Improvement of imperfect string matching based on asymmetric n-grams

Julian Szymański and Tomasz Boiński

Department of Computer Architecture,
Gdańsk University of Technology, Poland,
julian.szymanski@eti.pg.gda.pl
tomasz.boinski@eti.pg.gda.pl

Abstract. Typical approaches to string comparing treats them as either different or identical without taking into account the possibility of misspelling of the word. In this article we present an approach we used for improvement of imperfect string matching that allows one to reconstruct potential string distortions. The proposed method increases the quality of imperfect string matching, allowing the lookup of misspelled words without significant impact on computational effectiveness. The paper presents the proposed method, experimental data sets and obtained results of comparison to state of the art methods.

Keywords: imperfect string matching, information retrieval, recognition memory

1 Introduction

The transformation performed by recognition memory from the space of perceptions into a space of inner representations, allows one to put on the perceived things structures that organize the perception. This mechanism is also used on the linguistic level where particular words should be selected from the stream of perception, which is a prerequisite of language understanding. The recognition memory works for all different modes of perception. On a visual level it reconstructs elements of the images, and an on audio channel it allows to identify words.

In this article we focus on the aspect of recognition memory which allows the correction of string characters using the algorithm performing efficient imperfect string matching. It should be stressed that this is only one of the tasks that recognition memory can perform, and its implementation can serve as an elementary block of a cognitive system. The task of string recognition (or matching the string to the one stored in the dictionary) is a very important element in information retrieval [1] systems so it will be analysed also from that perspective. The task of imperfect string matching can also be applied in spell checking, thus making its implementation very useful.

String comparison performed by machines is typically a binary process: two literals are the same or not. The computers usually do not consider the strings as similar, but only the same or different. While we are processing the natural language in written form, where word misspellings are common, the selection of words, non-words and

their correspondences become an important issue. It should be noticed that selecting a subset of words from the set of all possible character combinations is significant.

In retrieval systems, selecting the exact matches of the particular user query is not sufficient, due to possible misspellings and also different inflections (not considering semantics of the utterances). Other uses of presented methods include sequence analysis systems such as comparing genomes [2].

2 Imperfect matching methods

Most of the approaches to imperfect string matching are based on metrics that describe distances between two strings. The most intuitive was introduced by Hamming [3]. The Hamming distance for two words of the same length is the number of places where they differ. For text correction this measure is not very useful as it takes into account only one possible typing error – change of a letter.

An improvement of the string matching algorithm has been proposed by Levenshtein [4]. The measure calculates a minimal number of operations, such as letter removal or inserting that leads to transforming one string to another. Two same strings will have a Levenshtein distance equal to 0. If they are different, minimal distance will be 1 and the maximal will be the length of the longer string. The algorithm in its basic form is a bit slow but it has been optimized and according to Exorbyte it allows one to check 2.5 mln words in 10 seconds [5] and in that form it is used eg. in *Yahoo!* search engine.

The Levenshtein distance has one serious drawback. It does not take into account operation of changing two letters. Research indicates that this type of error is one of the common misspellings that humans do (80 %) [6]. The measure was adopted to take into account this additional operation. The modified distance is named Damerau-Levenshtein. Despite the adding of one additional operation that does not seem to be complicated, it strongly influence the computational effectiveness of the algorithm.

Aforementioned methods for string comparison are basic approaches for imperfect string matching [7]. Their advantage is not very complicated implementations. The Hamming distance is fast but it is only limited to strings of the same length. The Damerau-Levenshtein measure is known to have approximately 80% efficiency in misspellings correction. The drawback of these methods is their limitation to the selected number of basic operations on strings that not cover all possible errors humans can do. E.g. semantically the Polish word *pies* (dog) is closer to the misspelled word *pias* than *wies* (countryside), but using presented distances they have the same distance because they have the same difference – a letter change.

