Accepted Manuscript

Noise-tolerance feasibility for restricted-domain Information Retrieval systems

Katia Vila, Antonio Fernández, José M. Gómez, Antonio Ferrández, Josval Díaz

PII: S0169-023X(13)00021-9
DOI: doi: 10.1016/j.datak.2013.02.002
Reference: DATAK 1420

To appear in: *Data & Knowledge Engineering*

Received date: 17 January 2011
Revised date: 18 September 2012
Accepted date: 14 February 2013

Please cite this article as: Katia Vila, Antonio Fernández, José M. Gómez, Antonio Ferrández, Josval Díaz, Noise-tolerance feasibility for restricted-domain Information Retrieval systems, *Data & Knowledge Engineering* (2013), doi: 10.1016/j.datak.2013.02.002

Noise-tolerance feasibility for restricted-domain Information Retrieval systems

Katia Vila[1,2], Antonio Fernández[1], José M. Gómez[2], Antonio Ferrández[2], Josval Díaz[1]

[1]University of Matanzas, Department of Informatics
Varadero Road, 40100 Matanzas, Cuba

[2]University of Alicante, Department of Software and Computing Systems
San Vicente del Raspeig Road, 03690 Alicante, Spain

{kvila,antonio,jmgomez}@dlsi.ua.es
{josval.diaz,antonio.fernandez}@umcc.cu

Contact information for the corresponding author:
- Name: Katia Vila
- Address: University of Alicante, Department of Software and Computing Systems, San Vicente del Raspeig Road, 03690 Alicante, Spain

Noise-tolerance feasibility for restricted-domain Information Retrieval systems

Abstract

Information Retrieval systems normally have to work with rather heterogeneous sources, such as Web sites or documents from Optical Character Recognition tools. The correct conversion of these sources into flat text files is not a trivial task since noise may easily be introduced as a result of spelling or typeset errors. Interestingly, this is not a great drawback when the size of the corpus is sufficiently large, since redundancy helps to overcome noise problems. However, noise becomes a serious problem in restricted-domain Information Retrieval specially when the corpus is small and has little or no redundancy. This paper devises an approach which adds noise-tolerance to Information Retrieval systems. A set of experiments carried out in the agricultural domain proves the effectiveness of the approach presented.

Keywords:
information retrieval, noise-tolerance, restricted domain, edit distance

1. Introduction

Human beings continuously confront noise in texts when they read or write documents. By noise we mean "any kind of difference in the surface form of an electronic text from the intended, correct or original text" [1]. Noise may appear as a result of writers' spelling mistakes, typeset errors

or problems with special character encoding, and these errors are currently particularly frequent in, for example, user-generated contents (wikis, blogs, emails, etc.). Noise may also be a result of errors caused by the automatic processing of documents. For example, Optical Character Recognition (OCR) tools convert handwritten, typewritten or printed documents into machine-encoded texts for their further processing by search engines. Common errors caused by OCR applications include the substitution of a character (e.g. *fear* vs. *tear*), the merging of two characters into one (*rna* vs. *ma*), the generation of two characters from one (*dam* vs. *clam*), or the division of a word through the insertion of spaces. The majority of computational approaches attempt to deal with these noise errors by comparing noisy terms with those stored in a lexicon. However, the main problem of these approaches is that many noisy terms may also be correct terms stored in the lexicon.

Noise errors are easily overcome by human beings, but cause erroneous results in applications that process electronic texts in an automatic manner [2, 3]. These applications also have to work on restricted domain texts, in which corpora are usually small, have little or no redundancy, and are focused on a technical and specific topic with a special vocabulary which is normally stored in Knowledge Organization Systems[1] (KOS) such as thesauri or ontologies (e.g. the AGROVOC[2] thesaurus in the agricultural domain or

[1]Knowledge Organization Systems include a variety of schemes that organize, manage, and retrieve information. This term is intended to encompass all types of schemes for promoting knowledge management [4].

[2]AGROVOC, http://www.fao.org/agrovoc/

the UMLS[3] in the medical domain).

Each application confronts noise problems in several ways. For example, [5, 6] presents a study of the effects of noise on automatic summarization from OCR documents. The authors of these approaches reach the conclusion that noise seriously decreases the precision of automatic summarization, principally as a result of incorrect sentence tokenization. They therefore propose to spell check the documents and to perform the summarization from words rather than sentences. Likewise, [7] suggests that the solution may be not to deal with noise, but to summarize, using document style features rather than sentences. Another work is that of Palmer and Ostendorf [8], in which the authors propose modeling the errors caused by a speech recognizer, but this approach requires a profound knowledge of the kind of noise errors that can be found in the data.

With regard to noise influence on Question Answering (QA) applications, it is important to mention the work of Aunimo et al. [9] in which a QA system that works with incomplete and noisy data (specifically emails and mobile short messages) is described. This system compares the user's question with a set of previously stored queries, each of which has its corresponding answer, thus signifying that neither answer extraction nor noise treatment is performed.

The approach presented in this paper extends previous work of the authors [10] by including a more exhaustive description and discussion of the proposed edit distances and how they are used to add noise-tolerance facili-

[3]UMLS: Unified Medical Language System, http://www.nlm.nih.gov/research/umls/

ties to Information Retrieval (IR) systems. Moreover, the experiments have been extended in order to measure and analyze the benefits obtained. It deals with the effects of noise in IR applications because IR is usually at the core of most of the previously mentioned applications, since it quickly reduces the quantity of text to which computationally expensive techniques are applied. Many IR systems do not have inbuilt support for dealing with noise in a given corpus. The rationale behind such a choice is because corpora usually consist of huge amounts of redundant documents in which the expected answer[4] to a query is often repeated in a large numbers of documents, with and without noise. A redundant corpus thus avoids the situation of IR systems being affected by noise problems. Unfortunately, this is only true for redundant open domain corpora, since restricted domain corpora may be small, and with little or no redundancy [11]. Non-redundant corpora therefore lead to a situation in which the information that the IR system is seeking may only be available in very few documents, and if they are affected by noise, the information may never be found. This is the scenario that we confront, one which hampers the use of IR systems in real-world situations in which (i) a restricted-domain and non-redundant corpus is used, and (ii) noise is unavoidable.

IR approaches dealing with noise are detailed in the following section (Sect. 2). Various edit distance algorithms are then studied in Sect. 3, of which the best is selected. In Sect. 4 an extension of an edit distance algorithm for considering comparisons between single words and multi-words

[4]Henceforth we use "answer" to mean the information required by the user's query. This information is in the document or passage returned by the IR system.

is presented. Our approach for adding noise-tolerance to IR systems is described in Sect. 5, while in Sect. 6 and Sect. 7 we respectively discuss the resources used and the set of experiments carried out. Our conclusions and future work are shown in Sect. 8.

2. Related work on dealing with noise in IR systems

IR systems are based on comparing text strings between the user's query and the corpus in which the answer should be found. Specifically, from a user's query, an IR system returns a list of relevant documents which may contain the answer to the query [12]. Noise can therefore appear in (i) the query, because its terms may be written incorrectly; or (ii) the corpus, since it must be automatically processed to obtain a set of text files as input of the IR system, such as the Web, PDF (Portable Document Format) files, or files processed from OCR or Automatic Speech Recognition tools [13].

2.1. Dealing with noise in IR queries

Most IR systems advocate noise correction by means of spell checkers [14]. In order to detect the noisy terms, they apply different heuristics, such as the non-inclusion in a previously defined lexicon or in a log of previous IR queries. They subsequently select the most similar stored terms according to distance measures (e.g. Levenshtein distance [15]). The main drawbacks of this are that there may not be a restricted-domain lexicon containing the required coverage in order to make this approach possible, and that they cannot deal with noisy terms which also appear in the lexicon as correct terms (e.g. *fear* vs. *tear*). Some approaches therefore add language models to these lexicons [16]. For example, Cucerzan and Brill [17] logs of user queries from

an internet search engine are used to obtain the language model which is used in the spelling correction of new queries. Li et al. [16] propose a method for the use of distributional similarity between two terms estimated from query logs in learning improved query spelling correction models. However, this method does not work with correct terms that are not in the lexicon or with less frequent noise errors. Other researchers [18] propose the use of new web searches in order to obtain alternatives for noisy terms. As was previously stated, this kind of approach requires open-domain corpora and performs better with high redundant corpora.

Similar approaches [19] measure the impact of noisy queries on the performance of classical stemming-based approaches on Spanish corpora. The authors adopted the noise correction scheme, in which the misspelled words in the query are replaced by their candidate corrections, proposed by several correction algorithms. They conclude that classic stemming-based approaches are highly sensitive to misspelled queries, particularly in the case of short queries. Such a negative impact is appreciably reduced by the use of contextual correction, although there is still an important decrease in precision (about -50% with an error rate of 50%). Moreover, this approach does not deal with noisy words that are legitimate words but semantically incorrect.

2.2. Dealing with noise in IR corpora

There are less approaches dealing with noise in IR corpora, because IR systems usually work in a huge repository of documents [20]. Most of these approaches carry out this task by means of spell checking, like Taghva and Stofsky [21], in which an approach that confronts OCR errors is presented.

Further similar approaches can be studied in TREC *Confusion Track* [22, 23].

Other approaches propose filtering the noise in the corpus by discarding noisy terms. Some examples can be found in bilingual corpora, principally when they are parallel (e.g. [24], [25], [26] or [27]). These kinds of approaches are also based on the redundancy of the corpus, and are not therefore effective in small and non-redundant restricted-domain corpora, in which the system cannot afford to discard any piece of information owing to the small size of the corpus.

With regard to noise tolerance approaches, some techniques [28, 29] enable approximate searches by manually generating a set of modified patterns from the original user pattern (e.g. phonetic similarities that often occur in multilingual scientific encyclopedias, along with normal typing errors such as omissions or the swapping of letters). The main drawback of this approach is that it requires manual adaptation to each corpus and language. Moreover, it also deals with the spelling noise introduced by users, but it does not work properly with errors introduced by automatic OCR tools. Other approaches add noise tolerance by means of query expansion with new terms obtained by adding common corruption errors previously found in the corpus or obtained from lists of pairs of correct and incorrect words, as can be seen in the work of Hawking et al. [30], but this requires a previous knowledge of the kind of errors in the corpus. Similarly, the work of Tong et al. [31] proposes query expansion by adding query term variants found in the terms that are not in the corpus. These are selected by using a statistical word bigram modeling and are measured by an edit distance. Likewise, Ng et al. [32] build a set of *double-dot-5-grams* for each topic statement. However, the performance of

this approach is quite low.

The approach presented in this paper overcomes the aforementioned drawbacks in that: (i) it does not require any special corpora (with redundancy or preprocessing); (ii) it does not require a profound knowledge of the kind of noise errors that can be found in the data; (iii) it is noise-tolerant because it does not correct or discard the noisy words in the corpus, since we intend to work in non-redundant restricted domains; (iv) it is based on restricted domain resources (lexicons, thesauri or ontologies), and it can deal with noisy terms that are also in these resources (e.g. *fear* vs. *tear*); (v) performance is maintained even though a noisy restricted-domain corpus is used; and (vi) it can deal with multi-words.

