 9 represents all these measures. Among the character-level measures, the LCS was the most sensitive to token change. Smith–Waterman, SWG, and the combination between Jaccard and q-grams dropped the same amount as Jaccard alone. The Euclidean and Manhattan measures had the same behavior no matter what character-level or q-grams measure it combines.

As assumed, the soft measures provided higher similarity scores than expected due to their ability to capture the similarity between similar and identical tokens. In Fig. 8, we observe that the similarity scores provided by the soft versions of the measures never reach 0, no matter how different the two strings are. Considering Needleman–Wunsch, Jaro, and Jaro-Winkler as secondary measures is not useful, as they provide high similarity regardless of the differences between the strings. Similar to character change, q-gram variants were always close to the expected results, having their worst performance with Simpson, Jaccard and Euclidean.

5.3. Correlation to human intuition

Next, we used the Titler data set to detect how well the similarity scores correlate with the human scores. We used the non-symmetric rank correlation *Somers' D* (Somers, 1962) of the computed similarity with respect to human score, because it takes equal similarities for unequal human scores into account. It is calculated as:

$$Somers'D = \frac{N_s - N_d}{N_s + N_d + N_t} \tag{4}$$

where N_s is the number of pairs ranked in the same order by both variables (see Fig. 10), N_d is the number of pairs ranked in reversed order, and N_t is the number of pairs that have different human scores but given equal scores by a similarity measure (Somers, 1962). Tie cases are ignored, as they do not have any impact on the correlation.

The results are summarized in Table 10. We observe that most measures have a positive correlation to the human ranking. The strength of the correlation is moderate (from 40% to 52%) for most of them. The correlation of the character-level measures Levenshtein, Damerau–Levenshtein, and Jaro, and the q-grams are slightly higher (from 50% to 52%) than that of the others (from 40% to 49%). Smith–Waterman and SWG have a weak correlation (16%). Word2Vec does not work well (4%) because titles usually consist

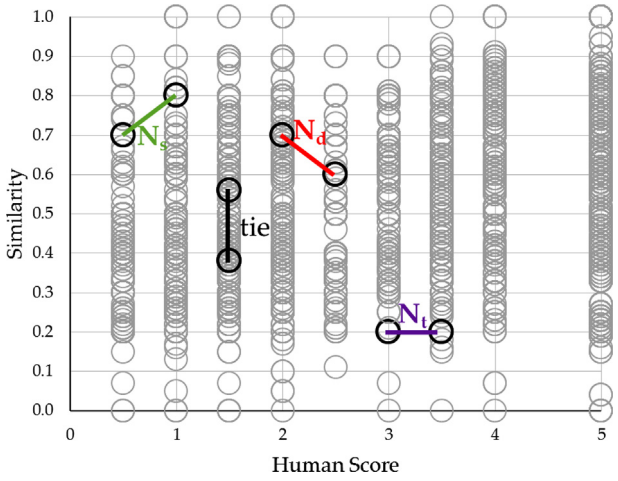

Fig. 10. Rank correlation parameters (Jaccard+3-Grams).

of multiple words that form meaningful entity; such form is not a part of the model.

The best performance of token-level measures was obtained by the soft versions of the set-matching methods when combined with q-grams (from 50% to 52%). The results indicate that there is at least one soft variant of each measure that correlates better to human judgment than with exact match. For example, Edit-LCS (52%) versus Edit-Exact (48%). Exceptions are the bag-of-tokens measures Euclidean and Manhattan, which are not improved by the soft versions. An exception among the set-matching methods is the Simpson measure, which has a weak correlation (14%), and its performance becomes even worse (from 14 to −1%) with almost all combinations. The reason is that Simpson provides similarity scores greater than 1 when minor differences exist between the strings. For example, it gives a similarity score of 1.6 to the strings *HotSpring* and *Hot spring home,* while the human score is only 3.

Word2Vec provides smaller correlation values than the syntactic measures. It often fails because of finding similarities when it should not. For example, it gives high soft similarity scores (∼90%) between the *Garfish seafood restaurant* and *seafood restaurant* because of the high similarity of the words' *garfish* and *seafood*. Semantically, the word Garfish is highly redundant but, as a restaurant name, it is an essential part of the title according to human intuition. Another example where semantic similarity fails is concatenated titles like *HotSprings* or *GetFit* because they are not part of the model dictionary.