The problems with measures based on editing distances have been compensated for by using extended representations of the strings. One of the most popular approaches are n-grams that are widely used in sequence analysis. An n-gram is an n–element subsequence taken from a given sequence. Usually the n-grams are created by dividing the original sequence on the pieces having equal n length that may overlap. The method is similar to the approach used in Support Vector Machines where the increase of dimensions with a particular kernel allows one to separate objects that are not separable in higher dimensions. The imperfect string matching based on n-grams representation

MOST WIEDZY Downloaded from mostwiedzy.pl

usually gives more precise results than the former presented approaches based on editing distances [8].

The n-gram approach also allows one to save disk memory. The methods based on editing distances require you to store a whole dictionary in memory. The n-grams approach can optimize memory usage by creating the all n-grams dictionary. It should be noticed that the size of the dictionary is proportional to n. If n is considerably smaller than average string length, it will take much less space. To optimize the n-grams approach and to create effective word representation it is required to store only references to the proper n-grams that store only reference i.d. instead of whole strings, which is known to be more efficient [8].

To compare the n-grams representations, usually two approaches are used: slower, based on Marcov models that allows capturing the order of n-grams, and a faster method that is based on comparison of sets, where order is omitted. Due to the effectiveness of n-grams representations usage of Tversky index (known for set-theoretic approach for string comparing) provide similar results to Marcov models [8].

The methods presented above are widely used. Among them open-source approaches for string correction should be mentioned, namely Hunspell, Aspell, Ispell [9]. They can be used both as stand alone applications, or by third party software. Ispell is based on Damerau-Levenshtein metrics where the distance is not higher then 1 and it does not take into account spelling rules. Its successor Aspell is optimized for over 70 languages. It is also based on Damerau-Levenshtein metrics but contains optimisations and extensions for improving the speed of comparisons by using cached list of elements called soundslike. The Aspell has been integrated with many applications such as Notepad++, the older version of Opera, Gedit, AbiWord. Hunspell is a spelling corrector and morphological analyser. It has been enriched with an approach based on n-grams. It has been integrated with Google Chrome, LibreOffice, LyX, Mozilla Firefox, Opera 10+.

3 Extension of n-grams: bi2quadro-grams

In our approach we use a different method of string representation than that typically used with n-grams. We performed series of experiments comparing different combinations of n-grams used for representation as well as a source word (given for assessment) and target words (words stored in the dictionary). The experiments have shown that much better results can be achieved if we use different representations of words in the dictionary and source words. Typical approaches use the same n for source and target word, typically $n = 2$. In our approach the dictionary words are represented with quadro-grams and the source word is represented with bi-grams. As representations of source and target words are different the key issue is the way of their comparison. We can not compare source and target n-grams directly, as their n numbers are different. Thus, the method for comparing different n-grams lengths is crucial here.

3.1 Different n-grams length

Having different n-grams length the calculation of the similarity metric is performed within different dimensions. Thus, both n-grams – source and target should be mapped

MOST WIEDZY Downloaded from mostwiedzy.pl

into the same dimension, in other words their number should be equal. The number ng of n-grams generated for a given string equals $ng = length(string) - (n-1)$. The solution for performing comparison of n-grams with different lengths was to add additional blank letters to the representation with bigger n, and perform the similarity evaluation after mapping the bi-gram onto fragments of higher n-grams. In that way each of bi-grams have their corresponding place in higher n-gram representation. In our case we test tri- and quadro-grams.

Tri-gram representation is extended by adding one blank letter at the end of the last gram, quadro-grams are extended by adding blank letter at the end of the last gram and blank letter at the beginning of the first one. The example of partition of the strings for comparing tri- and quadro-grams with bi-grams has been shown in Table 1.

