A Proposal of Automatic Error Correction in Text

Wulfrano A. Luna-Ramírez, Carlos R. Jaimez-González

Departamento de Tecnologías de la Información, Universidad Autónoma Metropolitana – Cuajimalpa, Av. Constituyentes No. 1054, Col. Lomas Altas, C.P. 11950, México, D.F.
{wluna,cjaimez}@correo.cua.uam.mx

Abstract. The great amount of information that can be stored in electronic media is growing up daily. Many of them is got mainly by typing, such as the huge of information obtained from web 2.0 sites; or scaned and processing by an Optical Character Recognition software, like the texts of libraries and goverment offices. Both processes introduce error in texts, so it is difficult to use the data for other purposes than just to read it, i.e. the processing of those texts by other applications like e-learning, learning of languages, electronic tutorials, data minning, information retrieval and even more specialized systems such as tiflologic software, specifically blinded people-oriented applications like automatic reading, where the text would be error free as possible in order to make easier the text to speech task, and so on. In this paper it is showed an application of automatic recognition and correction of ortographic errors in electronic texts. This task is composed of three stages: a) error detection; b) candidate corrections generation; and c) correction -selection of the best candidate. The proposal is based in part of speech text categorization, word similarity, word diccionaries, statistical measures, morphologic analisys and n-grams based language model of Spanish.

Keywords. Automatic error correction, language models, n-grams, word eskeleton, part of speech, text processing, string comparison, edit distance.

1 Introduction

The great amount of electronic texts comes mainly from the internet, where the irruption of social networks and the Web 2.0 has provide us a huge quantity of data in just a little period of time. Another great source of information is the data obtained from the scanning of printed documents by means of an Optical Character Recognition (OCR), used by the libraries and government offices in order to preserve files of historic information. Given this enormous quantity of data, there is a need to apply automatic tools to process, transform and extract this information. Even more, a lot of such text contains typing errors and this makes difficult to keep the processing of such texts by other applications; of course, OCR is another source of errors in texts [1],[2],[3],[6],[7]. So, to retrieve text without orthographic errors is a difficult task.

Additionally, when a software tool is used, the main problem of having text with errors is the difficult to use this information as the input of more specialized and even

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 323–338
Paper Recived 30-09-2012 and Acepted 01-11-2012

useful applications, like the wide set of Natural Language Processing (NLP) tools [2],[11]. Some examples of such applications are: data mining (and of course web mining), information searching and retrieval, document categorization, and those systems related with education (learning of foreign languages), phonetic translation, code and text computer aided edition, and aided to people with disabilities: text to speech conversion, blind people software assistance, accessibility applications, etc. [9],[10].

Fortunately, the same knowledge of the NLP field can help to design some methods to cope with errors in texts, like the linguistic knowledge. In this way, according to the linguistic point of view, there are different levels of text treatment or processing, some of them are: a) orthographic and morphologic level: describes the structure and external features of words, i.e. the way the letters are articulated in certain language; b) syntactic level: describes the paragraphs and the phases, i.e. the word organization within a text, and the grammatical categories of words, (commonly called Part of Speech -POS); c) semantic level: fully related with the meaning of text, takes care of the context where phrases and words appear; d) contextual or pragmatic level: describe the specific use of words, locutions, phrases, etc., in a certain situation (the discursive and temporal use) within a domain. On this way, the use of certain linguistic knowledge is an adequate approach to design automatic applications of text like the error correction task [1],[2].

In this work, it is used knowledge of the morphologic and contextual levels, specifically a POS tagger, and different techniques of word matching, additionally a language model based in n-grams is used [10],[11].

Briefly, the method is based on speech text categorization, word similarity, word dictionaries, statistical measures, and n-grams model of language.

In the next sections it is exposed the proposal. The section 2 describes the process of error correction and a brief classification of text errors is given; the section 3 describes the method of error correction; after that, some implementation details, experiments, results and discussion are exposed in section 4; section 5 includes conclusion and future work.

2 The Process of Error Correction in Electronic Text

The Error Correction in Electronic Text is performed in three stages [1],[6],[7]:

1. **Error detection**: It is oriented to text revision in order to identify the chain of characters from the text (words and other marks) as part of a given language, typically by means of a dictionary comparison, or a grammatical structure (morphological, syntactical or semantic).
2. **Candidate correction generation**: detects some possibilities of correction to a given error.
3. **Correction**: it is the selection of a specific candidate correction and the substitution of it in the text.

In **interactive error correction** the system corrector performs the first and the second stages in an autonomous way, and left to the user the final stage, where the

decision of which is the real correction is taken. In the other hand, the **automatic error correction**, the system does the three stages without the need of the user final decision [1].

As can be inferred, the automatic error correction is necessary to PLN applications where the full processing is performed with no user intervention.

Even more, the correction can be divided in two categories [1]: a) **isolated word error** correction: where the three stages described before are performed just word by word; b) **the contextual error correction**: where can be observed the context of the word, i.e. the phrase where it is used.

On the other hand, there is a brief classification of errors [1,7] that can be founded in text (obtained from humans –typing- or by machines –mainly OCR methods). Some error can derivate in linguistic mistakes by the accidental generation of invalid words within the language, or can be match with another valid word but not the correct one. The error classes could be part each other, but in order to understand the nature of error it is useful to try one classification, which is depicted in **Figure 1**: a) typographical: contains wrong characters in the string or word; b) grammatical: violate the articulation rules of a certain language; c) cognitive: the origin of the errors is the lack of knowledge of orthographic rules; d) phonetical: they are wrong representations of a given linguistic utterance (a phonetic chunk of language), those are the worst case of error, because generate a great word malformation and the semantic knowledge implied is high.

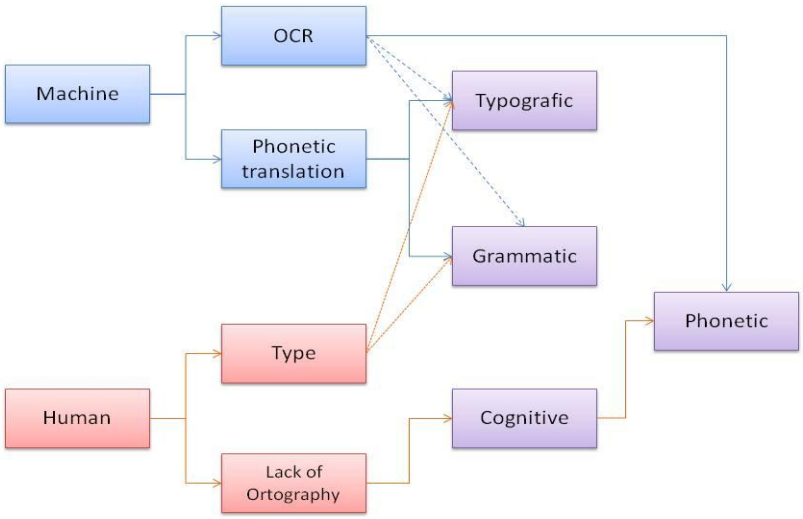


Figure 1. A classification of error in text according to their origin: human being or computer systems. One error can originate different cases of linguistic mistake, as is depicted,

for example an OCR error can be derivate in a phonetic error.

From the linguistic knowledge, those different types of errors can be organized accordingly with the language level related: a) morphologic; b) syntactic; c) semantic; d) speech structure; and e) pragmatic.

The types of errors are used to design techniques which can be applied to the stages of error correction in texts. One case of error that has been mainly treated by PLN applications are word error, which are described in the following subsection given the importance for the automatic or semiautomatic error correction methods that had been tried [1].

2.1 Word level errors

The level word errors can be treated relatively easy by a computer program, but the propagation of the mistake accordingly with the linguistic levels needs to be coped with techniques which imply knowledge about the other levels of the language. The most successful application use morphological and syntactic knowledge, but the superior levels when are incorporated in them originate domain dependent applications.

The two main sources of electronic texts (human or OCR) introduce certain classes of errors that had been afforded by different algorithms, like the edit distance (called Levenshtein distance) for the morphological errors; and n-gram based comparisons, for the syntactic and (in a very restricted sense) semantic errors [1]:

- **Insertion**: from the addition of one character in the string.
- **Duplication**: the doubling of one character in the string.
- **Deletion**: to omit one character in the string.
- **Substitution**: to replace one character by other.
- **Transposition**: where two characters are swapped.
- **Segmentation**: when two strings are formed from just one word.
- **Union**: when two words originate one string.

The segmentation errors are the most frequent in texts from OCR, on the other hand, the human being commonly produce union errors when typing.

2.2 N-grams

In order to verify if a sequence of linguistics elements (words, letters, phrases, etc.) within a text is valid for a given language, it is common use models of languages which represent the constraints of a certain combination of linguistics elements in order to build right structures in language [1],[11],[12].

One model of languages is n-grams, a statistical model of sequences of linguistic elements: typically n word or letters where n >= 2. It is based in the estimation of sequences probability calculated from a text corpus called the **transition probability**, it can give a prediction of the n next element from the n previous according to the frequency with they appear in corpus. N-grams model uses the history based Markov supposition: a linguistic element is affected by the local context, so the previous text

affects the future text [1],[11]. Some examples of n-grams or letters and words are showed in **Figure 2**.

Letters N-gram order	**N-gram example**
monogram	**{**i,n,f,o,r,m,a,t,i,c,s**}**
bigram	**{**in,nf,fo,or,rm,ma,at,ti,ic,cs**}**
trigram	**{**inf,nfo,for,orm,rma,mat,ati,tic,ics**}**

Words N-gram order	**N-gram example**
monogram	**{**computing,is,not,easy**}**
bigram	**{**computing,is**}** {is,not**}** {not,easy**}**
trigram	**{**computing,is,not**}** {is,not,easy**}**

Figure 2. The first table shows n-gram examples of the word *informatics*, and the second contains n-gram examples of the phrase: *computing is not easy*. Where n = {1,2,3}

N-grams had been used in error detection and the candidate generation stages of the error correction task [1]. In the proposal presented in this paper lexicons of bigram and trigram were used.

In the next sections the proposal of automatic error correction and the implementation are exposed.

3 The Automatic Error Correction Proposal

The error correction can be focused to the isolated words or taking account the local context of them. The proposal presented in this paper (POS-Tagged Automatic Error Correction -PAEC) is oriented to automatic error correction of the chain of words in a paragraph, and eventually in the full text [14]. It is based on POS text categorization, word similarity comparison, word dictionaries, statistical measures, and n-grams language models.

Some techniques used in automatic detection, candidate generation and selection of the correction use probability and morphology of words. The proposal adds to those techniques a POS-tagging process for the sake to augmenting the linguistic knowledge in the full process, i.e. uses the morphological and syntactical information present in the text under revision.

The proposal uses the following resources (texts written in Spanish): 1) a plain text corpora (CT), 2) a POS-tagged corpora (CA) with ten grammatical categories defined: verbs, nouns, conjunctions, idioms, articles, adjectives, adverbs, pronouns, interjections and miscellaneous; 3) a text corpora with inserted errors (CE), from a OCR and random deletions, insertions or substitutions; 4) a set of word lexicons: nine of each POS defined, and one of words from all grammatical categories; 5) a n-gram model of language (bigrams and trigrams); and 6) a n-gram model of language of POS-tags, which identify the n-gram occurrence of certain POS-tags (bigrams and trigrams).

The PAEC proposal is showed in **Figure 3**. It is composed of the following modules for to analyze a text:

1. **Preprocessing**: the abbreviation words are expanded, upper case and punctuation symbols are extracted, and the resulting text is separated in sentences and words.
2. **POS-tagging**. The text is POS-tagged in order to identify the grammatical category they belong. The POS-tagger used is an implementation of TBL POS-tagger [4], trained which semiautomatic annotated Spanish text.
3. **Word extraction**: it separates the words and the POS-tagged related, in order to recover the original text, but identifying the grammatical category of words. Produce a list of word for being analyzed.
4. **Contextual error detection**: this task makes a list of possible wrong words in text. It is carried out in two phases:
    a. Seeks each word from the text in their corresponding POS-lexicon.
    b. An analysis of bigrams that can be formed with the words from the text for identifying the abnormal combinations, because it can discover the presence of an error in text.