To investigate why none of the measures correlated strongly with the human judgments, we analyzed human scores further and we observed that humans focus more on distinct words. For example, for the restaurant *Ventuno pizzeria*, if two candidate titles exist *Ventuno* and *pizzeria*, users consider *Ventuno* more relevant to the place than *pizzeria*. Therefore, they give scores of 4 to *Ventuno* but 0 to *pizzeria*. None of the syntactic measures can distinguish between generic and specific words without the existence of external information such as a corpus; therefore, to a measure both words are equally important.

We further observed that users pay less attention to typographical differences, as they consider the following phrases excellent matches:

- Freda's – Fredas
- Drom UK – Dröm UK
- Hot Spring – HotSpring
- Park Hotel and Spa – Park Hotel & Spa
- Holiday Inn Bristol Filton – Holiday Inn Filton-Bristol

Table 10
Summary of results (%) for correlation to human; exact match refers to standard measures; best, moderate, bad.

			Token-level									
			Set-matching						Bag-of-tokens			Seq.
		Ch/Q	Bra-Ban	Simpson	Jacc	Dice	Rouge	Mon-Elk	Cos	Eucl	Manh	Edit
	Exact match	40	46	14	46	45	45	46	48	44	46	48
Character-level	**Hamming**	41	47	14	48	47	47	48	49	44	46	50
	Levenshtein	52	48	7	49	48	48	50	49	44	46	52
	Dam-Levenshtein	52	48	6	49	48	48	50	49	44	46	52
	Needle-Wunsch	49	43	4	45	34	34	48	42	43	47	51
	SW	16	46	-1	46	44	44	49	45	43	46	51
	SWG	16	44	-4	44	40	40	47	43	42	47	51
	Jaro	51	43	-1	42	39	39	47	43	44	47	49
	Jaro-Winkler	46	43	-1	42	39	39	46	43	44	47	49
	LCS	47	47	6	48	47	47	50	48	44	46	52
Grams	**2-Grams**	51	49	13	50	**50**	**50**	50	52	44	46	52
	3-Grams	52	50	14	50	**50**	**50**	50	51	44	46	52
Semantic	**Word2Vec**	4	34	-5	34	34	34	35	34	36	26	36

Furthermore, human concentrates on the correctness of a phrase's structure, which is not shown in the numerical evaluation. For example, human gave a lower score for the similarity of the following phrases, but the measures tend to consider them highly similar:

- Out of the Blue – Out the Blue
- Arcata Pizzeria – At Arcata Pizzeria
- 3 Degrees – Degrees

All these factors have a significant impact on the degree of correlation between the measures and the human scores.

5.4. Correlation to distance

Two geo-tagged photos taken at the same location are more likely to have more similar descriptions than those taken at different locations (see Fig. 11). Accordingly, for a given input photo, a good similarity measure should rank the nearby photo similar more often than a random far-away photo. We performed the following experiment: for every photo, we randomly selected two other photos of which one must have been taken nearby (<20 m). We counted how many times the nearby photo's description was more similar to the original than the random one. The expected result should be more than 50%, which would be the result if no relationship exists between the descriptions. The result was also not expected to reach 100% as not all nearby photos describe the same object, and likewise, far-away photos might sometime have similar description, such as a restaurant chain in two locations.

We used a subset of 1000 Mopsi photos that have a description. The results presented in Table 11 indicate that all measures correlate positively. The highest counts (normalized by the number of photos) were in the range of 70%–72%. Levenshtein and Damerau–Levenshtein provided the best results (both 70%) among the character-level measures with 8 percentage points of improvement over the exact match. 2-Grams performed slightly better (70%) than 3-Grams (67%) in this experiment, but not any better than the character-level candidates.

The token-level measures (exact match) failed to capture the similarity between similar tokens with small artifacts. They would give equally high scores to *snow hotel* versus *snow hoteli*, and *snow hotel* versus *snow football*. The performance of every token-level measure was improved by at least one soft variant (see Table 11). The only exceptions were the bag-of-tokens measures Euclidean and Manhattan, which do not seem to benefit from a soft version. The token-level edit distance measure improved by five percentage points when using Smith–Waterman or the 2-grams at the character-level. The best result (72%) was obtained when combining the set-matching methods with SWG and 3-grams. These combinations rank the nearby photos similar more often than random far-away photos. The only exception here is Simpson, which was not among the best-performing measures.

Despite the soft variants being useful, some combinations provided worse results; for example, Simpson-Jaro (59%) and Simpson-Jaro-Winkler (60%) performed worse than their exact variants Simpson (67%), Jaro (64%), and Jaro-Winkler (64%).