3. Selection of the edit distance algorithm

Various algorithms for computing string similarity currently exist. Edit Distance or Levenshtein distance [15] determines the differences between two words by computing the minimum number of operations required to transform one string into another. An "operation" can be an insertion, deletion or substitution of a character in the string. This distance is a generalization of the Hamming distance [33], which only considers the substitution operation for same-length strings. Some variants of distances that extend Edit Distance are: Damerau-Levenshtein distance [34], which considers the interchanges of two characters and a new operation called transposition; Needleman-Wunsch [35], which only adds a variable adjustment for the cost of the failures (insertion/deletion). Furthermore, the Jaro [36] or Jaro-Winkler [37] distance works properly when similarity is measured for

short strings (e.g., people's names). The Jaro-Winkler algorithm has a fixed length prefix in which transformations are carried out in a special manner by using a static scale factor.

However, these edit distance algorithms have several drawbacks, among others: (i) Levenshtein's distance does not take into account the position in which the operation occurs; (ii) Needleman-Wunsch distance applies penalties but without considering the kind of transformation that is carried out (e.g. different penalties should be applied for deletion and substitution operations, or for the replacement of an accented vowel with its non-accented counterpart, in comparison to the replacement of one consonant with another different one); and (iii) Jaro-Winkler distance fails in those cases that analyzed words have a prefix that is different from the fixed length prefix in the algorithm. These drawbacks hamper the use of these approaches in noise detection for restricted domain IR, thus requiring a new edit distance algorithm (described as follows).

3.1. Extended Edit Distance

In order to solve the aforementioned problems, we proposed a new algorithm for computing edit distances [38]: the Extended Edit Distance (DEx: "Distancia de Edición eXtendida"). This algorithm is an extension of Levenshtein's algorithm, with which penalties are applied by considering what kind of operation or transformation is carried out in what position, along with the character involved in the operation. DEx considers (i) an etymological analysis of the word, (ii) the occurrence of prosodic and orthographic alternations in several languages, and (iii) flexibility when typos occur. In addition to the cost matrixes used by Levenshtein, DEx also obtains the

Longest Common Subsequence (LCS) [39] and other helpful attributes for determining similarity between strings in a single iteration.

This information is used in DEx to apply certain penalties according to:

- Position of transformations: if they occur in the stem (i.e., further to the left) of the word, the penalty will be greater than for those occurring further to the right of the word.

- Character involved in the transformation and kind of transformation: this allows different penalties, because we consider that a higher penalty should be applied for a transformation between two characters with a high frequency in the language (the frequency for each character is automatically calculated from a dictionary of the language). In this way, we can deal with the prosodic alternations such as the replacement of an accented vowel with its non-accented counterpart, which is not highly penalized by assigning the same frequency to the vowel and its accented counterpart (e.g. a and á). Therefore, it allows a flexible adaptation to the specific language in which DEx is applied, an issue that is not considered by other distance measures. Similarly, substitution of one character by another or transposing two adjacent characters implies two single operations: the deletion and insertion of a character.

DEx algorithm consists of the following steps, which are defined by Fernández et al. [38]:

1. The Levenshtein matrix that contains the words to be analyzed is generated. For example, the matrix for words “afrecholk” and “afrechillo”

Table 1: Example of steps 1, 2 and 3 in DEx: obtaining Levenshtein matrix, LCS and OC.

		a	f	r	e	c	h	i	l	l	o
	0	1	2	3	4	5	6	7	8	9	10
a	1	0	1	2	3	4	5	6	7	8	9
f	2	1	0	1	2	3	4	5	6	7	8
r	3	2	1	0	1	2	3	4	5	6	7
e	4	3	2	1	0	1	2	3	4	5	6
c	5	4	3	2	1	0	1	2	3	4	5
h	6	5	4	3	2	1	0	1	2	3	4
o	7	6	5	4	3	2	1	1	2	3	3
l	8	7	6	5	4	3	2	2	1	2	3
k	9	8	7	6	5	4	3	3	2	2	**3**

		O	O	O	O	O	O	S	O	I	S

is shown in Table 1. The lowest cell on the far right of the matrix allows us to discover the minimum cost (in this case, three transformations should take place) of transforming the noisy word "afrecholk" into the word "afrechillo".

2. The path corresponding to the LCS is determined in accordance with Hirschberg [39]. In order to obtain this path, the starting point is the lowest right-hand cell. It is then necessary to move backwards through the matrix towards the top left-hand cell which is of minimum value, priority being given to the diagonal when the same value exists in bordering (or nearby) cells. In Table 1 the LCS of the example "afrecholk"-"afrechillo" is shown by coloring the cells in gray.

3. The Operation Chain (OC) is generated from the previously detected LCS. To this aim, each movement within the matrix is shown as a different kind of operation:

- A vertical movement is interpreted as a ***D****elete* operation.
- A horizontal movement is interpreted as an ***I****nsertion*.
- A diagonal movement is interpreted as a ***S****ubstitution*.
- Finally, if source and target cells have the same value, this is interpreted as a *N**O** operation*.

The OC would have an equal or higher length than the longest word of the compared words depending on the realized operations. According to the previous example, the OC would be: "OOOOOOSOIS". This is shown at the end of Table 1.

4. Evaluating Equation 1 with the OC found and those characters involved in each operation.

$$DEx = \sqrt[8]{\frac{\sum_{i=0}^{l-1} V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right) (2R_{max}+1)^{L-i}}{N}} \qquad (1)$$

where:

O : Operation chain (O-No operation, I-Insertion, D-Deletion, S-Substitution).

O_i : Operation in position i.

V : this is formalized as the following vector

$$V = \left\{ \begin{array}{c} (0,0) : o \\ (1,0) : i \\ (0,1) : d \\ (1,1) : s \end{array} \right\}$$

$c1$, $c2$: analyzed words or strings.

$c1_j$: character j of word $c1$.

$c2_k$: character k of word $c2$.

P : weight assigned to each character. These weights are obtained by calculating the frequency of each character in a general-language dictionary and some extended characters like the punctuation marks, etc. These characters are then ordered and a number is assigned to them, starting at 1, until the amount of characters is obtained in reverse order, as is shown as follows.

$$P = \left\{ \begin{array}{cccccccccccc}
a & : & 52 & c & : & 44 & g & : & 36 & í & : & 51 & ú & : & 41 \\
i & : & 51 & l & : & 43 & b & : & 35 & j & : & 27 & w & : & 20 \\
e & : & 50 & t & : & 42 & y & : & 34 & á & : & 52 & 1 & : & 19 \\
o & : & 49 & u & : & 41 & f & : & 33 &) & : & 25 & ñ & : & 18 \\
s & : & 48 & d & : & 40 & v & : & 32 & (& : & 25 & 0 & : & 17 \\
r & : & 47 & p & : & 39 & ó & : & 49 & q & : & 24 & 2 & : & 16 \\
n & : & 46 & m & : & 38 & x & : & 30 & k & : & 23 & - & : & 15 \\
 & : & 45 & h & : & 37 & z & : & 29 & é & : & 50 & 3 & : & 14
\end{array} \right.$$

$P_{(c1_j)}$: weight of character $c1_j$, where,

$$j = \left\{ \begin{array}{cc} j+1 & \text{if } O_i \neq I \\ j & \text{if } O_i = I \end{array} \right\}$$

$P_{(c2_k)}$: weight of character $c2_k$, where,

$$k = \left\{ \begin{array}{cc} k+1 & \text{if } O_i \neq D \\ k & \text{if } O_i = D \end{array} \right\}$$

L : length of the longest word in the selected dictionary. For example, as the longest length of a word in AGROVOC is 23 (e.g. *glicosfingofosfolípidos*), the value of L could be 23. However, it is worth noting that its value depends on the dictionary of words.

l : length of the edit operation chain.

R_{max} : maximum amount of characters in the general-language dictionary used for the generation of the set of weights P.

N : this is defined as follows, $N = \sum_{i=0}^{L-1}(2R_{max}+1)^i$. This value is calculated for the worst possible case, i.e., with a string of a length L (23) filled with the most costly characters in the dictionary (e.g. "a" and "i" with a cost of 52 and 51, respectively, according to P shown above) when the most costly operator (i.e., substitution) is applied.

In Equation 1, it can be observed that the term $V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)$ is the Cartesian product that analyzes the importance of carrying out the operation $V_{(O_i)}$ between characters $c1_j$ and $c2_k$. Term $(2R_{max}+1)^{L-i}$ penalizes the position of the operation in such a way that the further left the operation is (i.e., near the root of the word) the greater the penalty will be. N is the term that normalizes the distance in the $[0, 1]$ interval. The eighth root in Equation 1 is applied in order to avoid low results and to ensure that the order relation is not affected.

Table 2 shows the values of every parameter for calculating DEx between words "afrecholk" and "afrechillo". After applying DEx in the example, distance $DEx = 0.028$ is obtained between "afrecholk" and "afrechillo", thus

showing that they may be similar words.

Table 2: Example of step 4 of DEx: evaluation of Equation 1 with OC and involving characters.

O_i	$V_{(O_i)}$	c1	$P_{(c1_j)}$	c2	$P_{(c2_k)}$	$V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)$	i	$L-i$	$(2R_{max}+1)^{L-i}$	$V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max}+1)^{L-}$
O	$(0,0)$	a	52	a	52	0	0	24	$1.87E+49$	0
O	$(0,0)$	f	33	f	33	0	1	23	$1.66E+47$	0
O	$(0,0)$	r	47	r	47	0	2	22	$1.47E+45$	0
O	$(0,0)$	e	50	e	50	0	3	21	$1.30E+43$	0
O	$(0,0)$	c	44	c	44	0	4	20	$1.15E+41$	0
O	$(0,0)$	h	37	h	37	0	5	19	$1.01E+39$	0
S	$(1,1)$	o	49	i	51	100	6	18	$9.02E+36$	$9.02E+38$
O	$(0,0)$	l	43	l	43	0	7	17	$7.98E+34$	0
I	$(1,0)$		0	l	43	43	8	16	$7.06E+32$	$3,03E+34$
S	$(1,1)$	k	23	o	49	72	9	15	$6.25E+30$	$4.50E+32$

$Data:$			$CalculatingDEx:$	
$l=10$	$L=24$	$R_{max}=56$	$\sum_{i=0}^{l-1} V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max}+1)^{L-i} =$	$9,02E+38$
$N=\sum_{i=0}^{L-1}(2R_{max}+1)^i = 2.12E+51$			$\frac{\sum_{i=0}^{l-1} V_{(O_i)} * \left(P_{(c1_j)}, P_{(c2_k)}\right)(2R_{max}+1)^{L-i}}{N} =$	$4.25E-13$
			$DEx =$	0.028

Finally, as DEx is evaluated by using the minimal operation chains, and is generated by the application of the LCS algorithm in the dynamic programming matrix for DEx, the DEx algorithm's order being equal to the Edit Distance algorithm in $(O(m,n))$, where m and n are the length of compared strings.

Although experiments in sections 7.1.1 and 7.1.2 will show that DEx is a good candidate for noise tolerance, it is important to highlight that it must be extended if it is required to deal with multi-words. The rationale behind this is that multi-words commonly appear in restricted domains (e.g. scientific names of different kinds of fir tree within an agricultural domain: *abies alba*, *abies balsamea*, *abies sachalinensis*, etc.).