The process is showed in the **Algorithm 1**, as follows:

```
Input:
     P: List of words to be corrected
     LN: Lexicon of language model (n-grams)
Output:
     PE:  List of words identified as errors
Var:
     k: It represents the index of P word it is being reviewed in each iteration.
     P_k: It is the k member of P
     |P|: Total number of words in the list P
     LP: Word lexicon
     pa: It is the word is being reviewed in each iteration (P_k)
     ant: previous word of pa
     post: next word of pa
     flag: It indicates if a bigram of pa has been found
      <I>: It is an inserted pseudo-word at begin of each line
    Functions:
    add(l,e): add the e element to the list l
    Begin
1.  k = 0
2.  While k < |P|  do
3.   ant = P_k-1
4.   pa = P_k
5.   post = P_k+1
6.   If pa != <I>
7.       If pa not in LP then
8.          add(PE,pa)
```

```
9.       else
10.      If [ant,pa] not in LN then
11.         If ant not in LP then
12.            add(PE,ant)
13.         else
14.            flag = 1
15.         If [pa,post] not in LN  then
16.                If post not in LP then
17.                   add(PE,post)
18.                else
19.                   If post not in PE
20.                        flag = 2
21.         If flag = 2 then
22.                add(PE,pa)
23.         If flag = 1 then
24.                If ant not in PE then
25.                   add(PE,pa)
26.   k = k + 1
27.   Return PE
      End
```

Algorithm 1. Contextual Error Detection. This algorithm receives the list of words to be corrected and the lexicons where the words are searching. It generate the words identified as errors, this is the input for the next stages of error correction.

5. **Potential corrections generation**: taking in count the POS-tag of the potential errors detected in the previous modules, it makes a candidate generation of corrections based in n-grams and morphological comparisons (skeleton and edit distance), seeking in the right lexicon according to the grammatical category of words.
6. **Correction**: it selects the best potential correction from the candidate words, this is the responsible for taking the right one. This selection is based in: a) n-gram analysis for identify the most probable combination of those candidates and the local context of word; b) minimum edit distance of words and candidates; c) skeleton comparisons; and d) size of words and word skeletons.
7. **Postprocessing**: once the correction has been done in text, the upper case, abbreviations and punctuation symbols are reintroduced and the final text is saved.

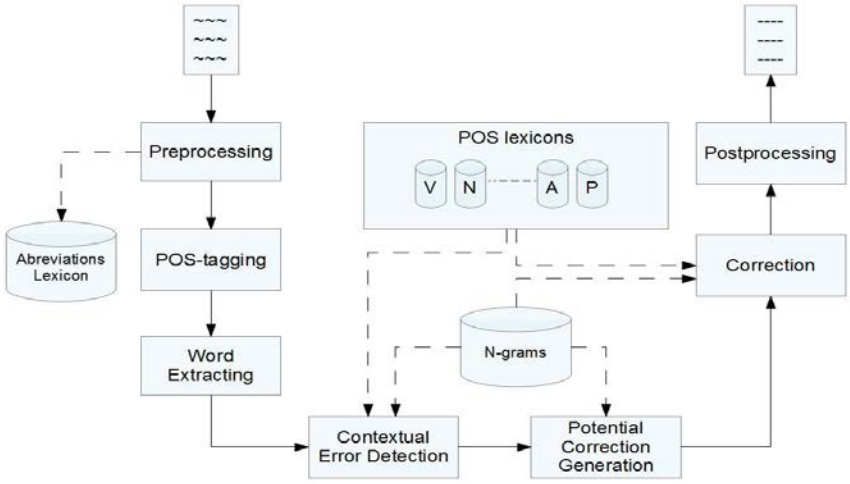


Figure 3. The PAEC proposal. The automatic error correction method is based in the identification of POS of the text and the model of language based in n-grams.

In the following section are discussed the implementation matters, and the experiments carried out in order to verify de efficacy of PAEC proposal.

4 Implementation, experiments and results

The PAEC proposal was implemented using the PERL programming language, because it is relatively easy to construct text analyzer functions, string matching, and lexicon searching [14].

In order to test the PAEC proposal of error correction an alternative method of correction was implemented Morphological Automatic Error Correction (MAEC). This method does not include a phase of POS-tagging for the sake of represent a more traditional way to do the error correction in text, i.e. it contains less linguistic knowledge, specifically morphologic and syntactic information, so a framework of comparison can be established. The MAEC method is depicted in **Figure 4**.

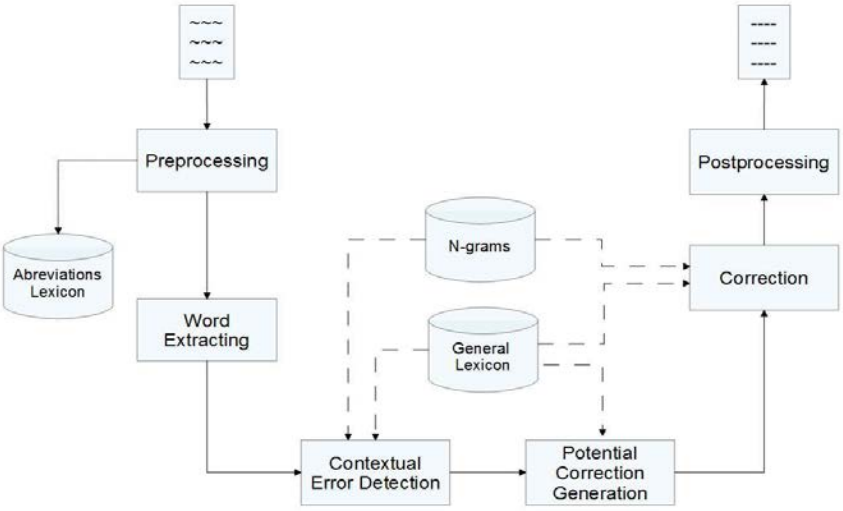


Figure 4. The Morphologic Automatic Error Correction (MAEC) method, it uses a n-grams model of language and a General Lexicon for making the three stages of text error correction. It was used for testing the PAEC proposal of error correction.

4.1 Data description

As it was mentioned in the previous section, many linguistic resources were used to implement the PAEC and MAEC methods of error correction in texts. They are described in the **Tables 1**, **2**, **3**, **4, 5** and **6**.

Table 1. Plain Text Corpus characteristics. The texts were extracted from different Web sites written in Spanish from the following domains: Essay, Politics, Literature and Philosophy.

Language	Spanish
Format	ASCII
Files	497
Text lines	137,103
Words	2,877,834
Characters	18,075,061

Table 2. The POS-tagging manually annotated text corpus characteristics. This corpus was used for the initial training of the POS-tagger which is an iterative and supervised process, where results in a bigger (automatically) annotated corpus for being used in the error correction method. In the first stage it was used 0.1509% of the plain text corpus.

Language	Spanish

Format	ASCII
Files	1
Text lines	243
Words	4,346

In the **Table 3** is described the set of testing files, which contains a lot of errors, one set is obtained from an OCR (Hewlett Packard, ScanJet 3200c, their features can be seen in the **Table 4**) process and the other one was generated according to the following process (for simulating typing errors):

1. The file is formatted in order to keep 10 words per line.
2. One word of each line is randomly selected
3. From each word it is randomly selected one character
4. In the position selected is randomly inserted one error (insertion, deletion or substitution)
5. The modified word is reinserted in its corresponding position in the file.

So, it tis gotten one random error each ten lines.

Table 3. Text Corpus with errors used for testing the error correction methods. The *Errors 1* column shows the automatically inserted errors. The *Errors 2* refers to the error from the OCR process.

Texts	Lines	Strings	Characters	Errors 1	Errors 2
1	15	79	587	7	5
2	17	92	634	7	8
3	16	126	831	13	2
4	11	147	1068	14	9
5	14	150	985	15	4
6	19	260	1770	25	28
7	17	310	2061	30	10
8	20	358	2045	55	52
9	17	591	3800	116	29
10	19	1153	6774	41	15
Total	**165**	**3,266**	**20,555**	**323**	**162**

Table 4. Features of the scanner and the OCR software used to generate the *Error 2* set of files.

Scanner	HP ScanJet 3200c
ORC sofware	HP Precisionscan LT
Resolution	600 x 1200 dpi
Format of output	Plain text (.txt) or riched text (.rtf)

The set of lexicons are showed in the **Table 5**. The General Lexicon is used by the MAEC method, while the other lexicons (POS-based) are used by the proposal PAEC. The contents of each lexicon were taken from the Dictionary of Usual Spanish [12], and the structure of them is given by the orthographic form or each word and

their n-grams of letters. Word between one to two letters has just their monogram, but words of four letter or more has either bigram and trigram.

Table 5. The features of the lexicons used by the error correction methods. A = words total number. B = larger size of word. C = smaller size of word. D = media of size of words.

Lexicon	A	B	C	D
General	131,648	42	1	17
Verbs	10,206	30	2	11
Prepositions	32	13	1	17
Pronouns	83	8	2	5
Interjections	26	7	2	5
Conjunctions	38	20	1	8
Articles	9	3	2	3
Adjectives	5,508	26	2	14
Adverbs	1,516	23	2	12
Nouns	14,517	25	1	12

Table 6. Features of the N-grams based model of language: larger, smaller and media size of just bigram are showed.

Bigrams	566,521
Larger frequency	25,542
smaller frequency	1
Media frequency	4

4.2 Experiment Description

The experiment carried out for testing the MAEC and PAEC methods of error correction in text is composed of two evaluations:

- **Simulation of typing errors** (errors automatically inserted): the purpose was to identify the performance of insertion, deletion and substitution errors.
- **OCR errors in text**: this is a life real set of evaluation, the files comes from an OCR method after being scanned.

Additionally, for the PAEC proposal, one more test was made: manually POS-tagged text was tested in order to obtain an ideal tagging, and verify the efficacy of the method by means to avoid the possibility of getting some deviation because the precision of the automatic POS-tagger.

The following accounts were made in the texts from both methods:

C: corrected words. Real errors detected and corrected properly.

E: detected errors. Total of errors identifies by the methods.

e: non detected errors. Total of errors not identified by the methods.

I: introduced errors. Wrong substituted words due to the methods identify some false errors.

F: false error correction. The candidate correction selected for correcting the error detected was wrong, so the error persist.

o: original errors. They are the original errors presents in text.

On the other hand, the rate or errors was calculated using the following formula [66]:

$$c = p - i / p$$

Where c: rate of error correction; p: words of text; and i: errors of text (wrong words).

4.3 Results of experiment

The results of the testing of MAEC method of error correction are showed in the **Table 7** and the results of PAEC method are showed in **Table 8**. The results of the third experiment (manually POS-tagged files) are showed in **Table 9**.

Table 7. The results of the automatic error correction based in n-grams and morphologic analysis MAEC. The first row is the result of the typing simulation set of testing, while the second row is the result of the texts from the OCR.

Set of testing	p	o	i	C	E	e	I	F	c
Errors 1	3266	323	140	202	311	31	19	90	0.957
Errors 2	3185	162	104	86	180	10	38	56	0.967

Table 8. The results of the automatic error correction based in n-grams and morphologic analysis PAEC. The first row is the result of the typing simulation set of testing, while the second row is the result of the texts from the OCR.

Set of testing	p	o	i	C	E	e	I	F	C
Errors 1	3266	323	370	112	474	28	167	180	0.886
Errors 2	3185	162	229	84	304	8	152	70	0.928

Table 9. The results of the automatic error correction based in n-grams and morphologic analysis PAEC. *Errors 3* row is the test performed with manual POS-tagged files that come from the simulated typing errors, while the *Errors 4* are files that come from an OCR process.

Set of testing	p	o	i	C	E	e	I	F	c
Errors 3	3266	323	133	217	238	26	27	80	**0.959**
Errors 4	3266	162	74	104	186	8	33	45	**0.977**

4.4 Discussion

As can be seen in the results tables, in the experiments on the sets of files *Errors 1* and *Errors 2* the MAEC method is better than PAEC proposal. These results are discussed as follows.

The MAEC method has a high rate of no detected errors (*e*), this affects mainly to the selection of errors because it introduces new errors (*I*) originated by errors that originate valid Spanish words, as a consequence the wrong word is not identified as error but belongs to a n-gram with low frequency; so, another word would be treated as an error.

The most mismatched words are words of one or two letters: y, a, e, la, el, lo, no; because when a deletion occurs they disappears or become in valid words difficult to identify and the introduction or error is possible. On this way, the number *F* is high too, because the n-gram model can't offers information when an error appears in a low frequency n-gram, so when it is weighted by the module of selection of candidate for making the correction there is a low possibility to select the right one.

On the other hand, the PAEC method was affected by the POS-tagging in two aspects:

- In error detection: when a word is taken as an error but the POS-tag is not so precise, then it is searched in the wrong dictionaries; so, *I* rate is increased.
- In potential correction generation: as in error detection, the candidates are generated from a possible wrong dictionary. So, the candidate can be different from the right one.