Table 1. Example for dividing the string on bi/tri/quadro-grams.

string	n-grams								
klawiatura	kl	la	aw	wi	ia	at	tu	ur	ra
klawiatura	kla	law	awi	wia	iat	atu	tur	ura	ra_
klawiatura	_kla	klaw	lawi	awia	wiat	iatu	atur	tura	ura_

3.2 Calculating n-grams similarity

The similarity between two strings can be expressed as a number of matched n-grams. This value is in the range $[0, source\_bi\text{-}grams\_number]$. After normalization the strings similarity represented with n-grams is expressed using Formula 1.

$$similarity = \frac{number\_of\_matched\_bi\text{-}grams}{total\_number\_of\_bi\text{-}grams}, \tag{1}$$

In the results we presented we use distance between two strings that is calculated as $distance = 1 - similarity$.

The similarity between two n-grams having different n can be calculated by counting the number of n-grams from the smaller n representation in the higher one. However, this approach gives good results only if the source and target words are the same. If in the source word some errors appear, such as insertion of an additional letter, or shifting of letters, the approach gave unintuitive results.

The examples in Tables 2 and 3 show such cases when a letter is removed or inserted. The examples also show that different similarity values can be achieved for different n-gram sizes – the last column shows the number of n-grams matched.

What can be seen from the examples provided in Tables 2 and 3 in the case of letter removal, using both tri-grams and quadro-grams representations for target word gave identical results. In the case of letter insertion, using quadro-grams gave better results than using bi- or tri-grams. However not all bi-grams of the source word could be matched.

As the comparison of n-grams should take into account different possibilities of spelling errors, we identify different types of matching:

MOST WIEDZY Downloaded from mostwiedzy.pl

Table 2. Letter removal – comparison of bi-, tri- and quadro-grams.

n-gram type	string	n-grams							matched n-grams
source string	podstwka	po	od	ds	st	tw	wk	ka	7
bi-grams	podstawka	po	od	ds	st	ta	aw	wk	4
tri-grams	podstawka	pod	ods	dst	sta	taw	awk	wka	6
quadro-grams	podstawka	_pod	pods	odst	dsta	staw	tawk	awka	6

Table 3. Letter insertion – comparison of bi-, tri- and quadro-grams.

n-gram type	string	n-grams							matched n-grams
source string	tabelica	ta	ab	be	el	li	ic	ca	7
bi-grams	tablica	ta	ab	bl	li	ic	ca	a_	2
tri-grams	tablica	tab	abl	bli	lic	ica	ca_	a__	2
quadro-grams	tablica	_tab	tabl	abli	blic	lica	ica_	ca__	5

- exact match: $ab \rightarrow abcd$.
- shifted match $ab \rightarrow acbd$.
- commutative match $ab \rightarrow cabd$.
- reverse match.: $ab \rightarrow bacd$.
- adding additional letter $ab \rightarrow abcx$

The introduction of those matchings increases efficiency of the algorithm. E.g. in case of letter removal the number of matched bi-grams increases (Table 4).

Table 4. Letter removal – comparison of bi-, tri- and quadro-grams.

n-gram type	string	n-grams							matched n-grams
source string	podstwka	po	od	ds	st	tw	wk	ka	7
bi-grams	podstawka	po	od	ds	st	ta	aw	wk	4
tri-grams	podstawka	pod	ods	dst	sta	taw	awk	wka	7
quadro-grams	podstawka	_pod	pods	odst	dsta	staw	tawk	awka	7

Table 5. Weights for different types of matchings

Type of matching	Symbol	Weight
exact match	ab	$1,0$
shifted match	bc, da	$0,8$
commutative match	ac, db	$0,8$
reverse match	ba, cb, ad, ca, bd	$0,1$
adding additional letter	x	$0,7$

MOST WIEDZY Downloaded from mostwiedzy.pl

The introduction of different types of matching allows us to add weights to the similarity measure that focus the similarity measure on the particular aspect of the matching. Without the weights the results can be misleading. E.g. distance between „podstwka" and „podstawka" will be 0 which is obviously not true. Thus each type of the aforementioned matchings was assigned a weight as presented in Table 5.

Taking weights into account gives more intuitive results. The example for a word with a removed letter is presented in Table 6. Type, number of found corresponding n-grams, and distance between the source and target word is presented in Table 7.