4. Extension of DEx for multi-words

A further example of the current shortcomings of the measures of distance among character strings (e.g. Levenshtein, Jaro-Winkler, DEx, etc.) is that they do not operate with multi-words. Other work extends these measures by considering multi-words as a whole string, but the problem is mainly due to word permutations (e.g. "University of Vermont" is much more similar to "University of Virginia" than is "Virginia, University"). Another choice to extend these measures to deal with multi-words is to pair up words by selecting the minimum edit distance between each pair of words, as is proposed by French [40]. After that, all these distances are added up to calculate the final edit distance for the multi-words. With regard to the proposal by Spasic and Ananiadou [41], the authors propose a measure of contextual similarity for biomedical terms. They represent the context of each term as a sequence of syntactic elements annotated with biomedical information retrieved from an ontology. The sequences of contextual elements may be matched approximately by edit distance, defined as the minimal cost incurred by the changes (including insertion, deletion and replacement) needed to transform one sequence into the other.

In this section, we describe our extension proposal of DEx to consider comparisons between single words and multi-words in an efficient manner, in order to make it useful in the aforementioned restricted domains. The proposal is more elaborated than the previous one [40], and differs from the one by Spasic and Ananiadou [41] because we do not require additional knowledge.

The Multi-words Distance (DM: "Distancia para Multipalabras" in Equa-

tion 2) is based on calculating DEx for each word in the multi-words stored in the KOS. The proposed algorithm for calculating DM is described as follows. An example is also provided (comparison between words "afrecho de trigo" and "afrechillo") for the sake of clarity.

1. The words to be compared, whether they are multi-words or not, are tokenized with the aim of analyzing each term as an independent entity.

2. A matrix is created and filled in with the analyzed words in accordance with Edit Distance, as in the DEx algorithm (see Sect. 3.1), the only difference being that the element to be compared is the value of DEx (calculated according to the Equation 1 in Sect. 3.1) between words rather than an exact match (see the following step in this algorithmic sequence). Table 5 shows the matrix for the example after simultaneously carrying out both this and the subsequent step of the algorithm.

3. The similarity between words is determined. To do this, a dynamic threshold is established by means of the DEx algorithm. This threshold is dynamic because it depends on the length of the Operation Chain (OC) of each pair of compared words. Therefore, this threshold is calculated for each pair of compared words and it will have the same value in those comparisons that have an OC of an equal length. The steps to calculate this threshold are:

   3.1 The Middle of the previously generated Operation Chain (Middle-OC) of compared words is found as follows:

$$\text{Middle-OC} = \left\{ \begin{array}{ll} \frac{length(OC)}{2} + 1 & \text{if } length(OC) = \text{even number} \\ \frac{length(OC)+1}{2} & \text{if } length(OC) = \text{odd number} \end{array} \right\}$$

Table 3: Example of obtaining the OC between “afrecho” and “afrechillo”.

		a	f	r	e	c	h	i	l	l	o
	0	1	2	3	4	5	6	7	8	9	10
a	1	0	1	2	3	4	5	6	7	8	9
f	2	1	0	1	2	3	4	5	6	7	8
r	3	2	1	0	1	2	3	4	5	6	7
e	4	3	2	1	0	1	2	3	4	5	6
c	5	4	3	2	1	0	1	2	3	4	5
h	6	5	4	3	2	1	0	1	2	3	4
o	7	6	5	4	3	2	1	1	2	3	**3**

		O	O	O	O	O	O	I	I	I	O

For example, in the comparison between “afrecho” and “afrechillo” the middle of the OC “OOOOOOIIIO” is the sixth position (see the end of Table 3).

3.2 A new OC with the same length as the original OC is created. This new OC has No-Operation (i.e. “O”) in each of its positions, except in the middle of the OC (found in the previous step). Instead, the position of the middle of the OC has the Insertion operation (“I”). According to the previous example, the new OC is “OOOOOIOOOO”.

3.3 The dynamic threshold is established from the DEx evaluation of the new OC (previously obtained) with the weight of the least important character in the dictionary (calculated from the ranking of characters in the dictionary; according to the set of weights P previously detailed in Sect. 3.1, the character 3 with a weight of

14 would be the least important character). For the example, the threshold is 0.028 (see Table 4).

Bearing these issues in mind, the similarity between compared words can then be decided. If DEx distance is not greater than the threshold, it is assumed that tokens are similar; otherwise they are likely to be different. For the example, this step is shown in Table 4, where it can be observed that DEx for comparing "afrecho" and "afrechillo" (0.026 as previously calculated in Sect. 3.1) is lower than the threshold of 0.028, and both words are thus considered to be similar.

4. Another matrix is simultaneously created to store the values of DEx between the tokens compared in the previous step.

Table 4: Example of calculating similarity between strings.

Pivot	**Analyzed word**	**Threshold**	**DEx**	**Similarity**
afrecho	afrechillo	0.028	0.026	YES ($DEx < Threshold$)
de	afrechillo	0.052	0.908	NO ($DEx > Threshold$)
trigo	afrechillo	0.052	0.909	NO ($DEx > Threshold$)

Table 5: Matrix for comparing the pivot word "afrecho de trigo" and the word "afrechillo".

		afrechillo
	0	
afrecho	1	0
de	2	1
trigo	3	2

5. The operation chain from the previously obtained matrix is determined (see Table 5). For the example this is "ODD", since there is no op-

eration ("O") between "afrecho" and "afrechillo" and there are two deletion operations ("DD") of the words "de" and "trigo".

6. The operation chain in the Equation DM is evaluated, which is defined according to Equation 2

$$DM = \sum_{i=1}^{L}(V(O_i) \cdot CP + CD \cdot DE_{x_i})$$

$$where: \quad V(O_i) = \begin{cases} 0 \; if \; O_i = o \\ 1 \; if \; O_i \neq o \end{cases} \tag{2}$$

Parameters and constants are described as follows:

O_i**:** Operation chain in the position i (O-No operation, I-Insertion, D-Deletion, S-Substitution).

$V(O_i)$**:** vector of operations. This has values of 0 if the operation chain in i is "no operation", and otherwise, 1.

L**:** Length of operation chain (O_i).

$FP = 2^{-1(i+1)}$**:** this is a penalty factor used to give a weight to the position in which the transformation between compared words occurs. Thanks to the exponential behavior of this factor, it is possible to impose a more rigorous penalization on those transformations that occur further to the left in the multi-word. Besides, as FP aims at establishing values without overlap, the CP and CD depend on FP (as shown next).

CP**:** Penalty factor for the position of the word within the string or multi-word (CP =0.95 of FP). This value was empirically obtained from several experiments.

CD**:** Penalty factor for DEx between compared words (CD =0.05 of FP). This value was empirically obtained from several experiments. CD is used to establish an order between comparisons that are similar.

DE_{x_i}**:** DEx, according to Equation 1, between words in position i within multi-words.

The evaluation for the example is shown in Table 6. Also, according to DM, the value of the distance between "afrecho de trigo" and "afrechillo" is 0.37; and both words are thus considered to be similar.

Table 6: Example of step 7 of DM: evaluating Equation 2.

i	$V_{(O_i)}$	FP	CP	CD	$V(O_i) \cdot CP$	DE_{x_i}	$CD \cdot DE_{x_i}$	$(V(O_i) \cdot CP + CD \cdot DE_{x_i})$
0	0	0.5	0.475	0.025	0	0.026	0.00065	0.00065
1	1	0.25	0.2375	0.0125	0.2375	0.908	0.01135	0.24885
2	1	0.125	0.11875	0.00625	0.11875	0.909	0.00568	0.12443
$DM = \sum_{i=1}^{L} (V(O_i) \cdot CP + CD \cdot DE_{x_i}) =$								0.3739

Finally, it is worth pointing out that the higher the value of DM is, the greater the distance between words is, and they are thus less similar.

In the following section we describe our proposal to obtain noisy-tolerance IR systems in restricted domains in which we have used the DM algorithm. Some comparisons of this and other distance measures have been made in Sect. 7.1.

5. Adding noise-tolerance to an IR system

In restricted-domain IR systems, the most important terms are those related to the domain. Therefore, if noise affects these terms then the precision of the restricted-domain IR systems decreases. These systems must consequently be especially aware of noise in restricted-domain terms appearing in the corpus. To this aim, our approach compares terms in a KOS with corpus and query terms by means of the *DM* algorithm. An overview of the approach is depicted in Fig. 1.

The approach has three main stages: one at indexing time (in the *offline* phase) and the other two at query time (in *online* phase). In the first stage (see Sect. 5.1 for more details) the *DM* distances between each indexed and KOS terms are calculated. In the second stage (see Sect. 5.2), *DM* distance between each query and KOS terms are calculated in order to obtain the terminology vectors of those terms. Next, in the last stage (see Sect. 5.3), the correspondence between query and indexed terms are defined. Finally, the methodology for applying query expansion to IR systems, using the relations obtained by the *DM* algorithm, is explained in Sect. 5.4 and Sect. 5.5.

5.1. Obtaining terminological vector for each indexed term

A terminology vector can be defined as follows: let T be the set of n terms from the KOS mapped with the *DM* algorithm. $t_r \in T$ denotes the term r in the set of terms. The terminology vector that represents the term t_s is then defined as the vector $V_{t_s} = [(t_1, w_1), (t_2, w_2), ..., (t_n, w_n)]$ where w_r denotes the distance between t_s and t_r.