In general, the soft measures performed better than character-level, q-gram, and token-level measures. The best combination was set-matching methods and the character-level measures Needleman–Wunsch, SWG, and q-grams.

Word2Vec performs very well in this application. This is because photos at the same location have typically different descriptions made by different people, who often use alternative partial synonyms such as *building, house* and *architecture*, or *statue* and *sculpture* or *scenery* and *view*. A semantic measure is therefore a good fit for this kind of situation.

5.5. Clustering

We next tested whether the measures are useful for clustering. From the Mopsi photos, we manually selected 180 photos and divided them manually into 15 groups, based on their text description, to represent distinct objects. This manual clustering represents the ground truth (GT) (see Fig. 12).

In testing, we grouped the photos into 15 clusters using an agglomerative clustering algorithm where, at each step, the pair of

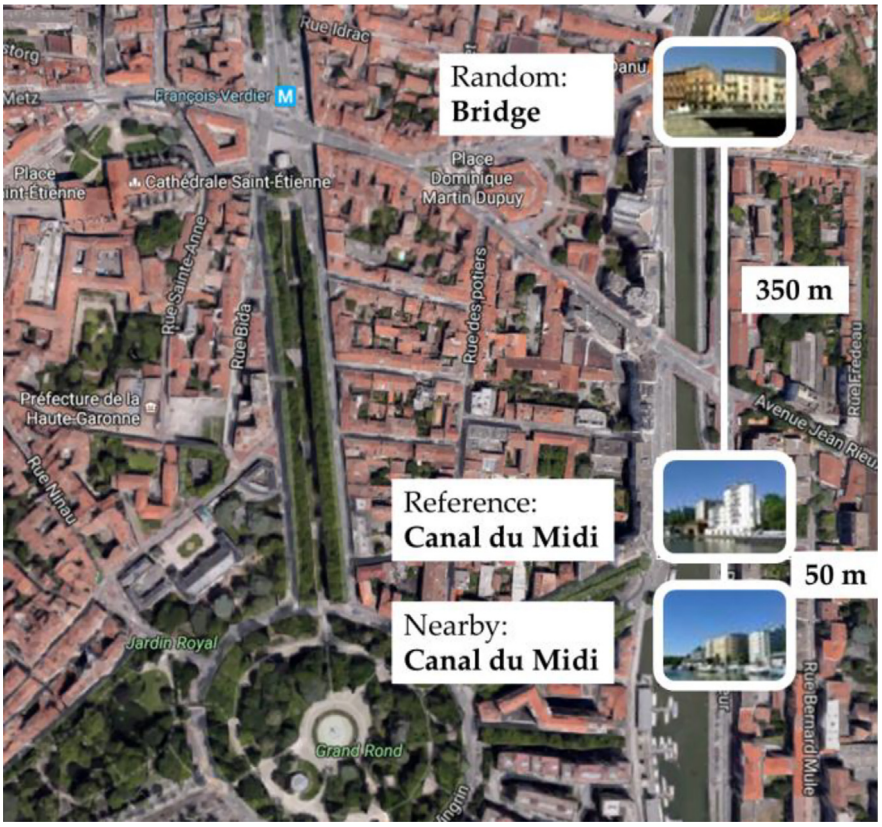

Fig. 11. Nearby photos have more similar descriptions than far-away photos.

Table 11
Summary of results (%) for correlation to distance; exact match refers to standard measures; best, good, neutral.

			Token-level									
			Set-matching						Bag-of-tokens			Seq.
		Ch/Q	Bra-Ban	Simpson	Jacc	Dice	Rouge	Mon-Elk	Cos	Eucl	Manh	Edit
	Exact match	62	65	65	65	65	65	65	65	67	65	65
Character-level	**Hamming**	63	67	66	66	66	67	66	66	67	65	65
	Levenshtein	70	67	67	67	67	67	67	69	65	65	62
	Dam-Levenshtein	70	67	67	67	67	67	66	68	65	64	65
	Needle-Wunsch	69	70	65	70	70	70	70	70	67	65	62
	SW	62	68	68	69	69	69	69	68	66	65	70
	SWG	62	72	69	72	72	72	67	70	65	65	67
	Jaro	64	67	59	67	67	67	67	66	65	65	62
	Jaro-Winkler	64	67	60	67	67	67	67	67	65	65	61
	LCS	67	69	68	69	69	69	70	71	66	66	65
Grams	**2-Grams**	70	71	69	70	**70**	**70**	70	70	69	65	70
	3-Grams	67	72	70	72	**72**	**72**	71	71	69	65	68
Semantic	**Word2Vec**	62	73	73	73	73	73	74	67	62	73	73