Table 6. Letter removal – example including weights.

n-gram type	string	n-grams							matched n-grams
source word	podstwka	po	od	ds	st	tw	wk	ka	7
bi-grams	podstawka	po ab	od ab	ds ab	st ab	ta -	aw -	wk -	4
tri-grams	podstawka	pod ab	ods ab	dst ab	sta ab	taw ac	awk bc	wka bc	7
quadro-grams	podstawka	_pod ab	pods ab	odst ab	dsta ab	staw ac	tawk bc	awka bc	7

Table 7. Letter removal – distance including weights.

n-gram type	string	number of bi-grams	distance
source word	podstwka	7	–
bi-grams	podstawka	$4ab$	$1 - 4 \cdot 1/7 \approx 0,43$
tri-grams	podstawka	$4ab$, $1ac$, $2bc$	$1 - (4 \cdot 1 + 1 \cdot 0,8 + 2 \cdot 0,8)/7 \approx 0,09$
quadro-grams	podstawka	$4ab$, $1ac$, $2bc$	$1 - (4 \cdot 1 + 1 \cdot 0,8 + 2 \cdot 0,8)/7 \approx 0,09$

In all cases, results obtained by using tri- and quadro-grams were better than when using only bi-grams. For exact, shifted, commutative and reverse matchings the results obtained for tri- and quadro-grams were identical. When the source word had an additional letter inserted, the usage of quadro-grams gave better results than bi- and tri-grams representations. Such an example is presented in Table 8. Type, number of found corresponding n-grams, and distance between the source and target word is presented in Table 9.

MOST WIEDZY Downloaded from mostwiedzy.pl

Table 8. Letter insertion – example with weights.

n-gram type	string	n-grams							matched n-grams
source word	tabelica	ta	ab	be	el	li	ic	ca	7
bi-grams	tablica	ta ab	ab ab	bl x	li x	ic -	ca -	a_ -	2
tri-grams	tablica	tab ab	abl ab	bli x	lic x	ica -	ca_ -	a__ -	2
quadro-grams	tablica	_tab ab	tabl ab	abli x	blic x	lica da	ica_ da	ca__ da	5

The examples presented above indicate that the standard n-grams approach allows you to precisely compare only identical words. This method does not take into account possible spelling errors. Introduction of weighted types of matchings allows for recognition of misspelled words based on their similarity to the correct ones. Thus, despite having to map different dimensions of n-grams, obtained results show that by the introduction of handling for spelling errors, quality of matching increases.

Table 9. Letter insertion – distance with weights.

n-gram type	string	number of bi-grams	distance
source word	tabelica	7	–
bi-grams	tablica	$2ab$, $2x$	$1-(2\cdot 1+2\cdot 0,7)/7\approx 0,51$
tri-grams	tablica	$2ab$, $2x$	$1-(2\cdot 1+2\cdot 0,7)/7\approx 0,51$
quadro-grams	tablica	$2ab$, $3da$, $2x$	$1-(2\cdot 1+3\cdot 0,8+2\cdot 0,7)/7\approx 0,17$

3.3 Different word lengths

Bi-grams matching allows string comparison when for each bi-gram of source word we have a corresponding bi-gram available for the target word. If a target word is shorter than the source one, some of bi-grams cannot be matched. In that case last grams from the target word will not be compared, and thus they do not influence the results. It results in the cases where comparing the word $klawiatora$ with $klawiatura$ or $klawiaturaaaaa$ gives the same result which is not acceptable.

To compensate for this we added a penalty for a difference in word length between target and source one. The penalty is calculated as presented by Formula 2. It is then added to the distance, thus increasing it for every additional letter.

$$penalty = \frac{length\_of\_source\_word - length\_of\_target\_word}{10} \tag{2}$$

The difference between lengths has been divided by 10 to be normalized into $[0, 1]$. We assume that the difference in lengths of source and target words is smaller than 10. The penalty allows one to eliminate the words that are similar to the source one in the n-grams located at the beginning, but which are longer.