In order to obtain the terminology vector for each indexed term, we first

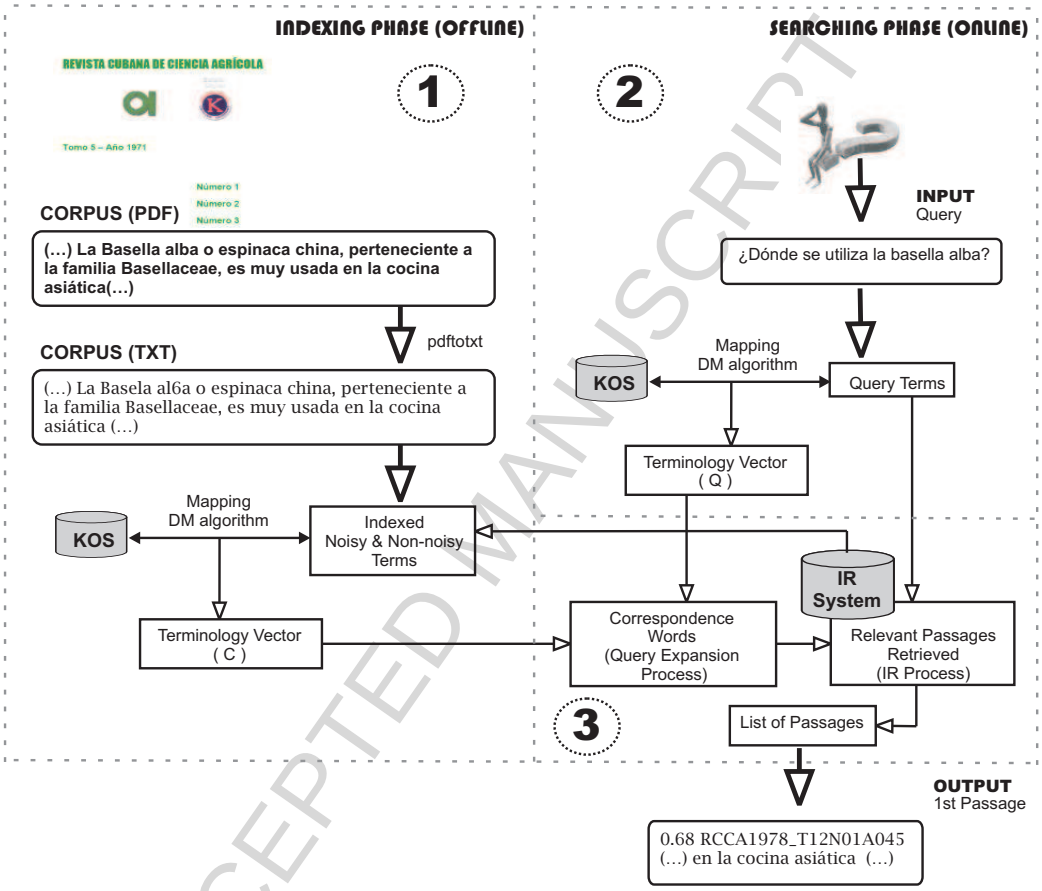


Figure 1: Overview of the authors' approach for adding noise-tolerance to an IR system.

convert the corpus into flat files and we tokenize the corpus terms. We next index these terms using an IR system without considering whether these terms are noisy. Every indexed term is then mapped to the terms in the domain-specific KOS by using the DM algorithm previously detailed in Sect. 4. For each indexed term, its mapped terms and their corresponding distances are kept in terminology vectors (C).

To illustrate our proposal, we take the text fragment with noise shown in Fig. 1: "La Basela al6a o espinaca china, perteneciente a la familia Basel-

laceae, es muy usada en la cocina asiática"; whose version without noise would be: "La Basella alba o espinaca china, perteneciente a la familia Basellaceae, es muy usada en la cocina asiática". In this example, we can appreciate the OCR errors in: "basela" with the omission of "l" character and "al6a" with the substitution of "b" for "6" character, where the difficulties to fix this kind of noise are due to the multi-word situation with both words affected by noise, and the fact that the first word, "Basela" is also a correct word in the language. By applying the method we will obtain the vector C represented as $C_{basela\_al6a} = [(basella\_alba, 0.241)$, $(baselaceas$, $0.248)$, $(basella, 0.249)$, $(basella\_rubra, 0.249)$, ...].

5.2. Obtaining terminological vectors of each relevant query term

In the second stage, we obtain the terminology vector for each query term. In order to do this, we follow steps similar to those of the previous stage: (i) the query terms are tokenized, (ii) relevant query terms are selected, (iii) these terms are mapped with related KOS terms by using DM algorithm, and (iv) their corresponding terminology vectors (Q) are obtained.

For example, if we consider the query *"¿Dónde se utiliza la basella alba?"* (*Where is the basella alba used?*), the system will take the multi-word "basella alba" and the word "basella" as relevant terms for the IR system and our approach will return the terminology vectors Q represented as $Q_{basella\_alba} = [(basella\_alba, 0)$, $(basella$, $0.247)$, $(basellaceae, 0.248)$, $(baselaceas$, $0.249)$, $(basella\_rubra, 0.249)$, ...] and $Q_{basella} = [(basella, 0)$, $(basellaceae, 0)$, $(baselaceas, 0)$, $(basella\_alba, 0.247)$, $(basella\_rubra$, $0.247)$, ...].

5.3. Obtaining the mapping between terms

In the last stage, and using the vectors Q and C obtained in previous stages, we attempt to match the word correspondences between both vectors. The criterion used to determine these correspondences is:

Whether the term i represented by the terminology vector $Q_i \in Q$ and the term j represented by the vector $C_j \in C$ fulfill the following conditions:

- $i \neq j$, and
- *more than a number NT of terms exist in Q_i which are also contained in C_j and whose distances are lower than a given maximum threshold*, or
- *at least a number NT of terms exist in Q_i which are also contained in C_j and whose distances are all equal to 0.*

These threshold and NT values depend on the application domain and must be defined empirically. In our case study, the values of the threshold and NT are 0.37 and 3, respectively. This maximum threshold for the DM algorithm optimizes the precision and recall in the comparison of simple words and multi-words (as shown at Sect. 7.1.3).

Once we have detected terminology vectors with corresponding terms that fulfill the previous rules, we expand the query by using the terms related to the vector C_j because terms related to the vector Q_i are the query terms themselves. Therefore, if the vector Q_i related to the query term i matches several $C_1, C_2, ..., C_j$ corpus vectors related to the corpus terms $1, 2, ..., j$, the query term i will be expanded by the corpus terms $1, 2, ..., j$.

By following the examples shown in Sect. 5.1 and Sect. 5.2 we can observe that the terms "*basella alba*", "*basella*","*basellaceae*", "*baselaceas*" and "*basella rubra*" appear in both vectors $Q_{basella\_alba}$ and C_{basela_al6a}. As $basela\_al6a \neq basella\_alba$ and both vectors contain more than three equal terms (i.e. the DM distance between each of them is lower than a given maximum threshold 0.37), the first and second rules are fulfilled and the word "*basela al6a*" is therefore used to expand the query term "*basella alba*". In the same way the corpus noisy term "*basela al6a* can be used to expand the query term "*basella*".

We have thus succeeded in relating the query terms "*basella alba*" or "*basella*" with the noisy term "*basela al6a*" stored in the corpus and the IR system has been able to find the passage with the correct answer shown in Fig. 1 as system output.

5.4. IR systems

In order to observe the independence of our method with regard to the IR system, for the experiments we have used two IR engines: JIRS[5] and Lucene[6].

5.4.1. JIRS

JAVA Information Retrieval System (JIRS) is a Passage Retrieval engine which is particularly suited to QA tasks, and was developed by Gómez [42]. Its purpose is to find pieces of text (passages) in each document which are organized according to the probability of containing the question structures

[5]http://sourceforge.net/projects/jirs/

[6]http://lucene.apache.org/

rather than just returning a set of relevant documents. To that end, JIRS uses the query structure itself, and attempts to find an equal or similar expression in the documents. The greater the similarity of the structure between the query and the passage is, the higher the passage relevance will be. For example, if the query is "*What is searched for by using an intravaginal sponge system made of polyurethane?*", JIRS will try to find a passage with the expression "*To set a farm-level, a suitable method is searched for by using an intravaginal sponge system made of polyurethane*". In this ideal example, both query and passage contain the same structures and, in these cases, the answer is frequently extremely similar.

JIRS is able to find query structures in a large document collection quickly and efficiently by using different n-gram models [43]. In that paper, several of these n-grams models were compared. However, in the work presented herein, we have only used the *Distance Density n-gram* model since Gómez et al. [43] proved to be the best during the experiments. The *Distance Density n-gram* model is based on searching for the heaviest n-grams (i.e., those with the greatest term weight) but taking into account the distance between them.

As was mentioned above, JIRS is an IR system which returns passages rather than documents. The size of these passages is defined by a number of sentences. This passage division method was discovered by Llopis [44], who demonstrated that this kind of passage extraction gives a better performance than those based on window or paragraph algorithms. Gómez et al. [43] carried out several experiments to determine the best size and overlap of the passages for QA tasks, and they concluded that an overlapped passage of 3 sentences has a good relation between answers found and passage size. An

overlapped passage of 3 sentences signifies that the first passage is composed of the first, the second and the third sentences of the document, whereas the second passage is formed of the second, the third and the fourth sentence, and so on. We have used JIRS because almost all the QA systems which took part in the Cross Language Evaluation Forum (CLEF) in 2005[7] and used this IR system as a search engine, obtained the first positions in the ranking [45].

5.4.2. Lucene

Lucene [46] is a high-performance, scalable, open source search engine written completely in JAVA[8]. It is available as part of the Apache Jakarta project[9]. This system measures the similarity between the query and the document using the dot product and the *tf-idf* [47] for the weight term. However, we adapt Lucene in order to convert it into a passage retrieval system instead of into a document retrieval engine, because this kind of application works better in QA tasks [44]. In order to obtain passages from documents, we split the documents in small pieces of text with a given number of sentences as we explain in Sect. 5.4.1.

Lucene combines two IR techniques: Boolean model [48] and vector space model [47]. For the second, the cosine similarity [47] is used, but with some modifications which are explained in their documentation.

We used Lucene because it is the most frequently used IR system in QA systems. Nevertheless, this system is based on document retrieval and it

[7]http://clef-qa.fbk.eu/2005/

[8]http://www.oracle.com/technetwork/java/index.html

[9]http://jakarta.apache.org

was therefore necessary to adapt it to obtain passages. This was done by building small overlapped documents of 3 sentences with the same criterion as explained above.

5.5. Applying query expansion to IR systems

After obtaining the mapping between query and corpus terms we expanded the queries in order to add noise-tolerance to IR systems (see Fig. 1). In this section we explain the process of query expansion in the IR systems (i.e. JIRS and Lucene) that are described above (at the Sect. 5.4) and will be used in our experiments.

JIRS has the advantage that it permits an input query composed of relations of query terms and expanded terms. This means that we can associate each query term with its expanded terms obtained with the DM algorithm, and JIRS takes this information into account in order to assign term weights. Nonetheless, it is necessary to adapt the expanded query to Lucene. Since Lucene does not accept this kind of query with related terms, we used two different approaches, the first of which involved using the OR boolean operator, and the second of which involved defining a query with a combination of OR and AND boolean operations. In the latter approach, we forced at least each query term or its expanded terms to appear in the passage. For example, if the query is *¿Cuáles son los metabolitos principales que vienen del tracto digestivo?* (*What are the main metabolites which appear in the gastrointestinal tract?*), the DM algorithm returns the terms which appear in Table 7.

With the set of original terms and its expanded terms, a new query for Lucene is formed with the following syntax: ($term_1$ OR $expanded_{1,1}$ OR

Table 7: Example of query expansion using DM algorithm

	Query terms				
DM expansion	metabolitos	principales	vienen	tracto	digestivo
	metabolic			tract	digest
	metabolica				digesta
	metabolicas				digestibilidad
	metabolico				digestibilidades
	...				...

$expanded_{1,2}$ OR ... OR $expanded_{1,m_1}$) AND ($term_2$ OR $expanded_{2,1}$ OR $expanded_{2,2}$ OR ... OR $expanded_{2,m_2}$) AND ... AND ($term_n$ OR $expanded_{n,1}$ OR $expanded_{n,2}$ OR ... OR $expanded_{n,m_n}$), where $term_i$ is the i-esim term of the query and $expanded_{i,j}$ is the j-esim expanded term from the original term i. This signifies that it is obligatory for each query term or an expanded term to appear in the passage. Following the previous example, the query for Lucene should be: (metabolites OR metabolic OR ...) AND principales AND vienen AND (tracto OR tract) AND (digestivo OR digest OR ...).