clusters that provides maximal improvement in total pairwise similarities within the clusters are merged. We applied each similarity measure to compute the pairwise similarities and compare the clustering result against the GT. The cluster quality was measured by the centroid similarity index (CSI) (Fränti, Rezaei, & Zhao, 2014); it determines how similar a clustering solution is to the GT solution. Given two clustering solutions $A=\{A_1, A_2 \ldots A_K\}$ and $B=\{B_1, B_2 \ldots B_K\}$ of K clusters, the CSI is computed as follows:

$$CSI = \frac{\sum_{i=1}^{K} n_{ij} + \sum_{j=1}^{K} n_{ji}}{2N} \quad (5)$$

Table 12
Summary of results (%) for clustering; exact match refers to standard measures; best, good, neutral, bad.

			Token-level									
			Set-matching						Bag-of-tokens			Seq.
		Ch/Q	Bra-Ban	Simpson	Jacc	Dice	Rouge	Mon-Elk	Cos	Eucl	Manh	Edit
	Exact match	47	63	74	67	66	66	66	67	58	66	63
Character-level	**Hamming**	42	60	69	59	69	69	71	69	58	69	61
	Levenshtein	64	63	72	66	62	62	68	69	61	68	63
	Dam-Levenshtein	64	58	71	66	70	70	68	72	61	69	63
	Needle-Wunsch	53	61	76	59	61	61	67	66	60	70	64
	SW	78	62	70	66	70	70	74	75	59	66	70
	SWG	72	60	62	63	64	64	73	67	61	65	65
	Jaro	59	49	50	49	53	53	63	51	60	69	54
	Jaro Winkler	57	48	55	56	54	54	67	52	61	69	57
	LCS	67	66	74	67	78	78	74	74	58	67	66
Grams	**2-Grams**	71	69	81	68	**74**	**74**	73	73	56	65	67
	3-Grams	72	69	75	72	**77**	**77**	69	73	60	65	69
Semantic	**Word2Vec**	46	60	74	60	61	61	67	58	65	71	57

Fig. 12. Example of three ground truth clusters.

where n_{ij} (respectively n_{ji}) is the number of objects that cluster A_i (respectively B_j) and its most similar cluster B_j (respectively A_i) have in common: $nij = |A_i \cap B_j|$. It can be calculated efficiently from the contingency table. Here, N is the total number of objects.

Random clustering would produce a value of CSI = 28%, so the expected result of a successful clustering should be higher than that. Viewing the results presented in Table 12, we observe that even the simplest method (exact match) provided CSI = 47%. The Smith–Waterman character-level measure provided CSI = 78%. The advantage is that it does not penalize one missing token. For example, it considers *Marriott* and *Bristol Marriott hotel* as a perfect match and, therefore, it correctly concludes that these two strings belong to the same cluster.

Among the token-level set-matching measures, Simpson (74%) performed better than any other token-level measure. This is because it normalizes by the length of the smaller string, which allows it to recognize the Marriott example as a perfect match.

The performance of all token-level measures was significantly improved by their soft variants. The percentage of improvement varies. Dice and Rouge benefit the most, having 12-percentage points of improvement when combined with 3-Grams. The Euclidean and Manhattan bag-of-tokens measures seemed not to benefit much from soft variants. The performance of the remaining measures improved by five to eight percentage points when combined with q-grams, Smith–Waterman, and LCS. The better performance of the soft variants can be explained by the fact that photo descriptions are often short (two tokens on average) with a higher probability of typing errors when using a small phone's keyboard, which is challenging to type on accurately, especially when in a hurry. Two such examples are *Lighttower* (Light tower) and *Bibliotechue Nationale* (Biblioteque Nationale).

In summary, soft measures outperformed character, q-gram and token-level measures in text clustering. The best clustering result (81%) was achieved when combining Simpson with 2-Grams. Using q-grams at the character level seem to benefit all token-level measures, but the combination with set-matching methods (Simpson, Dice, and Rouge) yielded the best results.

Word2Vec does not perform so well here mainly because of the multi-language text descriptions that it cannot handle. As a result, everything written in non-English language will be grouped into one cluster regardless of the meaning of the words.