Finally, in **Table 6** can be seen the result from the POS-tagged files. As can be appreciated, the results are improved with respect to both previous tests. On this way, the PAEC proposal improves it performance given these facts:

- In error detection: now it is possible to identify wrong words that originates no valid words in Spanish, because can be matched in the right dictionary according to their POS-tag. Additionally, the errors that originate valid words are detected because the POS-tag indicate a low frequency n-gram: a wrong word belongs to a different grammatical category due to the mistake, and it is identified.
- In potential correction generation: the candidate generation is performed by its morphologic features, the n-gram probability and its grammatical category adding by this means the morphologic and syntactic knowledge.
- In correction: due to the correct detection the right candidate is selected in most of the times, so the correction is right.

5 Conclusions and Future Work

According to the results presented in the previous section the PAEC method is better in error correction when a precise POS-tagging is performed on the text because the morphologic and syntactic knowledge is properly added to the process, so the rationale is the right one but there are some improvement to do; if it is not the case, there are a lot of mistakes in the entire correction process.

Additionally, in order to improve the performance of the PAEC method it is required to do:

- To explore the use of syntactic n-grams (sn-grams), which are an improved form of n-grams, constructed using paths in syntactic trees in order to reduce the noise given to presence of many arbitrary terms in the n-gram. The sn-grams can be applied to the three stages of the error correction process to calculate POS sn-grams and word sn-grams [13].

- To improve the precision of the POS-tagger:
  - To train the algorithm in a more intensive way, adding more annotated text to star the learning of tagging rules.
  - To develop an alternative way to tagging words not identified and increase the lexical files of the tagger.
  - To use more than one tagger in order to improve the rate of correct tagging by the weighting the results of them.
  - To assign more than one POS-tag to words, accordingly to the probability of tags.
- To improve the detection of errors for searching the potential errors in more than one grammatical category lexicon given the assignation of more than one POS-tag.
- To refine the lexicons in two ways:
  - Increase the words they contain, in order to extend its lexical coverage.
  - By means of a lemmatization process, so the index of word would be more flexible because it can contains no only the complete orthographic form.

As a final commentary is the method can be tested on real world errors and a larger amount of texts. They can be taken from:

- Contents of Web 2.0 pages.
- Government pages of complains, administrative or legal processes and opinion pages.
- A bank of OCR-processed documents from libraries and government offices.

References

1. Kukich K. (1992). Techniques for automatically correcting words in text. ACM Computing Survey. Vol. 24, pp. 377--439.
2. Jurafsky D. & Martin J. H. (2000). Speech and language processing. An introduction to natural language processing, computational linguistics and speech recognition. USA: Prentice-Hall.
3. Jones M. P. & Martin J. (1997). Contextual spelling correction using latent semantic analysis. In Proceedings of the Fifth Conference on Applied Natural Language Processing. (pp. 166--173). USA: Washington DC.
4. Brill E. (1995). Unsupervised learning of disambiguation rules for part of speech tagging. In D. Yarowsky & K. Church (Eds.), Proceedings of the third ACL Workshop on Very Large Corpora (WVLC-95). (pp. 1--13). Somertes, New Jersey: Association for Computational Linguistics.
5. Jin R., Hauptmann A. G., & Xiang Z. C. (2003, January). A content-based probabilistic correction model for ocr document retrieval. In T. Kanungo, E. H. Barney Smith, H. Jianying, P. B. Kantor (Eds.) Document Recognition and Retrieval X (Proceedings of SPIE/IS&T). (pp. 128--135). Sta. Clara, California: The Society for Imaging Science and Technology.
6. Klein, S. T., & Kopel, M. (2002, august). A voting system for automatic OCR correction. In K. J\"arvelin (General Chair), Information Retrieval and OCR: From Converting Con-

tent to Grasping Meaning. Workshop conducted at the 25th ACM SIGIR Conference, 2002, Tampere, Finland.
7. Taghva, K., & Stofsky, E. (2001). OCRSpell: An interactive spelling correction system for OCR errors in text. International Journal on Document Analysis and Recognition, 3(3), 125--137.
8. Sproat, R., & Olive, J. (1998). Text-to-speech synthesis. In V. K. Madisetti, & D. B. Williams (Eds.), The digital signal processing handbook (pp. 46-1--46-11). Boca Raton, Florida: CRC Press.
9. Alvarez Cabán, J. M. (2000). Alternatives for access and use of Spanish language assistive technology equipment by individuals with visual disabilities. In California State University Northridge (Comp.), Proceedings of the Technology and Persons with Disabilities Conference, 2000. Los Ángeles: California State University Northridge. Retrieved from http://www.csun.edu/cod/conf/2000/proceedings/0187Alvarez_Caban.html
10. Moreno Sandoval, A. (1998). Lingüística Computacional. Madrid: Síntesis.
11. Manning C. D. & Schütze H. (2001). Foundations of statistical natural language processing. Cambridge, Massachusetts: The MIT Press.
12. Lara Luis Fernando (s.f.). *Diccionario del Español Usual en México*. Biblioteca Virtual Miguel de Cervantes. Available in : http://bib.cervantesvirtual.com/servlet/SirveObras/06811650999196173088968/index.htm?na=26041
13. Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. Syntactic Dependency-based N-grams as Classification Features. LNAI 7629, 2012, pp. 1-11.
14. Luna Ramírez Wulfrano Arturo & Barrón Machado Jorge. (2004). Mejoras al reconocimiento óptico de caracteres y a la correccion de textos electronicos en los sistemas de lectura automatica de texto. Bsc. Thesis. Supervisor: Esmeralda Uraga Serratos. Classification: 001 00623 L1 2004. 366 pps. http://bcct.unam.mx/web/tesiunam.htm & http://oreon.dgbiblio.unam.mx:8991/F/YMJ36QMYP4YQCSJFJHG7QS5S22MP3K9VREJM6RAP6QSSUYANAG-37691?func=service&doc_number=000334454&line_number=0007&service_type=TAG"

Recognizing Textual Entailment with Similarity Metrics

Miguel Rios[1] and Alexander Gelbukh[2]

[1] University of Wolverhampton,
Research Group in Computational Linguistics,
Stafford Street, Wolverhampton, WV1 1SB, UK
M.Rios@wlv.ac.uk
[2] Center for Computing Research,
National Polytechnic Institute,
Mexico City, Mexico
www.gelbukh.com

Abstract. We present a system for the Recognizing Textual Entailment task, based on various similarity metrics, namely (i) string-based metrics, (ii) chunk-based metric, (iii) named entities-based metric, and (iv) shallow semantic metric. We propose the chunk-based and named entities-based metrics to address limitations of previous syntactic and semantic-based metrics. We add the scores of the metrics as features for a machine learning algorithm. Then, we compare our results with related work. The performance of our system is comparable with the average performance of the Recognizing Textual Entailment challenges systems, though lower than that of the best existing methods. However, unlike more sophisticated methods, our method uses only a small number of simple features.

1 Introduction

The Recognizing Textual Entailment (RTE) task consists in deciding, given two textual expressions, whether the meaning of one of them, called Hypothesis (H), is entailed by the meaning of the other one, called Text (T) [5]. The RTE Challenge is a generic task which addresses common semantic inference needs across Natural Language Processing (NLP) applications.

In order to address the task of RTE, different methods have been proposed. Most of these methods rely on machine learning (ML) algorithms. For example, a baseline method proposed by Mehdad and Magnini [9] consists in measuring the word overlap between the Text and Hypothesis; the word overlap is the number of words shared between the two textual expressions. Their method is organized into three main steps: (i) pre-processing: all T–H pairs are tokenized and lemmatized; (ii) computing of the word overlap; (iii) building a binary classifier. An overlap threshold is computed over the training data, and the test data is classified based on the learned threshold. If the word overlap score is greater than the threshold, then the entailment decision is TRUE (there is entailment), otherwise

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 339–349
Paper Recived and Acepted UK, Mexico

it is FALSE (there is no entailment). The motivation behind this paradigm is that a T–H pair with a strong similarity score has good chances to represent an entailment relation. Different types of similarity metrics are applied over the T–H pair in order to extract features and to train a classifier.

Similarity metrics that deal with semantics usually use information from ontologies or semantic representations given by parsers [2]. However, the comparison between texts is done by matching the semantic labels, and not by matching the content of those units.

In this work we describe an RTE system based on various similarity metrics. In addition, we propose new similarity metrics based on different representations of text for RTE that are: (i) chunks and (ii) Named Entities. The goal of the introduction of these new features is to address limitations of previous syntactic- and semantic-based metrics. We add the scores of the new metrics along with simple string-based similarity metrics and a shallow-semantic-based metric [11] as features for a machine learning method for RTE. Then, we compare our results with related work on RTE. The performance of our system is comparable with the average performance of the RTE challenges, though it is lower than that of the best known methods.

In the remainder of this paper we discuss the related work (Section 2), describe our RTE system (Section 3) and compare its performance with previous work (Section 4). Finally, we give conclusions and suggest some future work (Section 5).

2 Related Work

Burchardt et al. [2] introduced new features for RTE that involve deep linguistic analysis and shallow word overlap measure. Their method consists of three steps: first, they represent the T–H pair with the Frame Semantics (FS) and Lexical Functional Grammars (LFG) formalisms; this representation is similar to the Semantic Role Labeling. Then, they calculate a similarity score based on matching the LFG graphs, and finally make a statistical entailment decision. They used the RTE-2 and RTE-3 datasets as training data, and extracted 47 features from the deep and the shallow overlap. These features consist of combinations of predicates overlaps, grammatical functions match, and lexical overlaps.

The methods that use Semantic Role Labeling (SRL) for RTE use the annotation provided by a semantic parser to measure the similarity between texts. However, they only measure the similarity in terms of how many labels the two texts share (overlaps) and not in termos of the content marked with those labels.

Delmonte et al. [8] introduced semantic-mismatch features, such as locations, discourse markers, quantifiers, and antonyms. Their entailment decisions are based on applying rewards and penalties over the semantic similarity and shallow similarity scores. Later, Delmonte et al. [6] participated in the RTE-2 challenge with an enhanced version of their previous system. Their new system uses new features based on heuristics, such as Augmented Head Dependency Structures, grammatical relations, negations, and modal verbs.

Roth and Sammons [12] used semantic logical inferences for RTE, where the representation method is a Bag-of-Lexical-Items (BoLI). The BoLI relies in word overlap. It states that the entailment relation holds if the overlap score is above a certain threshold. An extended set of stopwords is used to select the most important concepts for the BoLI, such as auxiliary verbs, articles, exclamations, discourse markers, and words in WordNet. Also, in order to recognize relations in the T–H pairs, the system checks matchings between SRLs, and then applies a series of transformations over the semantic representations to make easier to determine the entailment. Their system uses the following transformation operations:

- *annotate*, which make some implicit property of the meaning of the sentence explicit;
- *simplify* and *transform*, which remove or alter some section of the text T in order to improve annotation accuracy or make it more similar to H;
- *compare*, which compares some elements of the two members of the entailment pair and assigns a score that correlates to how successfully those elements of the H can be subsumed by the T.

3 Experimental Design

The RTE task can be seen as a binary classification task, where the entailment relations are the classes. Then the RTE benchmark datasets can be used to train a classifier [4].

Our RTE system is based on a supervised machine learning algorithm. We train the machine learning algorithm with similarity scores computed over the T–H pairs extracted from different classes of metrics described below.

With these metrics we build a vector of similarity scores used as features to train a machine learning algorithm. We use the development datasets from the RTE 1 to 3 benchmark to train different ML algorithms, using their implementations from the WEKA toolset[3] without any parameter optimization. Then, we test the models with a tenfold cross-validation over the development datasets to decide which algorithm to use for the comparison against related work over the test datasets.

The metrics we used as as follows.

3.1 Lexical Metrics

We use the following string-based similarity metrics: precision, recall, and F_1:

$$\text{precision}(T, H) = \frac{|T \bigcap H|}{|H|} \tag{1}$$

$$\text{recall}(T, H) = \frac{|T \bigcap H|}{|T|} \tag{2}$$

[3] http://www.cs.waikato.ac.nz/ml/weka/

$$F_1(T,H) = 2 \times \frac{\text{precision}(T,H) \times \text{recall}(T,H)}{\text{precision}(T,H) + \text{recall}(T,H)} \tag{3}$$

As input for the metrics we use a bag-of-words (BoW) representation of the T–H pairs. However, we only use content words to compute the similarity score in the T–H pairs.