5.6. Performance of our noise-tolerance approach

As our system is a prototype developed with the unique purpose of carrying out research and evaluation, the algorithm's implementation could be greatly improved. In the indexing phase, the prototype is based on the comparison of each term in the document collection with each KOS term. If we consider that when comparing the strings X and Y, the temporal cost of DM, DEx and Levenshtein algorithms can be approximated as $O(|X| \cdot |Y|)$ where $|X|$ and $|Y|$ represent the lengths of such strings, then for a corpus of N_C words and a KOS of N_K terms, this cost is $O(N_C \cdot N_K)$ multiplied by the edit distance cost: $O(\overline{L_C} \cdot \overline{L_K})$, where $\overline{L_C}$ and $\overline{L_K}$ are the average length of

corpus and KOS terms, respectively. Moreover, since the average length of corpus and the KOS terms do not depend on their sizes, the complexity of the algorithm based on our approach is $O(N_C \cdot N_K)$.

Although the prototype implies a high temporal cost, this type of approach are thought to be used for small restricted-domain corpora, which do not contain redundancy, and are processed at indexing phase (offline). Therefore, the temporal cost is not decisive in our approach. Fortunately, there are approximation search algorithms, based on the Levenshtein distance, which can be adapted to DM and DEx, thus considerably reducing the cost of these operations. This adaptation could be applied in order to obtain higher performance and scalable systems. Some of these algorithms are analyzed and compared by Mihov and Schulz [49] using different and bigger KOS than Agrovoc KOS, and it is shown that it is feasible to search any word in real time, providing a response time of a few milliseconds within an acceptable memory cost using a 1.6 GHz Pentium IV machine.

6. Experimental resources

This section describes the resources used in order to perform the experiments shown in the following section.

6.1. RCCA corpus

The corpus used in the experiments is the Cuban Journal of Agricultural Science (RCCA: "Revista Cubana de Ciencia Agrícola")[10]. This was created

[10]http://www.ica.inf.cu/productos/rcca/

in both English and Spanish in 1966. To date the RCCA Journal has published 43 volumes, each with an average of three or four numbers, making a total of 140 numbers and 2000 articles (28.65 MB as PDF files). This journal comprises topics related to agricultural science, such as Pastures and Forages or Animal Science. In this paper, we use the Spanish part of this journal as corpus because this is the open-access part.

RCCA journal accomplishes the three conditions stated by Minock [11] for being a restricted domain corpus: (i) it is *circumscribed* because user queries are only related to agricultural science, (ii) it is *complex* since it contains a plethora of agricultural-specific terms, and (iii) it is *practical* because all agricultural researchers should be interested in it.

Importantly, as the RCCA has been publishing papers since 1966, many of the papers have been digitalized, which may imply even more noise in the corpus when they are converted to flat files: 1479 papers published between 1967-2000 were scanned and stored as PDF files which require OCR tools to extract the text documents, which represents a significant percentage (73.95% of total). We can therefore claim that the experiments have been carried out with *real* noisy data rather than us having to introduce simulated data corruption. This makes our case study highly representative in order to evaluate our approach.

6.1.1. Removing noise from a small piece of the corpus

In order to determine the upper bound of performance of the baseline IR system, noise has been manually removed from the 150 files of the corpus that contain the answers to test queries.

6.2. Test queries

A total of 329 queries were used in the experiments. They were formulated in natural language rather than as a list of keywords because they will be used in QA systems in future work (see Sect. 8). These queries are in Spanish to permit interaction with the Spanish corpus of the RCCA journal. Some sample queries are: *¿Qué es la necrosis cerebrocortical?* (*What is the cerebrocortical necrosis?*) or *¿Qué produce la cytophaga?* (*What is produced by the cytophaga?*). These queries were elicited by interviewing agricultural domain experts from the RCCA journal. From a total of the 329 test queries, the number of queries not affected by noise is representative (231), which allows us to conclude that our noisy-tolerance approach does not decrease the performance of the IR system.

6.2.1. Inserting noise to the queries

In order to test our approach with noise queries, noise was introduced into the collection of queries. The steps taken to accomplish this were: (i) the collection of 329 queries was printed; (ii) the printed documents were scanned with 100 dpi to obtain PDF files that contained the queries; and finally (iii) the OCR tools which were necessary to obtain the queries in flat text were applied. Upon carrying out this process, 134 queries appeared to be affected (representing 41% of the collection of queries). Of these affected queries, around 25% of their relevant terms had been damaged by the noise which had entered when applying the OCR tool.

6.3. AGROVOC thesaurus

In our case study of the agricultural domain, we used the AGROVOC thesaurus as the KOS. The AGROVOC thesaurus has a total number of 16700 descriptors, and 10758 non-descriptors, which are specific descriptors and terminological terms used in agricultural science. AGROVOC is a multilingual structured controlled vocabulary used for indexing and retrieving data in agricultural information systems.

7. Experiments

This section provides a detailed description of the experiments carried out. It is worth noting that, in spite of the fact that the experiments were executed in the Spanish language, our approach is equally feasible for other languages because both DM algorithm and our noise-tolerance scheme (see Fig. 1) are flexible enough to allow adaptation to different languages.

Firstly some preliminary experiments to check the effectiveness of DEx and DM algorithms were conducted. Therefore, the experiments that measure the benefits of DEx are presented in Sect. 7.1.1 and Sect. 7.1.2, whereas those of DM are presented in Sect. 7.1.3. Afterwards, other experiments were carried out as follows:

1. The corpus with and without noise were used with the aim of obtaining the lower and upper bound of performance of the baseline IR systems, respectively (whose results are presented in Sect. 7.2.1).

2. Our approach (see Fig. 1) was applied with the noisy corpus to add noise-tolerance to JIRS and Lucene IR systems (see Sect. 7.2.2).

3. It (see Fig. 1) was applied with the noisy queries previously created at Sect. 6.2.1 to test the noise-tolerance proposal (see Sect. 7.2.3).

The experiments in Sect. 7.2.1, 7.2.2 and 7.2.3 are designed with the goal to measure the benefits of the application of our proposal. Therefore, the experiment in Sect. 7.2.1 is carried out with the queries without the expansion proposed in this paper. However, for the experiments in Sect. 7.2.2 and Sect. 7.2.3, we expanded the queries using our approach.

7.1. Preliminary experiments with edit distance algorithms

In this section we describe preliminary experiments that compare several algorithms with regard to their performance when calculating edit distances among word inflexions and when they are applied to multi-words. With the first two experiments (see Sect. 7.1.1 and Sect. 7.1.2), we aim to demonstrate the general idea that DEx, unlike other algorithms, does not penalize those words with the same stem but different inflectional endings.

In scanned texts, words frequently appear together (without any separation) as a result of scan errors, and the edit distances are not usually able to deal with this. This kind of error leads to the creation of words which we have called multi-words because they are composed of two or more words that do not normally appear together. The third preliminary experiments (see Sect. 7.1.3) analyze the suitability of the adaptation of DEx (the *DM* algorithm), in order to work with these multi-words by means of a simple example.

7.1.1. Experiments with verb conjugations

Spanish, like other Romance languages, is characterized by a high number of word inflections, especially the verbs that are highly inflected terms. We have therefore selected a set of infinitive verbs and their conjugations in order to evaluate different edit distance algorithms in the most difficult scenario for an edit distance. The experiments were focused on proving that DEx can assign the shorter distance to word inflexions (e.g. between the lemma and the various tenses of a verb).

In order to carry out these experiments, three random verbs were taken: "enseñar" (to learn), "fabricar" (to manufacture) and "cabalgar" (to ride -a horse-), together with their respective 64, 66 and 55 conjugations. We have measured edit distances between these verbs and their corresponding conjugations by applying different algorithms: Jaro, Jaro-Winkler, Levenshtein, Needleman-Wunsch and DEx (which our approach is based on). Edit distances were calculated by using SimMetrics[11], an open source library written in JAVA and supported by the University of Sheffield[12]. Fig. 2 shows the results obtained for the verb "*enseñar*", but the same behavior was observed in the rest of the verbs (see previous work [38]).

As Fig. 2 shows, all distance algorithms penalize verb inflections, but DEx is the one that penalizes less. Although the Jaro based algorithms show satisfactory results for many conjugations, they fail when certain edit operations are involved. Upon analyzing Levenshtein and Needleman-Wunsch algorithms we can conclude that they have similar behavior and their dis-

[11] `http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html`

[12] `http://www.dcs.shef.ac.uk/~sam/stringmetrics.html`

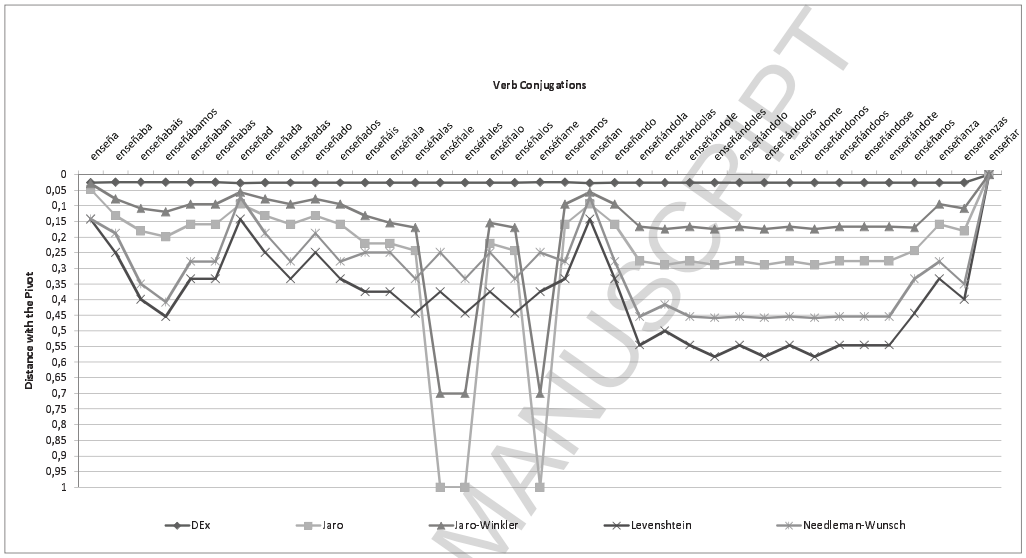


Figure 2: Edit distances between the verb "enseñar" and some of its conjugations

tances depend solely on the length of word endings, of which the longest are penalized to a greater extent. However, DEx algorithm does not overly penalize any of the tense conjugations because it considers word stems rather than word endings.

7.1.2. Experiments with noisy and non-noisy words

The objective of the second experiment was to analyze performance variation of DEx in comparison to different edit distance methods when the terms are affected by noise. By noise we mean both typographic (e.g. *bacteriasviablcs*) and orthographic (e.g. *bacteriaz*) errors and language switch (in our case from Spanish into English)[13]. For this aim, we used the RCCA corpus

[13]For the second experiment we considered the words in English as noisy words, because the pivot word and the corpus used are in Spanish.