5.6. Names matching

The quality of a measure can also be determined by its ability to find matching entries in databases created from different sources. We used match data sets and followed the testing procedure proposed in Cohen et al. (2003). Paired entries from two different databases belonging to the same set (domain) were compared and considered a match if their corresponding ID keys were identical.

In testing, we calculated the similarity score between the pairs. We sorted all the pairs according to the calculated similarity scores, as illustrated in Table 13. In an ideal case, all matches should have higher similarity scores and, as a result, appear in the sorted list before all the mismatch cases. We calculated precision

Table 13
Selected examples of pairs and their similarity scores. Red color indicates pairs that describe different entries and are therefore labelled as mismatches. The *F*-score for this ranking is 80%.

Similarity%	String 1	String 2	Key 1	Key 2
100	Hyperstudio	Hyperstudio	hyperstudio	hyperstudio
90.7	Mario Teaches Typing	Mario Teaches Typing 2	mariotype	foobar
74.9	Green Eggs and Ham	Green Eggs and Ham by Dr. Seuss	greeneggs	greeneggs
69.2	Fisher Price's Pirate Ship	Pirate Ship	pirateship	pirateship
69.1	Let's Color	Let's Learn Shapes & Colors	none	foobar
58.7	Catz	Catz, Your Computer Petz	catz	catz

Table 14
Maximum *F*-score (%) for the measure on matching problem; exact match refers to standard measures; best, good, neutral, bad.

			Token-level									
			Set-matching						Bag-of-tokens			Seq.
		Ch/Q	Bra-Ban	Simpson	Jacc	Dice	Rouge	Mon-Elk	Cos	Eucl	Manh	Edit
	Exact match	13	80	78	80	80	80	81	80	66	79	74
Character-level	**Hamming**	16	75	75	78	78	78	79	78	65	79	72
	Levenshtein	68	72	59	79	79	79	86	83	61	79	77
	Dam-Levenshtein	68	72	59	80	80	80	86	82	61	79	77
	Needle-Wunsch	58	69	56	80	80	79	85	84	57	80	70
	SW	73	63	21	62	62	61	84	62	59	80	72
	SWG	74	59	21	60	60	60	84	62	57	80	71
	Jaro	60	30	6	17	17	17	86	20	60	80	64
	Jaro Winkler	59	30	6	17	17	17	85	19	61	80	64
	LCS	65	75	69	81	81	81	86	83	62	80	77
Grams	**2-Grams**	76	80	78	**86**	**86**	**86**	87	83	64	78	79
	3-Grams	77	82	83	**87**	**87**	**87**	87	84	65	78	80
Semantic	**Word2Vec**	3	74	69	81	81	81	77	64	86	84	74

and recall as:

$$precision = \frac{c(i)}{i} \tag{6}$$

$$recall = \frac{c(i)}{m} \tag{7}$$

Where $c(i)$ is the number of correct matching pairs ranked before position i, and m is total number of correct matches. We, then computed the F-score of the ranking as:

$$F - score = \frac{2 \times precision \times recall}{precision + recall} \tag{8}$$

The results are summarized in Table 14 as averages of all domains. As can be observed, Smith–Waterman (73%) and SWG (74%) performed better than any other character-level measure. Q-grams performed slightly better (76% and 77%) than character-level candidates. Token-level measures performed better than character-level and q-gram measures by often providing 80% in the F-score.

The soft variants were less effective in this experiment. The majority of combinations led to either no improvement or even worse results; only Monge–Elkan benefitted from almost all soft versions (10 out of 11 variants). While the best result (87%) was obtained by combining 3-grams with the set-matching methods Jaccard, Dice, and Rouge and with Monge–Elkan, it seems that the soft version of the measures is generally not useful for this type of data set. This is because the data is well maintained, having no typing errors. Therefore, two entries with slightly different words such as *Silicon Valley Group, Inc* and *Silicon Valley Research Inc* are really different companies, but the soft variants would wrongly consider them as similar.

Some combinations with Word2Vec perform reasonably well, namely the Euclidean and Manhattan token level measures. However, the results are still worse than that of the q-grams.

In general, 3-grams were the most useful soft variant for this data as they benefitted 8 out of 10 token-level measures. The best results were achieved when 3-grams were combined with the set-matching methods Jaccard, Dice, and Rouge and with Monge–Elkan. This indicates that q-grams is a proper choice as a character-level measure regardless of the text type.