3.2 Chunking

Shallow parsing (or chunking) consists in tagging a text with syntactically correlated parts. This alternative to full parsing is more efficient and more robust. Chunks are non-overlapping regions of text that are sequences of constituents that form a group with a grammatical role (e.g. noun group). The motivation for introducing a chunking similarity metric consists in that a T–H pair with a similar syntax structure can hold an entailment relation. The chunking feature is defined as the average of the number of similar chunks (in the same order) in the T–H pair:

$$\text{chunking}(T,H) = \frac{1}{m} \sum_{n=1}^{m} simChunk(t_n, h_n), \tag{4}$$

where m is the number of chunks in T, t_n is the n-th chunk tag and content in the same order, and $simChunk(t_n, h_n) = 1$ if the content and annotation of the chunk are the same, and 0.5 if the content of the chunk is different but the chunk tag is still the same.

The following example shows how the chunking metric works. Consider:

T: *Along with chipmaker Intel , the companies include Sony Corp. , Microsoft Corp. , NNP Co. , IBM Corp. , Gateway Inc. and Nokia Corp.*
H: *Along with chip maker Intel , the companies include Sony , Microsoft , NNP , International Business Machines , Gateway , Nokia and others.*

First, for each chunk, this metric compares and scores the content of the tag: whether it is the same chunk group and whether it is the same order of chunks. Table 1 shows how this metric scores each chunk for the previous example.

Finally, the chunking metric (4) computes the individual scores and gives a final score of $chunking(T,H) = 0.64$ for this example.

3.3 Named Entities

Named Entity Recognition (NER) is a task that identifies and classifies parts of a text into predefined classes such as *names of persons*, *organizations*, *locations*, *expressions of times*, *quantities*, *monetary values*, *percentages*, etc. For example, from the text: "Acme Corp bought a new...", *Acme Corp* is identified as a named entity and classified as an *organization*.

Table 1. Example of partial scores given by the chunking metric

Tag	Content	Tag	Content	Score
PP	*Along*	PP	*Along*	1
PP	*with*	PP	*with*	1
NP	*chipmaker Intel*	NP	*chip maker Intel*	0.5
NP	*the companies*	NP	*the companies*	1
VP	*include*	VP	*include*	1
NP	*Sony Corp.*	NP	*Sony*	0.5
NP	*Microsoft Corp.*	NP	*Microsoft*	0.5
NP	*IBM Corp.*	NP	*International Business Machines*	0.5
NP	*Gateway Inc.*	NP	*Gateway*	0.5
NP	*Nokia Corp.*	NP	*Nokia and others.*	0.5

The motivation of a similarity measure based on NER is that the participants in H should be the same as those in T, and H should not include more participants in order to hold an entailment relation. The goal of the measure is to deal with synonymous entities.

Our approach for the NER similarity measure consists in the following: first, the named entities are grouped by type; then, the content of the same type of groups (e.g *Scripps Hospital* is an organization) is compared using the cosine similarity equation. However, if the surface realizations of the same named entity in T and H are different, we retrieve words that share the same context as the named entity in question; the words are retrieved from Dekang Lin's thesaurus. Therefore, the cosine similarity equation will have more information than just the named entity.

For instance, consider the T–H pair from the previous example. The entity from T: *IBM Corp.* and the entity from H: *International Business Machines* have the same tag *organization*. Our metric groups them and adds words from the similarity thesaurus resulting in the following bag-of-words (BoW) representation:

T's entities: {*IBM Corp., ..., Microsoft, Intel, Sun Microsystems, Motorola / Motorola, Hewlett-Packard / Hewlett-Packard, Novell, Apple Computer, ...*}

H's entities: {*International Business Machines, ..., Apple Computer, Yahoo, Microsoft, Alcoa, ...*}.

Finally, the metric computes the cosine similarity between these BoWs.

3.4 TINE

TINE [11] is an automatic metric based on the use of shallow semantics to align predicates and their respective arguments between a pair of sentences. The metric combines a lexical matching with a shallow semantic component to address adequacy for machine translation evaluation. The goal of this metric is to provide a flexible way of align shallow semantic representations (semantic role

labels) by using both the semantic structure of the sentence and the content of the semantic components.

A verb in the hypothesis H is aligned to a verb in the text T if they are related according to the following heuristics: (i) the two verbs share at least one class in VerbNet, or (ii) the pair of verbs holds a relation in VerbOcean.

For example, in VerbNet the verbs *spook* and *terrify* share the same class, namely, *amuse-31.1*, and in VerbOcean the verb *dress* is related to the verb *wear*.

The following example shows how the alignment of verbs and predicates is performed. Consider:

H: *The lack of snow discourages people from ordering ski stays in hotels and boarding houses.*
T: *The lack of snow is putting people off booking ski holidays in hotels and guest houses.*

Then, the algorithm proceeds with the following steps:

1. Extract verbs from H: $V_H = \{discourages, ordering\}$
2. Extract verbs from T: $V_T = \{putting, booking\}$
3. Similar verbs aligned with VerbNet (shared class get-13.5.1):
   V = $\{(v_H = order, v_T = book)\}$
4. Compare arguments of ($v_H = order$, $v_T = book$):
   A_H = {A0, A1, AM-LOC}
   A_T = {A0, A1, AM-LOC}
5. $A_H \cap A_T$ = {A0, A1, AM-LOC}
6. Exact matches:
   $H_{A0} = \{people\}$ and $T_{A0} = \{people\}$
7. Different word forms:
   expand the representation:
   $H_{A1} = \{ski, stays\}$ and $T_{A1} = \{ski, holidays\}$
   to:
   $H_{A1} = \{\{ski\}, \{stays, remain, ..., journey, ...\}\}$
   $T_{A1} = \{\{ski\}, \{holidays, vacations, trips, ..., journey, ...\}\}$
8. Similarly with H_{AM-LOC} and T_{AM-LOC}

Here, V_H is the set of verbs in the hypothesis H, V_T is the set of verbs in the text T, A_H is the set of arguments of the hypothesis H, and A_T is the set of arguments in the text T.

The metric aligns similar verbs with the ontology and similar arguments with a distributional thesaurus. Then, the metric computes a similarity score given the previous alignment points.

4 Experimental Results

We compared our method with other machine learning-based methods and with methods that use a SRL representation as one of its features.

Table 2. The 10-fold cross-validation accuracy results over the RTE development datasets

Algorithm	RTE-1	RTE-2	RTE-3
SVM	**64.90%**	**59.00%**	**66.62%**
NaïveBayes	62.25%	58.25%	64.50%
AdaBoost	**64.90%**	57.75%	62.75%
BayesNet	64.19%	**59.00%**	65.25%
LogitBoost	62.25%	52.50%	61.00%
MultiBoostAB	64.55%	**60.50%**	64.00%
RBFNetwork	61.90%	54.25%	64.80%
VotedPerceptron	63.31%	57.75%	65.80%

We used the RTE-1, RTE-2, and RTE-3 development datasets to train the classifiers. Table 2 shows the tenfold cross-validation results.

The SVM achieved the best results in the experiments during the training phase. We use this algorithm to perform the classification over the RTE test datasets. The data used for classification are the test datasets of the RTE challenge. The experimental results are summarized in Table 3.

Table 3. Comparison with previous accuracy results over the RTE test datasets

Method	RTE-1	RTE-2	RTE-3
Roth and Sammons [12]	–	–	65.56%
Burchardt and Frank [1], Burchardt et al. [2]	54.6%	59.8%	62.62%
Delmonte et al. [8], [6], [7]	59.25%	54.75%	58.75%
Our method with SVM	53.87%	55.37%	61.75%

Table 4 shows the overall accuracy results of the RTE systems on the RTE test datasets against our method. Our method is close to the average performance but below the best method.

However, the systems that showed the best results in the RTE challenge are complex and sophisticated systems. In contrast, our method relies on a small number of simple features. Our main semantic feature is focused in predicate-argument information, while other methods tackle several semantic phenomena such as negation and discourse information [12] or rely on a large number of features [2].

Table 4. Comparison with overall accuracy results over the RTE test datasets

Challenge	Our method	Average	Best
RTE-1	53.87%	55.12%	70.00%
RTE-2	55.37%	58.62%	75.38%
RTE-3	61.75%	61.14%	80.00%

Error analysis shows that the most common source of errors for our method is the TINE similarity metric. The following categories of errors made by this metric are the most common ones:

1. Lack of coverage in the ontologies, for example:

   T: *This year, women were awarded the Nobel Prize in all fields except physics.*
   H: *This year the women received the Nobel prizes in all categories less physical.*

   The lack of coverage in the VerbNet ontology prevented the detection of the similarity between *receive* and *award*.
2. Matching of unrelated verbs, for example:

   T: *If snow falls on the slopes this week, Christmas will sell out too, says Schiefert.*
   H: *If the roads remain snowfall during the week, the dates of Christmas will dry up, said Schiefert.*

   In VerbOcean, *remain* and *say* are incorrectly indicated to be related. The VerbOcean dictionary was created by a semi-automatic extraction algorithm [3] with an average accuracy of 65.5% and thus contain a considerable number of errors.
3. Incorrect tagging of the semantic roles by the semantic parser SENNA[4], for example:

   T: *Colder weather is forecast for Thursday, so if anything falls, it should be snow.*
   H: *On Thursday, must fall temperatures and, if there is rain, in the mountains should.*

   The position of the predicates affects the SRL tagging. The predicate *fall* has the roles (A1, V, and S-A1) in the reference, and the roles (AM-ADV, A0, AM-MOD, and AM-DIS) in the hypothesis H. As a consequence, the metric cannot match the fillers. Also, SRL systems do not detect phrasal verbs: e.g., the action *putting people off* is similar to *discourages* but current SRL systems do not detect this.

As we see, the quality of the semantic parser and the coverage of the ontologies are significant causes that affect the performance of our method.

In addition, on the RTE-1 test dataset with 800 T–H pairs, the coverage of the semantic metric is 491 pairs. This means that the system only predicts a certain amount of pairs. On the RTE-3 dataset, on which we obtain the best result, also has 800 T–H pairs, but the coverage on this dataset is much better: 556 pairs. Accordingly, our method has a smaller amount of errors due to a greater number of semantic-scored pairs.

[4] SENNA, http://ml.nec-labs.com/senna/

5 Conclusions and Future Work

We have presented a machine learning-based system for Recognizing Textual Entailment (RTE) task, based on new similarity metrics as well as simple string-based metrics and a shallow-semantic metric. The new similarity measures are based on: (i) chunking, (ii) named entities.

Our method has performance comparable with the average performance of methods in the RTE challenges. However, its performance is below that of the best know methods. On the other hand, our method relies on a small number of simple features, and our system only tackles one semantic phenomenon: predicate-argument information.

A preliminary error analysis shows that a main source of errors is the alignment of predicates by the TINE measure. However, if the system has more pairs tagged with predicate-argument information, then its performance improves.

In order to improve the performance of our current machine learning-based system, in our future work we will attempt to resolve the errors caused by the TINE metric based on the error analysis, or will use a different semantic approach to RTE [10].

Our semantic metric uses a distributional thesaurus to measure the similarity between arguments, so that, for example, *cat* and *dog* will be aligned because they share the same context. A possible direction to improve the semantic metric is to add hard constraints over the core arguments. These constrains can be defined as thresholds learned over the training dataset.

Acknowlegments

This work was partially supported by the Mexican National Council for Science and Technology (CONACYT), scholarship reference 309261, and SIP-IPN grant 20121823.