MOST WIEDZY Downloaded from mostwiedzy.pl

4 The experiments

For the experiments presented in this paper two open-source dictionaries were used:

- LibreOffice dictionary – 50 911 words,
- Open Source English words dictionary of 153 222 words [10]

They contains together 162 463 different words, as 41 670 were found in both dictionaries. From those dictionaries a corpora containing 1000 correct words. This corpora was then modified to include 10%, 20%, ... , 100% of misspelled words. As a set of target words we use three publicly available dictionaries: *wikipedia*, *aspell* and *trec*.

Five different approaches to correct distorted words were tested and compared with bi2quadro measure. The results for 1000-word corpora have been presented in Figure 1.

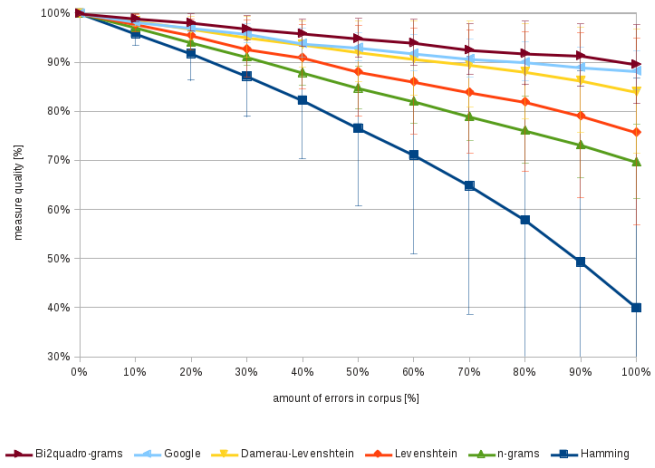

Fig. 1. Quality tests for different approaches to imperfect matching.

The quality of the algorithm depends on the degree of correctness of the corpora. For corpora that included only properly spelled words for all cases the proposed measure reaches 100% of correctness which indicates that it does not introduce noise to the dictionary. In all other cases, where test corpora contained erroneous words, our proposed measure achieved better results than other measures. It also behaves better than the one used by Google, that has been tested by providing sample strings into the Google search box and comparing the recommendation of correction returned by search engine.

The proposed measure was also tested using the *wikipedia*, *aspell* and *trec* corpora. The results have been computed using an F-measure and presented in Figure 2. Once again, bi2quadro-grams measure achieved better results than other widely used measures obtaining an average score around 90%.

MOST WIEDZY Downloaded from mostwiedzy.pl

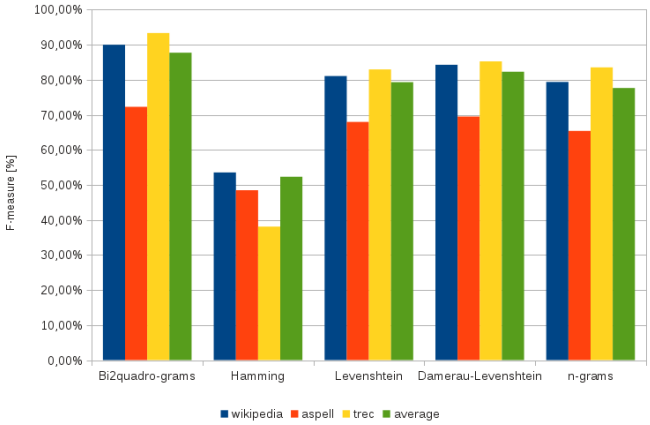

Fig. 2. Quality tests for different measures using *wikipedia*, *aspell* and *trec* corpora.

The proposed approach is also competitive in terms of performance. As in other approaches the time needed to perform word lookup is directly related to the size of the corpora. The proposed measure is only slightly slower than the standard n-grams approach, about 2-times slower than Hamming measure and faster than Levenshtein and Damerau-Levenshtein approach while yielding better results (Figure 3).