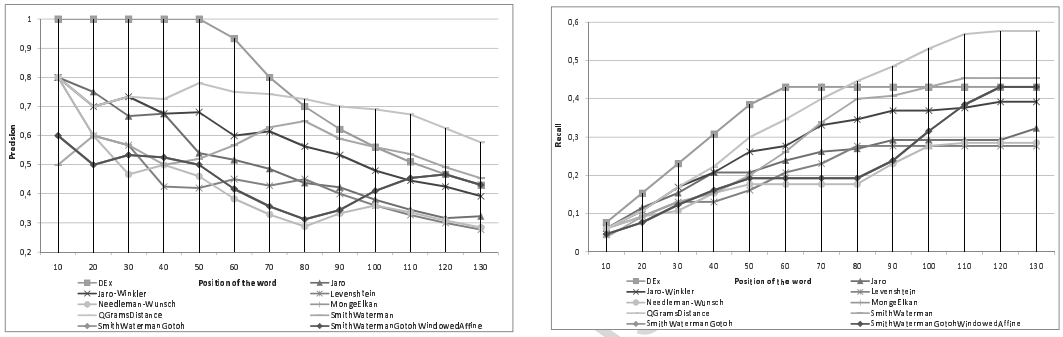
described in Sect. 6.1, as well as a pivot word "*bacterias*" and 130 words related to it, which were selected by a human expert. Words which in some way contained the stem of the pivot word were considered to be related (e.g. "*acto**bacter***", "***bacter**icida*" and "***bacter**oidaceae*). Table 8 shows more examples of terms related to "*bacteria*". In order to model noisy data we introduced some noise into 49 of the 130 related words (some examples are shown in the terms in *italics* in Table 8).

Table 8: Example of rightly related words for "*bacterias*" (noisy words shown in italics)

acetobacter	*bacteriascapacesde*	bacterizados	eubacterium
acetobacterium	*bacteriasviablcs*	*bacterjas*	fibrobacter
achromobacter	*bacteriaz*	*bacteroidacea*	flavobacterium
acinelobacter	bactericida	bacteroidaceae	*forbacteria*
acinetobacter	bacterina	*bacteroidacese*	fosfobacterias

It is worth mentioning that all the distance algorithms used in the experiments are also part of the SimMetrics library mentioned above.

In order to evaluate this experiment, the terms in the RCCA corpus were sorted by each edit distance method. Next, precision and recall were calculated for each edit distance. On the one hand, precision was calculated by the number of rightly related words retrieved divided by the number of retrieved words. On the other hand, recall was calculated by dividing the rightly related words retrieved by the total number of rightly related words, i.e., the 130 words. In Fig. 3(a) we can observe that DEx precision is better than the other algorithms, mainly when the number of words retrieved words is lower than 80. DEx also obtains a perfect precision, 20 points above that of another system when the number of words is 50 or less. QGramsDistance starts with a precision of 80%, like that of Jaro, Jaro-Winkler and Needleman-



(a) Precision. (b) Recall.

Figure 3: Edit distances between noisy and non-noisy words.

Wunsch, but it improves DEx when the number of terms retrieved exceeds 80. The worst systems are SmithWaterman's algorithms, which start with a precision of between 50% and 60%. All the distance measures evaluated (with the exception of DEx) obtained an average of 27 incorrect words.

If we now observe the recall in Fig. 3(b), we can see that DEx is also better with the first retrieved words, and achieve a recall of about 43%. QGramsDistance again improves DEx algorithms, but when 80 words are retrieved its recall reaches about 58%. The next algorithm with most recall is Smith-Waterman with 45%, but its recall in the first retrieved words is worse than the DEx and QGramsDistance methods.

The differences between DEx and these algorithms lie in the fact that the former hardly penalizes those words which contains stem changes, for example, in "***propioni****bacterias*" and other examples with long prefixes. Of the 130 words selected by the human expert, 50% of them had prefixes; this, logically, is detrimental to the precision and recall of DEx but, as is shown in the figures, only after the 50th word. DEx is able to retrieve the words

with suffixes perfectly, but not those with prefixes.

Fig. 4 shows the strong ability of DEx to retrieve words with noise compared with other algorithms.

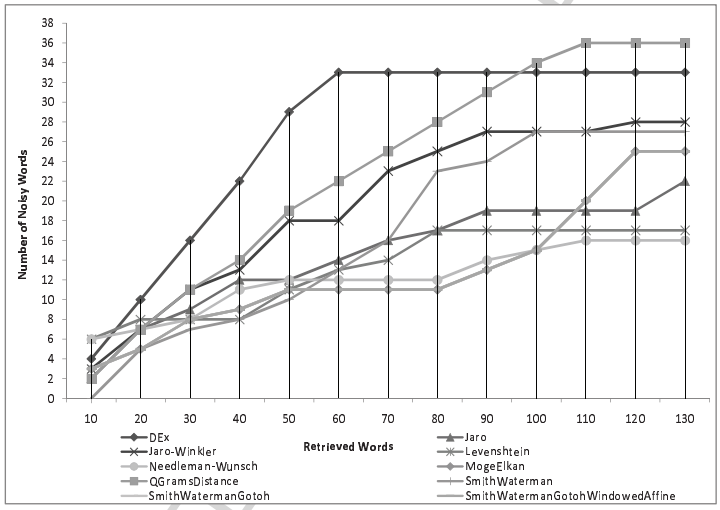
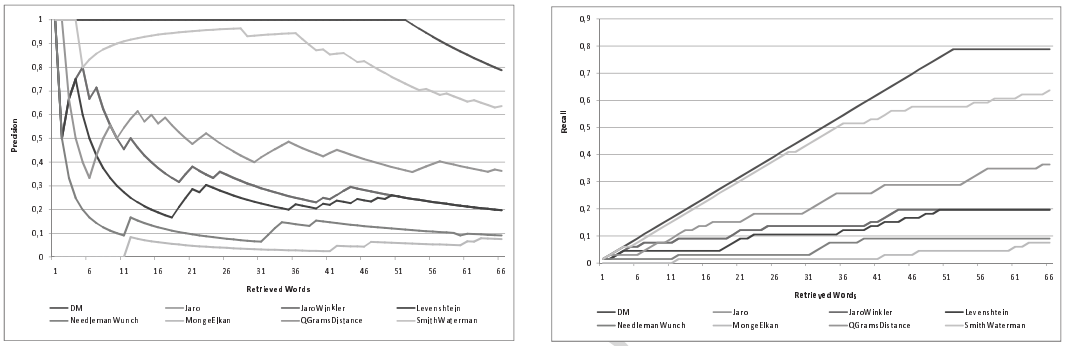


Figure 4: Number of retrieved noisy words using different edit distance algorithms

In general, when we introduce a noisy word as input to the edit distance algorithms, their performance decreases. But once again, DEx demonstrates that it is the best algorithm if the number of retrieved words is not very high. DEx is only exceeded by QGramsDistance when more than 100 words are retrieved.

With these preliminary experiments we have shown some examples of DEx's performance with regard to other edit distance algorithms. It is, of course, necessary to carry out more experiments in order to be certain of our hypothesis regarding the suitable use of DEx algorithm in noisy-tolerance IR systems, but the small-scale tests shown here may give us a slight insight into the final results.



(a) Precision. (b) Recall.

Figure 5: Edit distances between simple words and multi-words.

7.1.3. Experiments to measure the effectiveness of DM distance

The objective of this experiment is to compare the *DM* algorithm with the edit distance methods evaluated in the previous sections in order to show the effectiveness of our method when multi-words are involved. The evaluation corpus consisted of 66 words related to "bacteria", 47 of which were multi-words (e.g. "bacteria coliforme", "bacterias nitrificantes", "bacteria gram positiva", etc.).

Fig. 5(a) shows the precision of the various methods evaluated. In order to consider a word as being similar, for this experiment, a maximum edit distance of 0.37 and a number NT of terms of 3 were used for the *DM* algorithm (as is explained in Sect. 5.3). These values were obtained empirically in the previous work [10]. In this figure we can appreciate that *DM* improves the performance considerably, achieving a precision of 100% in the first 50 retrieved words. However, the other systems are unable to deal with multi-words properly, and their precision is very poor in comparison with previous experiments.

With regard to recall, as Fig. 5(b) shows, the behavior is very similar. DM increases the recall linearly until the 50th retrieved word. If we compare both figures, we can observe that DM retrieves the first 51 words perfectly, but is unable to find the remaining 15 words in the corpus. This is because DM, like DEx, is unable to retrieve words with different prefixes. The problem with other edit distance algorithms is that they do not consider the position of the edit operations. Therefore, when one word appears together with another, without any separation space, the systems define the entire extra word as being noisy, thus leading to a considerable increase in edit distance. These systems work well only when small changes occur at the end of the word or when shortened words are included.

The experiment carried out in this section provides us with an approach concerning the effectiveness of DM algorithms in dealing with simple words and multi-words. In the following sections we focus on DM in order to demonstrate with complete experiments the performance of this algorithm in noisy-tolerance IR systems.

7.2. Experiments for evaluating our noise-tolerance approach

The aim of these experiments is to evaluate the effectiveness of the approach for adding noise-tolerance to an IR system (see Sect. 5).

7.2.1. Experiments for determining bound values to evaluate our noise-tolerance approach

This experiment aims to obtain the maximum and the minimum values of several performance measures when JIRS or Lucene (i.e. the baseline IR systems) are used without using our approach. These values will later be

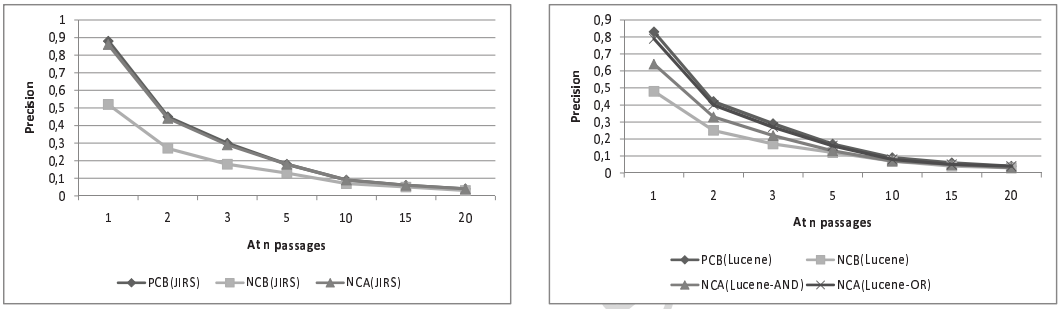
used to show the suitability of our proposal.

In the indexing phase of both IR systems on the entire RCCA corpus, 432,997 passages and 180,460 domain terms were obtained. With regard to the 150 documents that are going to be preprocessed to remove the noise, 6,795 passages and 10,437 domain terms were obtained (1894 of them containing noise, therefore, around 18% of the terms contained noise).

The experiment was conducted by using the 231 queries that were not affected by noise and the 98 queries affected by noise. The total amount of retrieved passages was 6,580 and the number of relevant retrieved passages was only 329. It is worth noting that we decided to return 20 passages per query in order to properly analyze the results and the position of the correct answer.