References

[1] Burchardt, A., Frank, A.: Approaching textual entailment with LFG and FrameNet frames. In: Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment. Venice, Italy (2006)

[2] Burchardt, A., Reiter, N., Thater, S., Frank, A.: A semantic approach to textual entailment: System evaluation and task analysis. In: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. pp. 10–15. Association for Computational Linguistics, Prague (June 2007)

[3] Chklovski, T., Pantel, P.: VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In: Lin, D., Wu, D. (eds.) Proceedings of EMNLP 2004. pp. 33–40. Barcelona, Spain (Jul 2004)

[4] Dagan, I., Dolan, B., Magnini, B., Roth, D.: Recognizing textual entailment: Rational, evaluation and approaches – erratum. Natural Language Engineering 16(1), 105 (2010)

[5] Dagan, I., Glickman, O.: The PASCAL Recognising Textual Entailment challenge. In: In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (2005)
[6] Delmonte, R., Bristot, A., Boniforti, M.A.P., Tonelli, S.: Coping with semantic uncertainty with VENSES. In: Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment. Venice, Italy (2006)
[7] Delmonte, R., Bristot, A., Piccolino Boniforti, M.A., Tonelli, S.: Entailment and anaphora resolution in RTE 3. In: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. pp. 48–53. Association for Computational Linguistics, Prague (June 2007)
[8] Delmonte, R., Tonelli, S., Piccolino Boniforti, M.A., Bristot, A., Pianta, E.: VENSES – a linguistically-based system for semantic evaluation. In: In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (2005)
[9] Mehdad, Y., Magnini, B.: A word overlap baseline for the Recognizing Textual Entailment task (2009)
[10] Pakray, P., Barman, U., Bandyopadhyay, S., Gelbukh, A.: A statistics-based semantic textual entailment system. Lecture Notes in Artificial Intelligence 7094, 267–276 (2011)
[11] Rios, M., Aziz, W., Specia, L.: TINE: A metric to assess MT adequacy. In: Proceedings of the Sixth Workshop on Statistical Machine Translation. pp. 116–122. Association for Computational Linguistics, Edinburgh, Scotland (July 2011)
[12] Roth, D., Sammons, M.: Semantic and logical inference model for textual entailment. In: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. pp. 107–112. Association for Computational Linguistics, Prague (June 2007)

Neural Networks & Unconventional Computation

Image transform based on an alpha-beta convolution model

Antonio Alarcón-Paredes, Elías Ventura-Molina, Oleksiy Pogrebnyak, and Amadeo Argüelles-Cruz

Center for Computing Research, National Polytechnic Institute. México, D.F.
aparedesb07@sagitario.cic.ipn.mx, eventura_a12@sagitario.cic.ipn.mx, olek@cic.ipn.mx, jamadeo@cic.ipn.mx

Abstract. In this paper, it is presented a novel method based on an alpha-beta convolution model, to be used at transformation stage in an image compression system. This method takes the alpha-beta's associative memories theory and is applied to a set of images in grayscale. Since these associative memories are used for data with binary inputs and outputs, it is also presented a modification to the original alpha and beta operators in order to be applied directly to the the pixel values of an image. The proposed method is applied as a traditional convolution, with the difference that instead of making sums of products, there are performed maximum or minimum operations of the alphas and betas. The Shannon entropy is used to measure the amount of bits of information contained in the images. The traditional images transform usually do not provide any kind of compression and they also use complex operations. Therefore, this new method represents an advantage by offering a lower amount of entropy in the transformed image that in the original image by making use of simple operations such as addition, subtraction, minimums, and maximums.

1 Introduction

The imminent growth in the amount of the existing information has given the guideline to think on mathematical methods that help us to represent the information in a compact manner, reducing the number of bits used for its representation. For this reason, data compression systems and, in a particular case, image compression systems have been created. There exists lossless image compression systems, whose stages are: transformation and coding; as well as image compression systems, whose stages are: transformation, quantization and coding [5], [13]. Although there are a large number of image transforms, the most common one is the DCT (Discrete Cosine Transform). The DCT was proposed by Ahmed, Nataraj and Rao in 1974 [1]. It is a transform that is applied to blocks of pixels of an image, each block is usually constituted by 8 x 8 or 16 x 16 pixels, and it consists in a bijective function that maps one to one the image values allowing DCT to be reversible, thus there is no loss of information, but it does not compress the image neither [10], [17]. From the early 80s, the CCITT (Consultative

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 351–362
Paper Recived 28-09-2012 and Acepted 07-11-2012

Committee for International Telegraphy and Telephony) and ISO (International Organization for Standardization) began to work together in order to develop an international standard for image compression, which was achieved in 1992 with the acronym JPEG (Joint Photographic Experts Group) [9], [20] that uses the DCT in its transformation stage. From this date, diverse modifications have been developed to the DCT for its fast implementation, such as Chen [4], who takes advantage of the symmetry of the cosine function to reduce the needed operations to implement the transform. Subsequently, Arai [3] develops the DCT only taking into consideration the real part of a DFT (Discrete Fourier transform) and using the FFT (Fast Fourier Transform) algorithm that was proposed by Winogard in [22]. In addition there have been new developments for the DCT which can be consulted in [12], [15], [16] and [24]. In Shannon information theory [18], the quantity of information is defined as a probabilistic process, taking the image as the information source; denoted by S with n elements $s_1, s_2, \ldots, s_n$ and $i = 1, 2, \ldots n$, its entropy is defined as:

$$H(S) = -\sum_i P(s_i) \log_2 (P(s_i)) \tag{1}$$

for this reason, it is used the Shannon entropy as a measure to know the amount of bits which represent each image in this paper.

This paper is organized as follows: in section 2 it is shown the theoretical framework for the development of the new method, section 3 describes the proposed method. The results and conclusions are shown in section 4 and section 5, respectively.

2 Theoretical Support

This paper presents a new method based on a modification of the original algorithm of the alpha-beta associative memories [23], and focuses in the image transformation stage. This section shows the theory which is the base for the proposed method.

2.1 Associative memories

The AM (Associative memories) [8] are pattern recognition's algorithms, whose purpose is to recover full patterns from input patterns that could be altered. Input patterns are represented by column-vectors $\mathbf{x}$ and output patterns by column-vector $\mathbf{y}$. For each input pattern $\mathbf{x}$, there is one and only one output pattern $\mathbf{y}$, forming an association as an ordered pair: $(\mathbf{x}, \mathbf{y})$. The set of the p pattern associations is named fundamental association set or simply fundamental set, with $\mu = 1, 2, \ldots, p$, and it is represented as:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \,|\mu = 1, 2, \ldots p\} \tag{2}$$

The operation of an AM is divided into two phases: the learning phase, where input patterns $\mathbf{x}$ are associated with their corresponding output patterns $\mathbf{y}$ to

generate the associative memory; and the recovery phase, in which we introduce a pattern $\mathbf{x}$ as the input to the memory, and as result we expect to receive a corresponding pattern $\mathbf{y}$ in the outcome. There are two types of AM which are classified according to the nature of their pattern: auto-associative memories and hetero-associative memories. The memory is auto-associative if it fulfills that $\forall \mu, \mathbf{x}^{\mu} = \mathbf{y}^{\mu}$, *i.e.* each input pattern is equal to its corresponding output pattern; on the other hand, the memory will be hetero-associative if it is true that $\exists \mu \in \{1, 2, \ldots, p\} \, | \mathbf{x}^{\mu} \neq \mathbf{y}^{\mu}$, *i.e.* if there exists at least an input pattern different to its corresponding output pattern.

Traditional models of AM, such as Lernmatrix [19], Correlograph [21], as well as Linear Associator [2], [11] operate within the theory of artificial neuron model of McCulloch and Pitts [14]. That is,its operation is based on sums of products. In addition there are the morphological AM [6] and the alpha-beta AM [23] which are the only ones that are handled outside of this theory and instead of sums of products, they use maximums (or minimums) values of the sums, in the case of the morphological; and maximum (or minimum) of alphas and betas, in the case of the alpha-beta.

2.2 Convolution

On the other hand, in the theory of image processing, convolution is widely used. The convolution of an image is an operation in the spatial domain, *i.e.*, a method which operates directly on the value of the pixels in the image. To perform a convolution is required to define a mask, also called window, whose choice of values should be carefully taken; the mask is usually $3px \times 3px$. Convolution can be expressed as follows:

$$g\left(x, y\right) = T\left[f\left(x, y\right)\right] \tag{3}$$

where $f(x, y)$ is the input image, $g(x; y)$ is the output image and T is an operator in f that is defined on specific neighbors points to $(x; y)$; we will place the mask on each of these points to make the necessary operations. These operations consist of multiplying the pixels in the neighborhood (x, y) with the coefficients of the mask, adding the results to obtain the response in the pixel (x, y) of the resulting image [7].

Note that in both the classical AM models, and the convolution of images there are used sums of products; in the case of the alpha-beta AM, to its operation the sum of products is changed to maximums (or minimums) of alpha and betas. For the development of this work, it was thought to exchange the traditional function of convolution for a new method that is implemented as a convolution based on maximum or minimum of alfas and betas.

2.3 Alpha and beta operations

The alpha-beta AM [23] uses maximums and minimums and two binary operations proposed specifically: alpha (α) and beta (β), that establish the name

of alpha-beta AM. In order to make the definition of the binary operations alpha and beta, there should be specified the set A and the set B beforehand, as following:

$$A = \{0, 1\} \text{ and } B = \{0, 1, 2\} \tag{4}$$

Binary operation $\alpha : A \times A \to B$ is defined as:

Table 1 $\alpha : A \times A \to B$

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

Binary operation $\beta : B \times A \to A$ is defined as:

Table 2 $\beta : B \times A \to A$

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

3 Proposed method

As can be shown, in *Table* 1 and *Table* 2, the alpha and beta operators could only handle binary inputs and outputs, hence it was required to extend them in order to handle real-valued numbers and thus operate directly over the pixel value of an image.

The new operation $\alpha_{\mathbb{R}} : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$ is defined as

$$\alpha_{\mathbb{R}}(x, y) = x - y + 1 \tag{5}$$

The alpha-beta AM can be max type and min type; besides, the method proposed in this paper can operate both types: max or min. Thus, the operation $\beta_{\mathbb{R}} : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$ has an operation for each type of recovery, max ($\beta^{\vee}_{\mathbb{R}}$) or min ($\beta^{\wedge}_{\mathbb{R}}$), as follows:

$$\text{If } x = y \longrightarrow \beta_{\mathbb{R}}(x, y) = 1 \tag{6}$$

$$\beta^{\vee}_{\mathbb{R}}(x, y) = y - |x| - 1 \tag{7}$$

$$\beta^{\wedge}_{\mathbb{R}}(x, y) = x - |y| - 1 \tag{8}$$

Definition 1. *Let* $\mathbf{A} = [a_{ij}]$ *be a matrix of size* $m \times n$ *representing an image, and let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix* $d \times d$ *-dimensional and represents an* ***image sub-block*** *of* $\mathbf{A}$*, such as:*

$$sb_{ij} = a_{r_i t_j} \tag{9}$$

where $i, j = 1, 2, \ldots, d$*,* $r = 1, 2, \ldots, m$*,* $t = 1, 2, \ldots, n$*, and* $sb_{ij} = a_{r_i t_j}$ *represents the pixel value given by coordinates* $(r + i, t + j)$*, where* (r, t) *and* $(r + d, t + d)$ *are the beginning and the end of the sub-block, respectively.*

Definition 2. *Let* $\mathbf{A} = [a_{ij}]$ *be a matrix of size* $m \times n$ *that represents an image. The value denoted by* ε *is a value greater than the maximum value that could assume a pixel of an image, thus:*

$$\varepsilon > \bigvee_{\forall i,j} a_{ij} \tag{10}$$

Definition 3. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix* $d \times d$ *-dimensional and represents an image sub-block, and let* $\varepsilon > \bigvee_{\forall i,j} sb_{ij}$ *be a value greater than the maximum value that could assume a pixel of the sub-block* $\mathbf{sb}$*. The* ***transformation mask*** *of* max *type, denoted as* $\mathbf{mt}^{\vee} = \left[mt_{ij}^{\vee}\right]_{h \times h}$*, is initialized to* 0*, except for its central pixel that assumes the value* ε*, when the* max *type* $(\beta_{\mathbb{R}}^{\vee})$ *is used:*

$$mt_{ij}^{\vee} = \begin{cases} \varepsilon & \textit{central pixel} \\ 0 & \textit{other case} \end{cases} \tag{11}$$

Definition 4. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix* $d \times d$ *-dimensional and represents an image sub-block, and let* $\varepsilon > \bigvee_{\forall i,j} sb_{ij}$ *be a value greater than the maximum value that could assume a pixel of the sub-block* $\mathbf{sb}$*. The* ***transformation mask*** *of* min *type, denoted as* $\mathbf{mt}^{\wedge} = \left[mt_{ij}^{\wedge}\right]_{h \times h}$*, is initialized to* 0*, except for its central pixel that assumes the value* $-\varepsilon$*, when the* min *type* $(\beta_{\mathbb{R}}^{\wedge})$ *is used:*

$$mt_{ij}^{\wedge} = \begin{cases} -\varepsilon & \textit{central pixel} \\ 0 & \textit{other case} \end{cases} \tag{12}$$

By means of simplicity, in the following definitions, the notation of the sub-block $\mathbf{sb} = [sb_{ij}]$ and the transformation mask (either if is max or min) $\mathbf{mt} = [mt_{ij}]$ as coordinates, thus, $[sb_{ij}] = \mathbf{sb}(i, j)$ y $[mt_{ij}] = \mathbf{mt}\,(i, j)$.