6. Conclusions

We have introduced a novel framework for a generic similarity measure. The framework has extensively tested and applied in four different applications. We also have reviewed and performed a systematic comparison of 143 similarity measures for various text types. We provide an open-source Java toolkit using unique adaptable components. It supports both the crisp and soft variants of all studied measures. The toolkit is easily available for researchers to verify the results and extend to other application. From the experiments, we learned several lessons as summarized below.

First, the crisp token-level measures perform better than the character-level measures when the order of the words varies. For

Table 15
Summary of the results in respect to the applications considered.

Char level	Both combined	Token level	Word2Vec
Text manipulation:			
Most methods (+)	Most methods (+)	Oversensitive (−)	
LCS oversensitive (−)			Oversensitive (−)
Q-grams oversensitive (−)			
Human intuition:			
Most methods (+)	Most token level + Q-grams (+)	Edit distance (+)	
Q-grams (+)	Edit distance + Any char level (+)	Simpson (−)	
Smith–Waterman/Gotoh (−)	Simpson + Any char level (−)		
Correlation to distance:			
Most methods (+/−)	Most token level + Q-grams (+)	Most methods (+/−)	
Damerau/Levenshtein (+)	Euclidean/ Manhattan/Edit (+/−)		Mostly best (+)
2-grams (+)			
Clustering:			
Smith–Waterman/Gotoh (+)	Most token level + Q-grams (+)	Simpson (+)	
Q-grams (+)	Bran-Ban worse (−)		
Hamming (−)			
Names matching:			
Hamming (−)	Most token level + Q-grams (+)		
	Monge–Elkan + Most char level (+)		
	Most token level + Jaro /Winkler (−)		

example, *Café Manta* and *Manta Café* would not be matched by character-level measures. However, the crisp variants start to perform poorly when even a single character is changed. Humans pay less attention to this kind of typographic errors. This is the main reason why the soft token-level measures performed better than their crisp variants. For example, the strings *gray color* and *colour grey* should be considered similar not only based on their semantic meaning but also syntactically. The crisp versions however, fail to recognize their similarity.

Thus, the main lesson is that the performance of all token-level measures was significantly improved by their soft variants. Overall, humans perfectly recognized the text with up to 20% of character changes, depending on the text.

The semantic measure Word2Vec, behaves poorly in most cases because even a single character change leads to the out-of-vocabulary situation. Further, words that were not included in the training material also lead to a failure. The correlation between the location and the photos descriptions is a positive exception because of the frequent use of synonyms by people when annotating photos describing the same location.

It is arguable that Word2Vec could produce better results if it was trained using a similar dataset as in the experiments. For example, the clustering dataset consists of mixed words of Finnish and English, which should be good for Word2Vec if trained accordingly. This also reveals the main deficiency of machine learning approaches: the need for application specific training data.

The performance of all measures also varies depending on the datasets. In case of well-maintained databases, the soft variants of the token-level measures do not provide additional benefits because the data having no typing errors. On the contrary, they can even slightly weaken the performance. For casual text containing errors, the soft variants are needed. Q-gram provides a good compromise, as they are also independent on the order of the words. In our experiments, they always performed close to the best token-level measures.

Based on the experiments, none of the measures can be classified as universal, i.e. working perfectly for all applications (see Table 15). However, among all tested combinations, the soft token-level measures based on set-matching methods and using q-grams at character-level provided good results in all experiments. These combinations are all novel:

- Dice (token-level) with 2-grams or 3-grams (character level)
- Rouge (token-level) with 2-grams or 3-grams (character level)
- Monge-Elkan (token-level) with LCS (character level)

There still remains a gap between the syntactic measures and human judgments caused by the lack of semantic analysis. In several cases, humans focus on the meaning of the words and they can conclude that the word *Ventuno* is more important than the word *Pizzeria* in the string *Ventuno Pizzeria*. If one of these two words were missing, humans give more weight on *Ventuno* because it identifies the service.

One limitation of the proposed framework is that it is designed mainly for text string consisting of natural language. It can be applied, into a certain extent, also to applications like bioinformatics. However, we have excluded measures utilizing application-specific features like *reverse distance* (Bulteau, Fertin, & Komusiewicz, 2014) and *string kernels* (Leslie, Eskin, & Noble, 2002) used in machine learning applications. Nevertheless, the results can still be highly relevant in bioinformatics. In fact, it would be an interesting research problem to see whether it would create a new state-of-the-art in that application area as well, possibly combining with the string reversing idea.

Another limitation is that, although the framework is general, it is an open question to what extent the results generalize to longer text like patent documents and to other applications. The DNA sequences would be worth to consider in the future.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2019.03.048.