The first part of the experiment consisted of obtaining the best performance for the IR system (i.e. JIRS or Lucene) by using the 150 documents that had been preprocessed to partially remove noise (henceforth PCB: Preprocessed Corpus and Baseline system). Secondly, the worst performance was obtained by using the noisy corpus (NCB: Noisy Corpus and Baseline system). Both experiments were carried out without our proposal of noise-tolerance. We can conclude that, in order to consider our proposal suitable, the upper and lower bound of relevant retrieved passages for all queries should be between 230 and 309 with JIRS, or between 217 and 291 with Lucene (see Fig. 7(b) when 20 passages per query are returned).

Other results obtained in this experiment for each corpus are shown in Figures 6 and 7. We calculated the following measures: precision, recall, F1 [50], and Mean Reciprocal Rank [51] (MRR). The values in these figures



(a) JIRS. (b) Lucene.

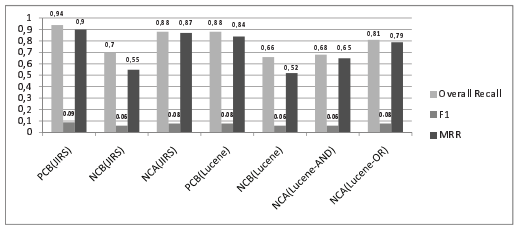
Figure 6: Precision results.

show that the noise greatly affects the results returned by the IR system (e.g. $MRR(PCB) = 0.90$ vs. $MRR(NCB) = 0.55$ with JIRS or $MRR(PCB) = 0.84$ vs. $MRR(NCB) = 0.52$ with Lucene).

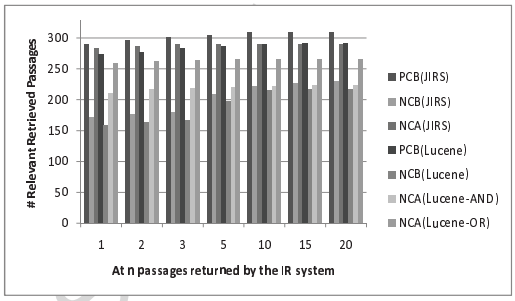
7.2.2. Experiments for evaluating our noise-tolerance approach with noisy corpus

The aim of this experiment is to evaluate the effectiveness of our noise-tolerance approach (see Sect. 5) by comparing the results with the best and worst performance of both IR systems. The expected results must be found between both values, and therefore, the nearer the results are to the best performance, the better our approach will work.

Once terms in the noisy corpus have been indexed with the JIRS and Lucene systems in the previous experiment (as shown in Sect. 7.2.1) and mapped with the Agrovoc KOS by means of our approach (see Sect. 5.3), three executions of our approach were realized in this experiment: (i) using JIRS and the query expansion, (ii) employing the OR Boolean operator in Lucene, and (iii) combining the OR and AND Boolean operations in Lucene



(a) Other Measures.



(b) Relevant Retrieved Passages.

Figure 7: Other results.

(previously explained in Sect. 5.5). The results of this experiment are shown in NCA (Noisy Corpus and our noisy-tolerance Approach) in Figures 6 and 7. These results considerably improve the baseline values obtained in the previous experiment by using the noisy corpus, while they are near the optimal results returned by the IR system when a non-noisy corpus is used. For example, the precision obtained by using both IR systems and our proposal has close values to the results obtained with the non-noisy corpus (see Fig. 6).

It is worth highlighting that the values of F1 in Fig. 7(a) are similar for our approach and for the baseline IR (i.e. both IR systems JIRS and Lucene) with the non-noisy corpus, while the difference in the overall recall is only 0.06 for JIRS and 0.07 for Lucene with OR operator. Moreover, the difference in MRR is 0.03 for JIRS and 0.05 for Lucene with OR operator. An in-depth analysis of all the evaluated measures shows that, in our approach ("NCA"), recall is affected since 19 less relevant passages are retrieved. The main reason is that some noisy terms have no counterparts in AGROVOC owing to the fact that they were too deformed, or that they are not in the thesaurus. However, the weighted harmonic mean (F1) of precision and recall

obtain a similar result in both experiments.

Another important conclusion obtained from analyzing the results of this experiment is that only 3% of the answers became worse by using our approach with regard to baseline results over a non-noisy corpus. Specifically, the correct answers to 10 queries were retrieved in more 'away' positions than in the baseline experiment. We were thus able to measure that our noisy-tolerance approach does not decrease the performance of the IR system (when the noise does not affect the query or the answer to the query).

7.2.3. Experiments for evaluating our noise-tolerance approach with noisy queries

This experiment aims to further evaluate our approach when the noise is found directly in the query that the user asks the IR system. To carry out this experiment, noise was introduced into the collection of queries in an artificial manner (as previously described in Sect 6.2.1). As with the previous experiment, we can now also determine the maximum and minimum values between which the results should range to consider our approach as a valid one.

Fig. 8 shows the results of precision (with regard to the first three passages retrieved), overall recall and MRR for the experiments carried out. Several experiments were carried out with the aim of: (i) defining the minimum value to be obtained with our approach for it to be considered valid, using queries with introduced noise (NQB: Noisy Queries and Baseline system); (ii) defining the maximum value which our approach should be near to, for it to be considered valid, by using original queries without noise (CQB: Non-noisy Queries and Baseline system, the results being equal to those from PCB of

the previous experiment); (iii) measuring the effectiveness of our approach by using noisy queries (NQA: Noisy Queries and our noisy-tolerance Approach). From the analysis of Fig. 8, it can be observed that the results of our approach (NQA) are very near to optimal values (CQB) for all measures. We can therefore conclude that our approach is also valid for adding fault-tolerance to an IR system when noise appears in the query.

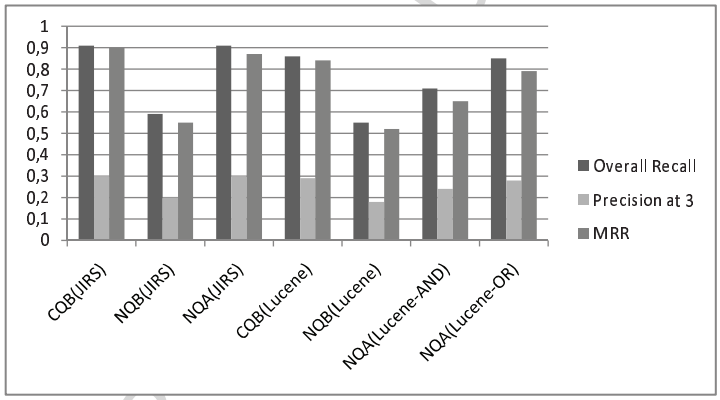


Figure 8: Different Measures over noisy queries.

8. Conclusions and Future Work

Real-world data are inherently noisy, thus signifying that techniques to process noise are crucial in IR systems if useful and actionable results are to be obtained [1]. Owing to the huge amount of redundancy inherent in vast open-domain corpora, they are insensitive to noise. Nevertheless, corpora in restricted domains are usually rather smaller, and therefore have little or no redundancy. IR systems using small and non-redundant restricted-domain corpora are consequently likely to fail. In order to overcome this problem,

in this paper, we show how noise tolerance can be added to the retrieval process.

Our approach contributes to the state-of-the-art in the following issues: (i) It is based on the hypothesis that the most important terms are those related to the domain, which we obtain from the KOS and the corpora. These KOS terms and the terms in the corpora are compared in order to detect noisy terms. (ii) The comparisons between terms are carried out with the use of DM, a new algorithm that adapts DEx edit distance (which was also proposed by the authors) to consider multi-words. Both DEx and DM outperforms previous distance algorithms (see experiments in section 7.1). (iii) Our approach deals with all kinds of noise, even that which occurs with noisy terms that also appear in the lexicon as correct terms (e.g. tear vs. fear). (iv) The performance of an IR system is maintained although noisy restricted-domain corpora are used, as shown in experiments in section 7.2.

Future work is focused on extending the authors' proposals: (i) to prove that our noise-tolerance approach is valid for different languages other than Spanish, (ii) to improve the DM algorithm to be used in other Natural Language Processing tasks such as Question Answering.

References

[1] C. A. Knoblock, D. P. Lopresti, S. Roy, L. V. Subramaniam, Special issue on noisy text analytics, IJDAR 10 (3-4) (2007) 127–128.

[2] J. Bourdaillet, J.-G. Ganascia, in: IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data.

[3] L. V. Subramaniam, S. Roy, T. A. Faruquie, S. Negi, A survey of types of text noise and techniques to handle noisy text, in: AND '09: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data, 2009, pp. 115–122.

[4] G. Hodge, Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files, The Digital Library Federation Council on Library and Information Resources, 2000.

[5] H. Jing, D. Lopresti, C. Shih, Summarization of noisy documents: A pilot study, in: D. Radev, S. Teufel (Eds.), Proceedings of the HLT-NAACL 03 Text Summarization Workshop.

[6] D. P. Lopresti, S. Roy, K. Schulz, L. V. Subramaniam, Special issue on noisy text analytics, IJDAR 12 (2009) 139–140.

[7] F. R. Chen, D. S. Bloomberg, Summarization of imaged documents without OCR, Computer Vision and Image Understanding 70 (3) (1998) 307–320.

[8] D. D. Palmer, M. Ostendorf, Improving information extraction by modeling errors in speech recognizer output, in: HLT '01: Proceedings of the first international conference on Human language technology research, 2001, pp. 1–5.

[9] L. Aunimo, O. Heinonen, R. Kuuskoski, J. Makkonen, R. Petit, O. Virtanen, Question Answering system for incomplete and noisy data, in: ECIR, 2003, pp. 193–206.

[10] K. Vila, J. Díaz, A. Fernández, A. Ferrández, An Approach for Adding Noise-Tolerance to Restricted-Domain Information Retrieval, in: C. J. Hopfe, Y. Rezgui, E. Métais, A. D. Preece, H. Li (Eds.), NLDB, Vol. 6177 of Lecture Notes in Computer Science, Springer, 2010, pp. 1–12.

[11] M. Minock, Where are the "killer applications" of restricted domain Question Answering?, in: Proceedings of the IJCAI Workshop on Knowledge Reasoning in Question Answering, Edinburgh, Scotland, 2005, p. 4.

[12] R. A. Baeza-Yates, B. A. Ribeiro-Neto, Modern Information Retrieval, ACM Press / Addison-Wesley, 1999.

[13] A. Vinciarelli, Noisy text categorization, IEEE Trans. Pattern Anal. Mach. Intell. 27 (12) (2005) 1882–1895.

[14] K. Kukich, Techniques for automatically correcting words in text, ACM Comput. Surv. 24 (4) (1992) 377–439.

[15] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Tech. Rep. 8 (1966).

[16] M. Li, M. Zhu, Y. Zhang, M. Zhou, Exploring distributional similarity based models for query spelling correction, in: ACL, 2006.

[17] S. Cucerzan, E. Brill, Spelling correction as an iterative process that exploits the collective knowledge of web users, in: EMNLP, 2004, pp. 293–300.

[18] Q. Chen, M. Li, M. Zhou, Improving query spelling correction using web search results, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 181–189.

[19] J. Vilares, M. Vilares, J. Otero, Managing misspelled queries in IR applications, Information Processing & Management In Press, Corrected Proof.

[20] K. Taghva, J. Borsack, A. Condit, Effects of OCR errors on ranking and feedback using the vector space model, Inf. Proc. and Management 32 (3) (1996) 317–327.