Definition 5. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt_{rt}^{\vee}]_{h \times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***alpha*** max ***convolution*** *operation* $\left(*\alpha_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right)\right)$*, is expressed as:*

$$\mathbf{t}\,(i, j) = *\alpha_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right) = \bigvee_{i=-a}^{a} \bigvee_{j=-b}^{b} \alpha_{\mathbb{R}}\left(\mathbf{sb}\,(i + r, j + t)\,, \mathbf{mt}^{\vee}\,(r, t)\right) \tag{13}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Definition 6. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt^{\vee}_{rt}]_{h\times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***beta*** min ***convolution*** *operation* $\left(*\beta_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right)\right)$*, is expressed as:*

$$\mathbf{t}\left(i,j\right) = *\beta_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\vee}\right) = \bigwedge_{i=-a}^{a}\bigwedge_{j=-b}^{b} \beta^{\wedge}_{\mathbb{R}}\left(\mathbf{sb}\left(i+r, j+t\right), \mathbf{mt}^{\vee}\left(r,t\right)\right) \tag{14}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Definition 7. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt^{\vee}_{rt}]_{h\times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***alpha*** min ***convolution*** *operation* $\left(*\alpha_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right)\right)$*, is expressed as:*

$$\mathbf{t}\left(i,j\right) = *\alpha_{\min}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right) = \bigwedge_{i=-a}^{a}\bigwedge_{j=-b}^{b} \alpha_{\mathbb{R}}\left(\mathbf{sb}\left(i+r, j+t\right), \mathbf{mt}^{\wedge}\left(r,t\right)\right) \tag{15}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Definition 8. *Let* $\mathbf{sb} = [sb_{ij}]$ *be a matrix of size* $d \times d$ *representing an image sub-block, let* $\mathbf{mt}^{\vee} = [mt^{\vee}_{rt}]_{h\times h}$ *be a transformation mask* max *type, and let* $\mathbf{t} = [t_{ij}]$ *be the transformed sub-block; the* ***beta*** max ***convolution*** *operation* $\left(*\beta_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right)\right)$*, is expressed as:*

$$\mathbf{t}\left(i,j\right) = *\beta_{\max}\left(\mathbf{sb}, \mathbf{mt}^{\wedge}\right) = \bigvee_{i=-a}^{a}\bigvee_{j=-b}^{b} \beta^{\vee}_{\mathbb{R}}\left(\mathbf{sb}\left(i+r, j+t\right), \mathbf{mt}^{\wedge}\left(r,t\right)\right) \tag{16}$$

where $a = \frac{h-1}{2}$ *and* $b = \frac{h-1}{2}$*.*

Alpha-beta convolution transform algorithm

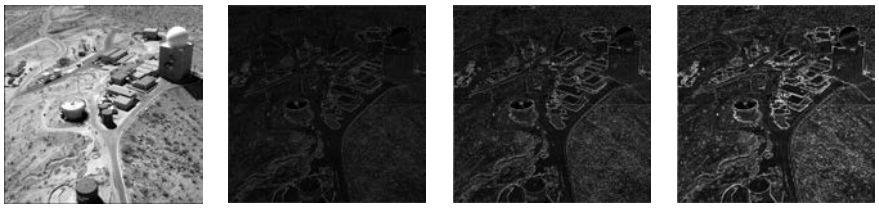
1. As usual on traditional image transforming methods, the alpha-beta convolution method proposed in here, is applied individually to an image sub-blocks of size $d \times d$. The image denoted as $\mathbf{A} = [a_{ij}]_{m\times n}$ is divided into $\kappa = (m/d) \cdot (n/d)$ sub-blocks $\mathbf{sb}^{\omega} | \omega = 1, 2, \ldots, \kappa$, where m and n are the image height and width respectively, a_{ij} is the $ij-th$ pixel of image with $a \in \{0, 1, 2, \ldots, L-1\}$ being L the number of bits that represents the value of a pixel.
2. Initialize to 0 all values in the resulting image $\mathbf{T} = [t_{ij}]_{m\times n}$.
3. Create the transforming mask **mt** depending on the desired usage: max or min, according to the *Definition 2*, *Definition 3* and *Definition 4*.
4. Apply the *alpha* max (or min) *convolution* to each sub-block according to the *Definition 5* and *Definition 7* and place the outcome on resulting image **T**. □

Inverse alpha-beta convolution transform algorithm

1. The inverse alpha-beta convolution method, is applied individually to the sub-blocks of transformed image **T**.
2. Initialize to 0 all values in the resulting recovered image $\mathbf{A}' = \left[a'_{ij}\right]_{m\times n}$.
3. Locate each sub-block of the transformed image **T** and apply the *beta* min (or max) *convolution* according to the *Definition 8* and *Definition 6*, using **the same transforming mask** used in the alpha-beta convolution transform algorithm and place the outcome on recovered image $\mathbf{A}'$. □

Since each time the **mt** is created, the maximum value of the current sub-block is taken, it is required to store every maximum in a vector.

4 Results



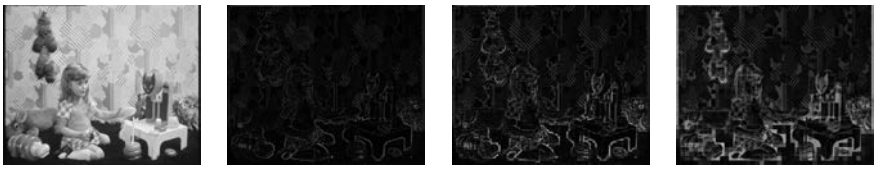
a) b) c) d)

Fig. 1 *Aerial* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

In order to measure the proposed transform, a comparison between the original and the transformed image was performed using the Shannon entropy [18]. The method presented in this paper was applied to a set of 20 images widely used in many other papers, and was applied varying the size of **sb** (4×4, 8×8, 16×16) for each image, obtaining 60 transformed images. The set of testing images are grayscale, and 8 bits/pixel, and have different sizes.

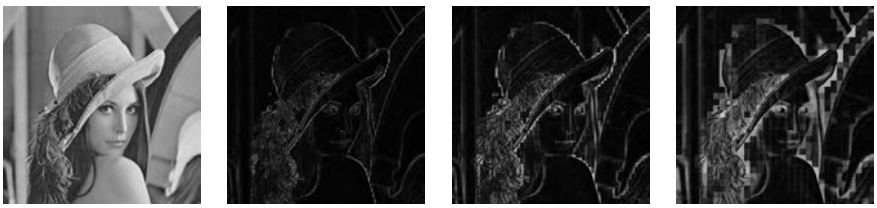
a) b) c) d)

Fig. 2 *Baloon* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).



a) b) c) d)

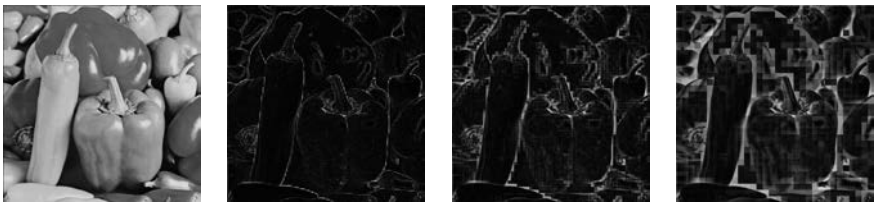
Fig. 3 *Girl* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).



a) b) c) d)

Fig. 4 *Lenna* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

Figures 1 to 6 show the original image (a) and some transformed images using 4×4 sub-block size at (b), 8×8 sub-block size at (c), and 16×16 sub-block size at (d). The results that compare the entropy between the original image versus the transformed images are shown at *Table* 3.



a) b) c) d)

Fig. 5 *Peppers* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

a) b) c) d)

Fig. 6 *Zelda* image (a), with sub-block different sizes: 4×4 (b), 8×8 (c), 16×16 (d).

Table 3. Entropy results with different **sb** sizes.

Image	Size	Entropy	Entropy 4×4	Entropy 8×8	Entropy 16×16
Aerial	2048×2048	7.1947	5.7846	6.4473	6.8813
Baboon	512×512	7.3577	6.4610	6.9275	7.1501
Baloon	720×576	7.3459	3.8442	4.7928	5.6843
Barbara	720×576	7.4838	5.7309	6.3724	6.8072
Bike	2048×2560	7.0219	5.3234	6.0152	6.5303
Board	720×576	6.8280	4.5892	5.3763	6.2891
Boats	512×512	7.1914	5.5520	6.2688	6.8104
Couple	512×512	7.2010	5.3092	6.0276	6.5776
Elaine	512×512	7.5060	5.1970	5.8841	6.5101
F-16	512×512	6.7043	4.8730	5.5588	6.0325
Girl	720×576	7.2878	5.0300	5.8262	6.4912
Goldhill	720×576	7.5300	5.3532	6.1223	6.6593
Hotel	720×576	7.5461	5.4036	6.2258	6.9116
Lenna	512×512	7.4474	5.1027	5.8879	6.5444
Man	512×512	7.1926	5.5016	6.3273	6.9066
Peppers	512×512	7.5943	5.1029	5.9029	6.662
Sailboat	512×512	7.4847	5.7920	6.5942	7.1399
Tiffany	512×512	6.6002	4.7339	5.3748	5.8841
Woman	2048×2560	7.2515	5.4033	6.0797	6.5413
Zelda	720×576	7.3335	4.5692	5.3577	6.1714

The data contained in the table refers to the image dimensions in pixels, entropy of the original image (Entropy), the entropy of the transformed image using a **mt** of 4×4 pixels (Entropy 4×4), the entropy of the transformed image using a **mt** of 8×8 pixels (Entropy 8×8), and the entropy of the transformed image using a **mt** of 16×16 pixels (Entropy 16×16). The transformation stage of an image compressor does not provide any information reduction, its main purpose is to make easier the compression at following steps (quantization and coding). Since the alpha-beta convolution transform provides information reduction, it is clear that represents an advantage over the traditional transforming methods, besides the method proposed in this paper uses very simple operations such as addition, substraction, minimums and maximums (comparisons).

5 Conclusions

The modified alpha and beta operators are capable to handle real valued inputs and may be used as well in image processing, and it is clear the next step could be the use of modified alpha-beta associative memories to perform the image transform.

By replacing the sums of products in traditional convolution with the maximums or minimums of alpha and beta, was possible to create an alternative for image transform in an image compression system, offering low computational cost by means of using simple operations.

The experimental results show that the alpha-beta convolution transform is reversible, that is, this new method has no information loss. Although 8 definitions were presented, it is needed the proposition and demonstration of some lemmas or theorems is required in order to formally prove this method is reversible.

As mentioned, the transformation stage in an image compression system does not provide any compression to images, so since the method proposed in this paper offers a smaller entropy on the transformed images, it also represents an advantage for the quantization and coding stages in an image compression system.

References

1. Ahmed, N., Natarajan, T., Rao, K. R.: Discrete cosine transform. IEEE Transactions on computers. **23** (1974) 90-93
2. Anderson, J. A.: A simple neural network generating an interactive memory, Mathematical Biosciences. **14** (1972) 197-220
3. Arai, Y., Agui, T., Nakajima, M.: A fast DCT-SQ scheme for image. Transactions of the IEICE. **71** (1988) 1095-1097
4. Chen, W., H., Smith, C., H., Fralick, S. C.: A fast computational algorithm for the discrete cosine transform. IEEE Transactions on Communications. **25** (1977) 1004-1009
5. Cosman, P., Gray, R., Olshen, R.: Handbook of medical imaging processing and analysis. Academic Press. USA. (2000) ISBN 0-12-077790-8
6. Díaz de León Santiago, J.L. & Yáñez Márquez, C.: Memorias Morfológicas Heteroasociativas. IT-57, Serie Verde, ISBN 970-18-6697-5, CIC-IPN, México (2001)
7. González, R. C., Woods, R. E. & Eddins, S. L.: Digital Image Processing using MATLAB. Prentice Hall. (2003). ISBN 0130085197
8. Hassoun, M. H.: Associative Neural Memories. Oxford University Press, New York. (1993)
9. ISO/IEC IS 10918-1 | CCITT T.81: Digital Compression and Coding of Continuous-Tone Still Image. (1992) ISO/IEC
10. Jain, A. K.: A sinusoidal family of unitary transforms. IEEE Transaction on Pattern Analysis and Machine Intelligence. **4** (1979) 356-365
11. Kohonen, T.: Correlation matrix memories, IEEE Transactions on Computers, C-21, **4** (1972) 353-359