5 Conclusions

The proposed method of bi2quadro-grams achieves better results than other widely used approaches. The proposed measure achieved almost 90% accuracy, outperforming other known metrics like Hamming, Levenshtein or n-grams measures. The proposed metric also outperforms widely used dictionaries found in Microsoft Word in terms of time. For the dictionary of 162 463 words, the lookup took only 10 ms, whereas Word needed over 15 ms. Bi2quadro-grams also usually produces only one word as a result. Most of the widely-used dictionaries present lengthy lists of words for the user to choose from. As such, the proposed metric could be used in any word lookup and correction applications, yielding better results than most of the available solutions while preserving the computational time.

The presented approach we plan to use in our search engine `http://kask.eti.pg.gda.pl/BetterSearch` aiming at improving information retrieval from Wikipedia. The module that allows you to correct the keywords entered by the user should improve precision of the search. Beside information retrieval, this system offers additional functionalities that allows one to go beyond keyword-based search [11].

ACKNOWLEDGEMENTS

This work has been supported by the National Centre for Research and Development (NCBiR) under research Grant No. SP/I/1/77065/1 SYNAT: ”Establishment of the uni-

MOST WIEDZY Downloaded from mostwiedzy.pl

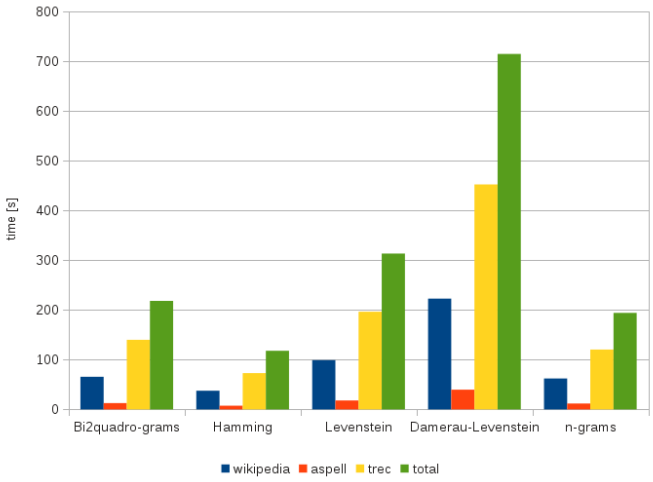

Fig. 3. Performance tests for different measures using *wikipedia*, *aspell* and *trec* corpora.

versal, open, hosting and communication, repository platform for network resources of knowledge to be used by science, education and open knowledge society".

References

1. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
2. Saxena, S., Jónsson, Z., Dutta, A.: Small rnas with imperfect match to endogenous mrna repress translation. Journal of Biological Chemistry **278** (2003) 44312–44319
3. Hamming, R.: Error detecting and error correcting codes. Bell System technical journal **29** (1950) 147–160
4. Lcvenshtcin, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics-Doklady. Volume 10. (1966)
5. Sulzberger, C.: Efficient implementation of the levenshtein-algorithm. `http://www.levenshtein.net/` (2009) [Online: 28.02.2012].
6. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. ACM **7** (1964) 171–176
7. Hall, P., Dowling, G.: Approximate string matching. ACM Computing Surveys (CSUR) **12** (1980) 381–402
8. Navarro, G., Baeza-Yates, R., Sutinen, E., Tarhio, J.: Indexing methods for approximate string matching. IEEE Data Engineering Bulletin **24** (2001) 19–27
9. Atkinson, K.: Gnu aspell. `http://aspell.net/` (2011) [Online: 07.03.2012].
10. WinEdt: Winedt dictionaries - english (uk). `tug.ctan.org/tex-archive/systems/win32/winedt/dict/uk.zip` (2010) [Online: 14.03.2012].
11. Deptula, M., Szymański, J., Krawczyk, H.: Interactive information search in text data collections, Springer (in print) (2012)

MOST WIEDZY Downloaded from mostwiedzy.pl