Credit authorship contribution statement

Najlah Gali: Conceptualization, Methodology, Validation, Investigation, Resources, Writing - original draft, Writing - review & editing, Visualization, Supervision. **Radu Mariescu-Istodor:** Conceptualization, Formal analysis, Methodology, Software, Validation, Resources, Data curation, Writing - review & editing, Visualization. **Damien Hostettler:** Software, Validation, Resources, Data curation, Writing - review & editing, Visualization. **Pasi Fränti:** Conceptualization, Methodology, Writing - review & editing, Supervision.

References

Achananuparp, P., Hu, X., & Shen, X. (2008). The evaluation of sentence similarity measures. In *International conference on data warehousing and knowledge discovery* (pp. 305–316). Springer.

Agbehadji, I. E., Yang, H., Fong, S., & Millham, R. (2018). The comparative analysis of Smith-Waterman algorithm with Jaro-Winkler algorithm for the detection of duplicate health related records. In *IEEE international conference on advances in Big Data, computing and data communication systems* (pp. 1–10).

Alenazi, M., Reddy, D., & Niu, N. (2018). Assuring virtual PLC in the context of SysML models. In *International conference on software reuse* (pp. 121–136). Springer.

Barrón-Cedeño, A., Gupta, P., & Rosso, P. (2013). Methods for cross-language plagiarism detection. *Knowledge-Based Systems, 50*, 211–217.

Bär, D., Zesch, T., & Gurevych, I. (2013). Dkpro similarity: An open source framework for text similarity. In *Proceedings of the 51st annual meeting of the association for computational linguistics: System demonstrations* (pp. 121–126).

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., & Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems, 18*(5), 16–23.

Brew, C., & McKelvie, D. (1996). Word-pair extraction for lexicography. In *Proceedings of the 2nd international conference on new methods in language processing* (pp. 45–55).

Bulteau, L., Fertin, G., & Komusiewicz, C. (2014). Reversal distances for strings with few blocks or small alphabets. In *Lecture notes in computer science: 8486* (pp. 50–59). Springer.

Chaudhuri, S., Ganjam, K., Ganti, V., & Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning. In *Proceedings of the 2003 ACM SIGMOD international conference on management of data* (pp. 313–324).

Cheatham, M., & Hitzler, P. (2013). String similarity metrics for ontology alignment. In *International semantic web conference* (pp. 294–309). Springer.

Christen, P. (2006). A comparison of personal name matching: Techniques and practical issues. In *Sixth IEEE international conference on data mining-workshops* (pp. 290–294).

Choi, S. S., Cha, S. H., & Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics, 8*(1), 43–48.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003a). A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation: 3* (pp. 73–78).

Cohen, W., Ravikumar, P., & Fienberg, S. (2003b). A comparison of string distance metrics for name-matching tasks. In *II Web* (pp. 73–78).

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

Ehsan, N., & Shakery, A. (2016). Candidate document retrieval for cross-lingual plagiarism detection using two-level proximity information. *Information Processing & Management, 52*(6), 1004–1017.

Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering, 19*(1), 1–16.

Franco-Salvador, M., Rosso, P., & Montes-y-Gómez, M. (2016). A systematic study of knowledge graph analysis for cross-language plagiarism detection. *Information Processing & Management, 52*(4), 550–570.

Fränti, P., Rezaei, M., & Zhao, Q. (2014). Centroid index: Cluster level similarity measure. *Pattern Recognition, 47*(9), 3034–3045.

Friedman, C., & Sideli, R. (1992). Tolerating spelling errors during patient validation. *Computers and Biomedical Research, 25*(5), 486–509.

Gali, N., Mariescu-Istodor, R., & Fränti, P. (2016). Similarity measures for title matching. In *23rd International conference on pattern recognition (ICPR)* (pp. 1548–1553).

Gali, N., Mariescu-Istodor, R., & Fränti, P. (2017). Using linguistic features to automatically extract web page title. *Expert Systems with Applications, 79*, 296–312.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of Molecular Biology, 162*(3), 705–708.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell Labs Technical Journal, 29*(2), 147–160.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association, 84*(406), 414–420.

Jimenez, S., Becerra, C., Gelbukh, A., & Gonzalez, F. (2009). Generalized mongue-elkan method for approximate text string comparison. In *International conference on intelligent text processing and computational linguistics* (pp. 559–570). Springer.