[21] K. Taghva, E. Stofsky, Ocrspell: An interactive spelling correction system for OCR errors in text, Intl. Journal on Document Analysis and Recognition 3 (3) (2001) 125–137.

[22] P. B. Kantor, E. M. Voorhees, The TREC-5 Confusion Track: Comparing retrieval methods for scanned text, Inf. Retr. 2 (2/3) (2000) 165–176.

[23] P. B. Kantor, E. M. Voorhees, Report on the TREC-5 Confusion Track, in: TREC, 1996.

[24] J-Y.Nie, J. Cai, Filtering noisy parallel corpora of web pages, in: IEEE symposium on NLP and Knowledge Engineering, Tucson, 2001, pp. 453–458.

[25] L. Shi, J.-Y. Nie, Filtering or adapting: two strategies to exploit noisy parallel corpora for cross-language Information Retrieval, in: CIKM '06:

Proceedings of the 15th ACM international conference on Information and knowledge management, 2006, pp. 814–815.

[26] K. Imamura, E. Sumita, Bilingual corpus cleaning focusing on translation literality, in: In: 7th International Conference on Spoken Language Processing (ICSLP-2002, 2002, pp. 1713–1716.

[27] S. Khadivi, H. Ney, Automatic filtering of bilingual corpora for statistical Machine Translation, in: NLDB, 2005, pp. 263–274.

[28] W. M. Esser, Fault-tolerant fulltext Information Retrieval in digital multilingual encyclopedias with weighted pattern morphing, in: ECIR, 2004, pp. 338–352.

[29] W. M. Esser, Fault-tolerant fulltext search for large multilingual scientific text corpora, Journal of Digital Information 6 (1) (2004) 1368–7506.

[30] D. Hawking, P. B. Thistlewaite, P. Bailey, ANU/ACSys TREC-5 Experiments, in: TREC, 1996.

[31] X. Tong, C. Zhai, N. Milic-Frayling, D. A. Evans, OCR Correction and Query Expansion for Retrieval on OCR data – CLARIT TREC-5 Confusion Track Report, in: TREC, 1996.

[32] K. B. Ng, D. Loewenstern, C. Basu, H. Hirsh, P. B. Kantor, Data Fusion of Machine-Learning Methods for the TREC-5 Routing Task (and other work), in: TREC, 1996.

[33] B. Manthey, R. Reischuk, The intractability of computing the hamming distance, in: ISAAC, 2003, pp. 88–97.

[34] F. Damerau, A technique for computer detection and correction of spelling errors, Commun. ACM 7 (3) (1964) 171–176.

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

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

[37] W. E. Winkler, The state of record linkage and current research problems, Tech. rep., Statistical Research Division, U.S. Census Bureau, Washington, DC (1999).

[38] A. C. Fernández, J. Díaz, A. Fundora, R. Muñoz, Un algoritmo para la extracción de características lexicográficas en la comparación de palabras, in: IV Convención Científica Internacional de La Universidad De Matanzas CIUM'09, Matanzas, Cuba, 2009.

[39] D. S. Hirschberg, Algorithms for the longest common subsequence problem, J. ACM 24 (4) (1977) 664–675.

[40] J. C. French, A. L. Powell, E. Schulman, Applications of approximate word matching in information retrieval, in: CIKM, 1997, pp. 9–15.

[41] I. Spasic, S. Ananiadou, A flexible measure of contextual similarity for biomedical terms, in: Pacific Symposium on Biocomputing, 2005.

[42] J. M. Gómez, Recuperación de pasajes multilingüe para la búsqueda de respuestas, Phd. thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain (2007).

[43] J. M. Gómez, D. Buscaldi, P. Rosso, E. Sanchis, Jirs: Language-independent passage retrieval system: A comparative study, in: 5th International Conference on Natural Language Processing 2006, Hyderabad, India, 2007.

[44] F. L. Pascual, IR-n: Un sistema de recuperación de información basado en pasajes, Phd. Thesis, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Alicante, Spain (2001).

[45] A. Vallin, B. Magnini, D. Giampiccolo, L. Aunimo, C. Ayache, P. Osenova, A. Peñas, M. de Rijke, B. Sacaleanu, D. Santos, R. Sutcliffe, Overview of the CLEF 2005 Multilingual. Question Answering Track, in: Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum, Vol. 4022 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Viena, Austria, 2006, pp. 307–331.

[46] E. Hatcher, O. Gospodnetic, Lucene in Action (In Action series), Manning Publications Co., Greenwich, CT, USA, 2004.

[47] G. Salton, A. Wong, C. S. Yang, A vector space model for automatic indexing, Commun. ACM 18 (11) (1975) 613–620.

[48] F. W. Lancaster, E. G. Fayen, Information Retrieval: on-line, Melville Pub. Co. Los Angeles, 1973.

[49] S. Mihov, K. U. Schulz, Fast approximate search in large dictionaries, Computational Linguistics 30 (4) (2004) 451–477.

[50] C. J. van Rijsbergen, Information Retrieval, 2nd Edition, Butterworths, London, 1979.

[51] E. M. Voorhess, The TREC-8 Question Answering track report, in: Proceedings of the 8th Text REtrieval Conference (TREC-8), 1999, pp. 77–82.

Appendix A

Pivot: bacteria

#	Relevant words
1	antagonistas bacterianas
2	Antibacterianos
3	antigenos bacterianos
4	Azotobacteriaceae
5	Bacteremia
6	Bacteria
7	bacteria acetic
8	bacteria aerobia
9	bacteria anaerobia
10	bacteria butírica
11	bacteria coliforme
12	bacteria estimul crecimiento planta
13	bacteria fijadora del nitrógeno
14	bacteria generadora de hielo
15	bacteria gram negativa
16	bacteria gram positiva
17	bacteria inmovilizada
18	bacteria maloláctica
19	bacteria mesófila
20	bacteria metanógena
21	bacteria termófila
22	bacterias acidolácticas
23	bacterias acidopropiónicas
24	bacterias antagonistas
25	bacterias del rumen
26	bacterias diazotróficas
27	bacterias entomógenas
28	bacterias entomopatógenas
29	bacterias logradas genéticamente
30	bacterias metanotróficas
31	bacterias modificadas genéticamente
32	bacterias nitrificantes
33	bacterias patógenas
34	Bactericidas
35	Bacterinas
36	Bacteriocinas
37	Bacteriófagos
38	Bacteriología
39	Bacteriólogos
40	Bacteriosis
41	Bacteriostático
42	bacterium coli
43	Bacteroides
44	conteo bacteriano
45	control de bacterias (almacenamien)
46	control de bacterias (desinfección)
47	control de bacterias (enfermedad)
48	Cyanobacteria
49	enfermedades bacterianas
50	enterobacteriaceae
51	esporas bacterianas
52	flora bacteriana
53	fuego bacteriano
54	infecciones por micobacterias
55	insecticidas bacterianos
56	lawsonia (bacteria)
57	Bacteroidaceae
58	Micobacterias
59	plaguicidas bacterianos
60	propiedades antibacterianas
61	proteínas bacterianas
62	proteínas bacterianas (producto)
63	quema bacteriana
64	rhodococcus (bacteria)
65	Rizobacterias
66	toxinas bacterianas

Retrieved Words	Precision	Recall	DM
Bacteria	1	0.015151515	0
Bactericidas	1	0.03030303	0.000355509
Bacterinas	1	0.045454545	0.000357035
Bacteriosis	1	0.060606061	0.000360254
Bacteremia	1	0.075757576	0.000651976
Bacteria butírica	1	0.090909091	0.24379391
Bacteria acetic	1	0.106060606	0.248296519
Bacteria anaerobia	1	0.121212121	0.248296792
Bacteria aerobia	1	0.136363636	0.248296804
Bacteria metanógena	1	0.151515152	0.248424087
Bacterias acidopropiónicas	1	0.166666667	0.248495304
Bacterias antagonistas	1	0.181818182	0.248495574
Bacterias nitrificantes	1	0.196969697	0.248528667
Bacterias metanotróficas	1	0.212121212	0.248622871
Bacteria coliforme	1	0.227272727	0.248633338
Bacterium coli	1	0.242424242	0.248666942
Bacterias diazotróficas	1	0.257575758	0.24870292
Bacteria inmovilizada	1	0.272727273	0.2488367
Bacterias entomopatógenas	1	0.287878788	0.248999091
Bacterias entomógenas	1	0.303030303	0.248999092
Bacterias acidolácticas	1	0.318181818	0.249053723
Bacteria maloláctica	1	0.333333333	0.249335677
Bacteria mesófila	1	0.348484848	0.249353949
Bacteria termófila	1	0.363636364	0.249432302
Bacterias patógenas	1	0.378787879	0.249554605
Bacterias del rumen	1	0.393939394	0.372678173
Bacteria gram positive	1	0.409090909	0.373001145
Bacteria gram negative	1	0.424242424	0.373067892
Bacterias modificadas genéticamente	1	0.439393939	0.373728191
Bacterias logradas genéticamente	1	0.454545455	0.373825589
Bacteria generadora de hielo	1	0.46969697	0.434562287
Bacteria estimul crecimiento planta	1	0.484848485	0.434764505
Bacteria fijadora del nitrógeno	1	0.5	0.435581631
Conteo bacteriano	1	0.515151515	0.475098981
Fuego bacteriano	1	0.53030303	0.475098981
Insecticidas bacterianos	1	0.545454545	0.475098982
Antigenos bacterianos	1	0.560606061	0.475098982
Plaguicidas bacterianos	1	0.575757576	0.475098982
Flora bacteriana	1	0.590909091	0.475098988
Quema bacteriana	1	0.606060606	0.475098988
Proteínas bacterianas	1	0.621212121	0.475098989
Toxinas bacterianas	1	0.636363636	0.475098989
Enfermedades bacterianas	1	0.651515152	0.475098989
Esporas bacterianas	1	0.666666667	0.475098989
Antagonistas bacterianas	1	0.681818182	0.475098989
Bacteriófagos	1	0.696969697	0.475360129
Bacteriólogos	1	0.712121212	0.475360214
Bacteriología	1	0.727272727	0.475360214
Bacteriocinas	1	0.742424242	0.475360222
Bacteriostático	1	0.757575758	0.475360254
Bacteroidaceae	1	0.772727273	0.475649792
Bacteroides	1	0.787878788	0.475649792
Bactrocera	0.981132075	0.787878788	0.477109992
Bactofugación	0.962962963	0.787878788	0.47712015
Bactris	0.945454545	0.787878788	0.477123916
Bactra	0.928571429	0.787878788	0.477123937
Backusella	0.912280702	0.787878788	0.478487248
Baculoviridae	0.896551724	0.787878788	0.478745462
Baculovirus	0.881355932	0.787878788	0.478745462
Bacua	0.866666667	0.787878788	0.478757341
Bacón	0.852459016	0.787878788	0.478757342
Báculo	0.838709677	0.787878788	0.478757377
Bacota	0.825396825	0.787878788	0.478757384
Bacona	0.8125	0.787878788	0.478757384
Bacará	0.8	0.787878788	0.478762369
Bacalao	0.787878788	0.787878788	0.478762446