12. Kok, C. W.: Fast algorithm for computing discrete cosine transform. IEEE Transactions on Signal Processing. **45** (1997) 757-760
13. Lakac, R., Plataniotis, K. N.: Color image processing methods and applications. CRC Press, Taylor & Francis. USA. (2007) ISBN 978-0-8493-9774-5
14. McCulloch, W. & Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics. **5** (1943) 115-133
15. Pan, W.: A fast 2-D DCT algorithm via distributed arithmetic optimization. IEEE Proceeding Image Processing. **3** (2000) 114-117
16. Ramírez, J., García, A., Frenadse, P. G., Parrilla, L., Lloris, A.: A new architecture to compute the discrete cosine transform using the quadratic residue number system. IEEE Int. Symposium on Circuits and Systems. (2000)
17. Rao, K. R., Yip, P., C.: The transform and data compression handbook. The Electrical Engineering and Signal Processing Series. (2001) ISBN 0-8493-3692-9
18. Shannon, C. E.: A mathematical theory of communication. Bell system technical journal. **27** (1948) 379-423 y 623-656
19. Steinbuch, K. V.: Die Lernmatrix. Kybernetik, **1**, **1** (1961) 36-45
20. Wallace, G. K.: The JPEG still picture compression standard. Proceedings of Communications of the ACM. **34** (1991) 30-44
21. Willshaw, D., Buneman, O. & Longuet-Higgins, H.: Non-holographic associative memory, Nature. **222** (1969) 960-962
22. Winograd, S.: On computing the discrete Fourier transform. Proceedings of the National Academy of Sciences of the United States of America. **73** (1976) 1005-1006
23. Yáñez, C.: Associative memories based on order relations and binary operators (in Spanish). Doctoral Theses. Center for Computing Research. (2002)
24. Yu, S., Swartzlander, E. E.: DCT implementation with distributed arithmetic. IEEE Transactions on Computers. **50** (2001) 985-991

Security Token for Web Bank Applications Using a Linear and Congruential Random Number Generator

Luis Orantes[1], Marco Ramírez[2], Pablo Manrique[2], Victor Ponce[2], Aniceto Orantes[3], Victor Salazar[3], Antonio Montes[3], Carlos Hernández[4], Eric Gómez[5]

[1]Center for Research in Computing, Av. Juan de Dios Bátiz, Mexico City 07738, Mexico
lorantesg1101@alumno.ipn.mx
[2]Center for Research in Computing, Av. Juan de Dios Bátiz, Mexico City 07738, Mexico
{mars, pmanriq, vponce}@cic.ipn.mx
[3]HighBits, Av. Central Poniente #847 Int. 3, Tuxtla Gutiérrez, Chiapas 29000, Mexico
{aorantes, vsalazar, antonio}@highbits.com
[4]Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas, Av. Instituto Politécnico Nacional 2580, Mexico City 07340, Mexico
carlos@highsecret.com
[5]Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Zacatenco, Av. Instituto Politécnico Nacional S./N. Unidad Profesional Adolfo López Mateos. Mexico City 07738, Mexico
ergomez@ipn.mx

Abstract. This paper presents a new algorithm for using with an one-time password security token; the objective is to provide security for the authentication of customers using bank websites even in the cases when the user has been the victim of a phishing or spyware attack and their bank account secret password has been stolen. For the token's performance, the algorithm make use of a Linear and Congruential Random Number Generator (LCG) (for a better understanding of the presented algorithm a short introduction to this arena is given), and an exhaustive algorithm for the validation of the one-time password keys is presented. This paper shows that the present algorithm is easy to implement and safer than a competing algorithm widely used in today's security tokens.

Keywords: Security token, cryptography, random number generators.

1 Introduction

Bank institutions have modernized their operations by allowing their customers to perform almost any account transaction using the Internet. One of these operations has been to transfer funds electronically to other bank accounts, bringing with this a profound danger. The user may be a victim of phishing or may have, without knowing it, installed spyware on their computer. Thieves use the user's stolen passwords to empty the user's bank account and transferring the funds to a ghost bank account for later withdrawal.

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 363–372
Paper Recived 20-08-2012 and Acepted 07-11-2012

One solution that has been given for this problem is the use of one-time password generators (security tokens), which create a password that is valid for bank access just once. With this technique, if the password is stolen it is rendered useless for accessing the bank website. In addition to the conventional password the bank website will request the one-time password which if stolen it's useless because it will be already expired just right after the legitimate user introduced it to the bank website. In other words, the token system proves that the user is who claims to be and acts as an electronic key to access the bank website services, offering security to the user even in cases where their password has been stolen.

A security token is a hardware device usually with a LCD screen (this kind of security token doesn't need to be connected to the computer when used) or provided with a USB plug (this kind of security token needs to be connected to the computer when used). Regarding the kind of approach they are used, security tokens can be classified into several categories with some of the most common approaches: 1) one-time passwords, 2) time-synchronized passwords and 3) challenge/response passwords. In this paper, the algorithm presented belongs to the one-time password category.

It has been done plenty of research about token security but in this paper just one reference will be done, just to the most common security token used nowadays. This security token is the secureID token developed by RSA Security which uses a 64 bit secret key for a hash function called Alleged SecureID Hash Function (ASHF). In [14] it has been shown that the core of this security token can be broken in a few milliseconds and they conclude that it doesn't provide the security demanded by institutions nowadays including banks. In contrast, the algorithm presented in this paper offer much higher security just as it is (with a 64 bits secret key) but it may be virtually unbreakable resizing the presented algorithm to a larger key size as explained later.

The algorithm presented in this paper uses a Pseudo Random Number Generator (RNG) as the encryption mechanism which is necessary for the generation of the one-time passwords. The following section has a brief explanation of what a RNG is because randomness is the core of the presented algorithm and a basic background on this topic is required for a better understanding of it.

2 Random Number Generators (RNG)

Computers can't be random. What computers can do is to simulate a random process by using a RNG. A RNG is an equation, which can generate a sequence of pseudorandom numbers. This sequence of pseudorandom numbers is finite and, after a certain quantity of pseudorandom numbers created, the sequence is repeated again in a cyclic way. The length of the cycle is called the RNG period and this is given, in the best cases, by the number of bits of the RNG and it's limited by the bit number of

the mathematical operations that can be computed. In ordinary PCs it is only 32 bits, nevertheless it is possible to simulate 64 bits in C++ simply by defining a long long variable type or even larger number of bits using a long int library.

There are several random number generators; the generator is selected depending on the type of a specific. There are several kinds of applications for a RNG; the most typical applications are: simulations (e.g., of physical systems to be simulated with the Monte Carlo method), cryptography and procedural generation. For the selection of the appropriate random number generator it should be taken on account not just the quality of the random numbers the generator creates but also the complexity of the generator.

For instance, for the particular application presented, it should be taken on account that this algorithm may be implemented in hardware devices (e.g. a microcontroller). These kind of devices may have limited resources such as computing capabilities and battery life. Another factor to take on account in the generator selection is energy consumption because we want the token to last as much as possible (at least a couple of years). If the generator is too complex (i.e. computationally speaking too expensive) the token life would last just a few months. A token with a short life span would be too impractical for being used in real life applications. Also it is necessary the token to generate the one-time passwords instantaneously and for achieving this goal a random number generator quick to compute is mandatory.

On one hand there are very quick generators such as xorshift [1] [2] [3] which in some cases may generate a full period but they generate very low quality random sequences. The Linear feedback shift register RNG (LSFR) [4] is a popular generator which in the past has been implemented in hardware [5] but it doesn't generate a full period. This is a very popular generator which has been implemented in several applications; important LFSR-based stream ciphers include A5/1 and A5/2, used in GSM cell phones, E0, used in Bluetooth, and the shrinking generator. Nevertheless its drawbacks were reveled when the A5/2 cipher has been broken and both A5/1 and E0 have serious weaknesses [6] [7].

On the other hand there are very high quality random number generators such as Blum Blum Shub [8] [9], Yarrow algorithm [10] (incorporated in Mac OS X and FreeBSD), Fortuna [11] [12] and CryptGenRandom [13] (incorporated in Windows) that have the inconvenient of being computationally speaking too expensive for the purpose of this algorithm.

2.1 The selected random number generator

The selected random number generator for being incorporated in the algorithm presented in this paper is the Linear and Congruential Generator". This RNG was selected by its satisfactory quality; also it has the advantage that it computes the random

sequences very quickly. Because of its low complexity this RNG is suitable for being implemented in hardware applications with limited resources (e.g. a microcontroller).

A sequence of random numbers is obtained by evaluating the following equation:

$$Z_i = (aZ_{i-} + \cdot) \bmod m \quad (1)$$

This RNG requires a seed Z_0 which is the initial state of the RNG; this can be seen as the index or initial point of the random table and it will be the first value of the random table. The next value of the random table is calculated by replacing Z_{i-1} with the new obtained value and this process is repeated again. In other words, Z_{i-1} is the value of the previously computed random number.

a, *c* and *m* are constants, nevertheless these values of these constants need to be chosen carefully because the quality of the random table depends on them. There are values for these constants that generate a poor quality random table and others that don't generate any random table at all. The value of m gives the cycle size of the random table, that is to say, from which number the random table would repeat again. It is desirable to have as many random numbers as possible; therefore the value of m usually is as big as the maximum value it can be computed.

Multiple researches have been conducted to find out what the values of these constants are the best. There are several approaches for determining the quality of the pseudorandom number sequence generated by a given constant values, for instance in [15] this quality is examined by scatter plots and spectral test but there are many other techniques for determining the quality of a pseudorandom sequence and if this quality is acceptable enough. The generator, "Linear and Conguential Generator" is especially very sensitive to the choice of these constants values and in the past have been poor choices for the values of these constant with not good results [16].

These constants need to meet some conditions; for instance to guarantee a generation of a full period for any seed values when having a non-zero value for *c*; they need to meet the following conditions [15]:

1. *c* and m must be relative primes,
2. *a-1* must be divisible by all prime factors of m,
3. *a-1* must be a multiple of 4 if m is a multiple of 4.

It is recommended for a 64-bit variable such as an ordinary PC can simulate by defining a long long variable type in C++ to be the next:

a = 6364136223846793005
c = 1442695040888963407
$m = 2^{64}$

3 Algorithm for the generation of one-time passwords on the token side

Equation 1 is capable of generating a pseudo-random table; nevertheless it is necessary to have a unique random table for every token that is constructed. It is desirable to have a different sequence of numbers for every token. It is possible to achieve this by encrypting equation 1 by performing a XOR operation of the generated random number with a secret key *K*. After this, it is necessary to resize the encrypted sequence to values that can fit within the eight digits of a LCD display (i.e. values that range from 0..99,999,999 = 9^8) This is done by performing a modulus operation of the calculated Z_i value with 9^8+1. With this the one-time passwords will range from 0 to 9^8. The equation then becomes:

$$Z_i = [aZ_{i-1} + c] \bmod m \otimes \quad (2)$$

$$OneTimePassword = Zi \bmod (9^8 + 1) \quad (3)$$

The security token needs to be initialized, it is necessary to have a seed value for this and the same value of the secret key K as the seed for the token which will be the Z_i value. This value will be stored in the token's memory and it will be used for computing the next one-time password by the security token and also at the server side to validate a one-time password. The first 1,000 one-time passwords are generated at the security token side for being wasted (this is done for allowing detection of 1,000 expired one-time password as explained later).

On the token side the value of *K* is obtained and is used as seed Z_0 for evaluating Equation 2 and 3 and generating a one-time password. This equation is evaluated again to obtain another one-time password. In this way, every time the "generate a one-time password" button is pushed a pseudorandom encrypted sequence is obtained. See Table 1 for an example of this.

Table 1. Example of a sequence of numbers generated by the token

Counter	One-time password
1	82475249
2	82040631
3	72383201
4	68714439
5	32340945
6	88383319
7	94725313
8	10436071

4 Algorithm for the one-time password validation at the server side

In order to validate a one-time password on the server side, it is necessary to decrypt the one-time password introduced by the user. However, a modulus operation can't be reversed (the resulting double modulus operation used in Equation 3 is even more irreversible). Therefore it is not possible to decrypt the one-time password introduced by the user as usual (i.e. performing the inverse operations); for this reason it is decrypted using an exhaustive algorithm.

At the server side, the one-time password is validated by computing N_{exp} (it's computed for 1,000) expired one-time passwords generated from the actual value of Z_i that is stored in the server using the algorithm previously explained (refer to section 3). These are previous valid one-time passwords that were introduced by the user but they already expired; this is done to give feedback to the user and the user may distinguish between an invalid or an expired one-time password (valid but expired; it was already introduced by the user to the bank website before). If this is found within the N_{exp} expired it comes to inform the user that introduced one-time password was valid in the past but it already expired (i.e. it won't be accepted as valid anymore).

After this, N_{val} (10,000) valid one-time passwords are generated using the same previous approach. These are the possible valid keys for the actual state of the token. If the one-time password introduced by the user is within this range the key is accepted as valid. The value of Z_i for the key that matched minus N_{exp} expired is stored in the server. This value will be necessary for computing the N_{exp} and N_{val} one-time passwords for a future validation. If no match is found it comes to reject the introduced one-time password by the user and the Z_i value stored at the server side remains unchanged.