Jimenez, S., Gonzalez, F., & Gelbukh, A. (2010). Text comparison using soft cardinality. In *International symposium on string processing and information retrieval* (pp. 297–302). Springer.

Keskustalo, H., Pirkola, A., Visala, K., Leppänen, E., & Järvelin, K. (2003). Non-adjacent digrams improve matching of cross-lingual spelling variants. In *International symposium on string processing and information retrieval* (pp. 252–265). Springer.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological Review, *104*(2), 211.

Lertvittayakumjorn, P., Kertkeidkachorn, N., & Ichise, R. (2017). Resolving range violations in DBpedia. In *Joint international semantic technology conference* (pp. 121–137). Cham.: Springer.

Leslie, C., Eskin, E., & Noble, W. S. (2002). The spectrum kernel: A string kernel for SVM protein classification. *Biocomputing*, 565–575.

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

Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. In Text summarization branches out: Proceedings of the ACL-04 workshop *(8)*.

Malakasiotis, P., & Androutsopoulos, I. (2007). Learning textual entailment using SVMs and string similarity measures. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing* (pp. 42–47).

Metzler, D., Dumais, S., & Meek, C. (2007). Similarity measures for short segments of text. In *European conference on information retrieval* (pp. 16–27). Springer.

Michelson, M., & Knoblock, C. A. (2007). Unsupervised information extraction from unstructured, ungrammatical data sources on the world wide web. *International Journal of Document Analysis and Recognition (IJDAR), 10*(3–4), 211–226.

Mihalcea, R., Corley, C., & Strapparava, C. (2006). Corpus-based and knowledge-based measures of text semantic similarity. *American Association for Artificial Intelligence, 6*, 775–780.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM, 38*(11), 39–41.

Monge, A. E., & Elkan, C. (1996). The field matching problem: Algorithms and applications. In *KDD-96 proceedings* (pp. 267–270).

Moreau, E., Yvon, F., & Cappé, O. (2008). Robust similarity measures for named entities matching. In Proceedings of the 22nd international conference on computational linguistics *(1)* (pp. 593–600).

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology, 48*(3), 443–453.

Noh, H., Jo, Y., & Lee, S. (2015). Keyword selection and processing strategy for applying text mining to patent analysis. *Expert Systems with Applications, 42*(9), 4348–4360.

Recchia, G., & Louwerse, M. M. (2013). A comparison of string similarity measures for toponym matching. In *COMP@ SIGSPATIAL* (pp. 54–61).

Rezaei, M., & Fränti, P. (2014). Matching similarity for keyword-based clustering. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 193–202). Springer.

Rezaei, M., & Fränti, P. (2016). Set matching measures for external cluster validity. *IEEE Transactions on Knowledge and Data Engineering, 28*(8), 2173–2186.

Rieck, K., & Wressnegger, C. (2016). Harry: A tool for measuring string similarity. *The Journal of Machine Learning Research, 17*(1), 258–262.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*(3), 379–423.

Sidorov, G., Gelbukh, A., Gómez-Adorno, H., & Pinto, D. (2014). Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas, 18*(3), 491–504.

Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology, 147*(1), 195–197.

Snae, C. (2007). A comparison and analysis of name matching algorithms. *International Journal of Applied Science, Engineering and Technology, 4*(1), 252–257.

Somers, R. H. (1962). A new asymmetric measure of association for ordinal variables. *American Sociological Review,* 27(6), 799–811.

Song, S., Zhu, H., & Chen, L. (2014). Probabilistic correlation-based similarity measure on text records. *Information Sciences, 289*, 8–24.

Song, Y., Batjargal, B., & Maeda, A. (2019). Cross-language record linkage based on semantic matching of metadata. *The Database Society of Japan English Journal, 17*(1).

Sun, Y., Ma, L., & Wang, S. (2015). A comparative evaluation of string similarity metrics for ontology alignment. *Journal of Information and Computational Science, 12*(3), 957–964.

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *European conference on machine learning* (pp. 491–502). Springer.

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

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management, 50*(1), 104–112.

Van der Loo, M. P. (2014). The stringdist package for approximate string matching. *The R Journal, 6*(1), 111–122.

Vargas, S. G. J. (2008). *A knowledge-based information extraction prototype for data-rich documents in the information technology domain*. National University of Colombia (Bogota).

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the Fellegi–Sunter model of record linkage. In *Proceedings of the section on survey research methods* (pp. 354–359).

Wu, Z., & Palmer, M. (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics* (pp. 133–138).