One advantage of the algorithm presented in this paper is that a range of only N_{val} valid keys are accepted. One-time passwords that are not valid in one moment become valid in another. This is according to the actual value of Z_i stored at the present moment at the server side. This allows us to have a very low probability that a one-time password is accepted by the server as valid, which redounds to a highly secure system.

It is taken into account that the user may waste one-time passwords by generating and not using them (i.e. the user didn't introduced to the bank website a one-time password generated by the token); for this reason there is a range N valid. This range gives us a tolerance in the number of acceptable one-time passwords; it is necessary this value to be as small as possible because this will give us a lower probability that an attacker, in a random way, may guess a one-time password. This further contri-

butes to a higher level of security. Given that these one-time passwords are of eight numeric digits and have a range N_{val} (of 10,000 valid one-time passwords). This results in a probability of almost one in 100 million that a given one-time password will be accepted as valid. On the other hand, if the user wastes more than 10,000 keys by generating one-time passwords and not introducing it at the server, a desynchronization occurs and the token becomes useless; therefore the value for the range of N_{val} should be carefully considered.

Noteworthy this maximum value for wasted one-time passwords is reset between validations, which means that if the user has a considerable amount of one-time passwords wasted the range N_{val} is reset to 10,000 at the server side once a one-time password is accepted, giving the user the maximum number of one-time passwords that can be wasted again.

One advantage of this algorithm is that it isn't necessary to store the generated one-time passwords in the server. only the value of Z_i is stored in the server. For this reason, this algorithm doesn't waste space in the server for storing expired one-time passwords nor resources to determine if the one-time password introduced by the user is part of the expired ones. Perhaps this is not significant with one user but it's taken into account that a bank institution may have millions of clients then the saved space and resources becomes significant. The algorithm also has the advantage that once a one-time password has been accepted and validated, all the previous ones are automatically expired, even in cases where they haven't been introduced to the server.

5 Number of maximum generated one-time passwords

In spite of the fact, that this algorithm is able to compute eight digits range, it is not recommended to use the full range. It is possible that an attacker may be storing the one-time passwords that have been generated by the token as they are introduced to the website by the user. This leads to the hypothetical possibility that the probability of guessing a one-time password in an arbitrary way increases. This is because the attacker knows which one-time passwords already were used in the past and he wouldn't try them again. The recommendation is to cancel the token after a million one-time passwords are remaining within it. Thereby, the probability of guessing a one-time password goes from approximately 1-in-100 million to 1-in-one million. This calculation was done by assuming that there was an attacker storing the entire history of the generated one-time passwords by the token for years. This scenario is very impractical. This is a very theoretical scenario but this is done as an extra security measure.

6 Attacking the algorithm

The way to attack the presented algorithm is to have an attacker storing the entire one-time password introduced by the user in the bank web site. Let´s say the attacker has two one-time passwords collected; to attack the algorithm a full search through the 64 bits generated random sequence is performed. This is done for every possible key *K* to find which key produce a sequence that matches those the attacker has. It may occur that there is more than one single *K* than match the sequence; if this is the case the attacker needs to wait for another one-time password. This will help to reduce the number of candidate secret keys *K* which match the stored sequence and could be the secret key. This process needs to be repeated again and again until it's found that only one key *K* match the sequence of one-time passwords the attacker has collected. If this is the case the secret key *K* has been for the token under attack. The following one-time passwords can be predicted for sure, therefore breaking the token security.

Performing this attack as the algorithm was presented in this paper (i.e. having a 64 bits Linear and Congruential Random Number Generator) is already computationally speaking a hard code to break with a PC because the attacker needs to try with 2^{64} one-time passwords times the 2^{64} possible keys resulting in 2^{128} one-time passwords in total. Nevertheless it is possible to strengthen the algorithm and protect it from this kind of attack by using a larger number of bits for the RNG; (let's say 1024 bits for instance). The algorithm presented in this paper was of 64 bits; this is because 64 bits operations can be computed with ordinary PC instructions. As explained previously it is possible to simulate operations of larger number of bits using a long integer library (this library is commonly used for cryptographic applications). With this it is possible to assure that the algorithm is totally unbreakable.

7 Open Research Issues

Equation 2 is capable to generate multiple derived random sequences for a given values for the constants *a*, *c* and *m*. A research needs to be conducted for determining if the derived random sequences are as random as the original one, or at least random enough. Another issue that is left for future research is related with the constants of the RNG. In this paper the algorithm was presented assuming a 64 bits RNG and it was also proposed to enlarge the bit number of the RNG to make the algorithm stronger. Nevertheless, a research needs to be conducted to find out the best constants values for *a* and *c* that generate the best random sequence for a 1024 bit *m* size.

To keep in touch with the research advances of this project please visit http://www.highsecret.com where related information will be posted continuously.

Acknowledgments. The first author acknowledges support from the Mexican Council of Science and Technology (CONACYT) to pursue MSc studies at CIC-IPN. The second author acknowledges National Polytechnic Institute of Mexico.

Bibliography

1. Marsaglia, George (July 2003). "Xorshift RNGs". *Journal of Statistical Software*
2. Brent, Richard P. (August 2004). "Note on Marsaglia's Xorshift Random Number Generators". *Journal of Statistical Software*
3. Panneton, François (October 2005). "On the xorshift random number generators". *ACM Transactions on Modeling and Computer Simulation (TOMACS)*
4. M. Goresky and A. Klapper, Algebraic Shift Register Sequences, Cambridge University Press, 2012
5. Linear Feedback Shift Registers in Virtex Devices, Maria George and Peter Alfke, *Xilinx press*
6. Barkam, Elad; Biham, Eli; Keller, Nathan (2008), *Journal of Cryptology*
7. Lu, Yi; Willi Meier; Serge Vaudenay (2005). "The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption"
8. Lenore Blum, Manuel Blum, and Michael Shub. "A Simple Unpredictable Pseudo-Random Number Generator", *SIAM Journal on Computing*
9. Lenore Blum, Manuel Blum, and Michael Shub. "Comparison of two pseudo-random number generators", *Advances in Cryptology: Proceedings of Crypto*
10. Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator, J. Kelsey, B. Schneier, and N. Ferguson, *Sixth Annual Workshop on Selected Areas in Cryptography, Springer Verlag*
11. Niels Ferguson and Bruce Schneier, *Practical Cryptography*, published by Wiley
12. John Viega, "Practical Random Number Generation in Software," acsac, pp. 129, 19th Annual Computer Security Applications Conference
13. Dorrendorf, Leo; Zvi Gutterman, Benny Pinkas. "Cryptanalysis of the Random Number Generator of the Windows Operating System"
14. Cryptanalysis of the Alleged SecurID Hash Function, Biryukov Alex
15. A Collection of Selected Pseudorandom Number Generators, Kart Entacher
16. Press, William H., et al. (1992). Numerical Recipes *in FORTRAN 77: The Art of Scientific Computing* (2nd ed.).

Simulation & Modeling

Classical Realization of Grover's Quantum Search Algorithm using Toffoli gates

Manuel-Iván Casillas-del-Llano[1] and Álvaro-Lorenzo Salas-Brito[2]

[1]Universidad Autónoma Metropolitana. Unidad Azcapotzalco. D.F., México
al210180113@alumnos.azc.uam.mx
[2]Universidad Autónoma Metropolitana. Unidad Azcapotzalco. D.F., México
asb@correo.azc.uam.mx

Abstract. Grover's algorithm is used to search for quantum data. However, this algorithm procedure is described by means of concepts and operators from quantum theory; concepts hardly known by computer scientists. In this work we propose an alternative classical computing model of Grover's algorithm, using Toffoli gates connected with elementary gates. Our model has been programmed on a high-level programming language and tested using arbitrary elements on a data set. Our results are concordant with those presented on the reference section.

Keywords: Grover algorithm, Toffoli gates, Quantum computing.

1 Introduction

A computer is a physical device that aids us to process information while running some algorithms. An algorithm is well defined procedure, with finite description, that executes some information processing task. A task of this kind can be done by means of physical processes.

At the design level of complex algorithms, it is useful and essential to work with some idealized computational model. However, while analyzing the true limitations of a computer device, especially for practical reasons, it is important not to forget the link between computing and physics Idealized models can not fully represent all the details of these computational devices.

Classical computing has several limitations. There are problems that cannot be deal with actual computing, such as the impossibility to run on polynomial time the travelling agent problem algorithm or integer factorization.

However, it has been shown that those kinds of problems can be handled by **quantum computing**. Quantum computing uses the phenomena described by quantum theory in order to process information and execute tasks faster than classical computing. Devices that process quantum information are named **quantum computers**.

Advances in Computer Science and Engineering
Research in Computing Science 58, 2012 pp. 375–390
Paper Recived 30-09-2012 and Acepted 07-11-2012

2 Grover's Algorithm

Suppose there is a non sorted data base of size N consisting (without loss of generality) of numbers from 0 to N – 1. Using traditional algorithms, we must look up for every element on the data base in order to find the desired item. The average number of steps needed is N/2, and N on the worst case scenario; therefore, searching for an element has order of $O(N)$ time complexity. However, using quantum mechanics procedures, Grover's algorithm only requires $O\left(\sqrt{N}\right)$ steps. [1]

Initially, an *n* qubit system, with $N = 2^n$ elements, is set in an equal superposition of all basis states, expressed as

$$\left|\Psi_0\right\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left|i\right\rangle \tag{1}$$

It is posible to search for a specific element inside this system. This particular element is defined as the **marked state**, while the remaining elements of the set are defined as the **collective state** [1]. In order to perform the searching of the marked state, two special operators C and D are used, defined as inversion and diffusion operator, respectively. Operator C has the effect to invert or change the sign of the amplitude in the marked state, and ignores the rest of the elements belonging to the collective state. When operator D is applied to the superposition of states, it increases the amplitude of the marked state, decresing the amplitude of the collective state. If we define the compound operator as $U \equiv DC$, then each operation of *U* is called an iteration. It has been shown that after *U* is repeated $O\left(\sqrt{N}\right)$ times, the probability of getting the marked state when a mesaurement is made approeaches 1 [1].

2.1 Representation of inversion and diffusion operators.

Before reading the following sections, we encourage you to read [4], where the most important operations on quantum computing are explained in great detail. Also, for a wide explanation of Grover's Algorithm insights, we recommend reading [1].

Given the superposition of states $\left|\Psi_0\right\rangle = \left(1/\sqrt{N}\right)\sum_{i=0}^{N-1}\left|i\right\rangle$, we will denote the marked state as $\left|M\right\rangle$.

Inversion operator C is defined as $C \equiv I - 2|M\rangle\langle M|$, where *I* is the identity matrix[5]. Similary, diffusion operator D is defined as $D \equiv 2|\Psi\rangle\langle\Psi| - I$.

To be able to represent these operators by means of Toffoli gates and elementary operations, it is necessary to know the effect produced by them on the superposition of states. Let's split $|\Psi_0\rangle$ into two parts: the marked state and the collective state, that is, a linear combination of $|\Psi_0\rangle$ defined as:

$$|\Psi_0\rangle = \alpha|\Psi\rangle + \beta|M\rangle \tag{2}$$

where α and β are their respective amplitudes of the collective state and the marked state.

Next, we will apply the inversion operator to this linear combination of states:

$$C(\alpha|\Psi\rangle + \beta|M\rangle) = (I - 2|M\rangle\langle M|)(\alpha|\Psi\rangle + \beta|M\rangle)$$

$$C(\alpha|\Psi\rangle + \beta|M\rangle) = \alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle \tag{3}$$

At this moment, C operator has changed the sign of the marked state. Next step is to apply the diffusion operator on equation (2). That is:

$$D\left[\alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle\right] = (2|\Psi\rangle\langle\Psi| - I)\left[\alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle\right]$$

$$D\left[\alpha|\Psi\rangle - \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle\right] = \left(\alpha - \frac{4\alpha}{N} - \frac{2\beta}{\sqrt{N}}\right)|\Psi\rangle + \left(\frac{2\alpha}{\sqrt{N}} + \beta\right)|M\rangle \tag{4}$$

We have applied once the compound operator $U \equiv DC$, and so, an iteration of Grover's algorithm have been made. According to [7], the number of iterations needed to approach the amplitude of the marked state to 1 is $\lfloor \pi\sqrt{N}/4 \rfloor$.

2.2 Alternative representation of Grover's algorithm using different approaches.

Grover algorithm can be also represented by circuits using interconnected quantum gates and the Toffoli gate [8]. A circuit for 2 qubit system is shown in Figure 1.

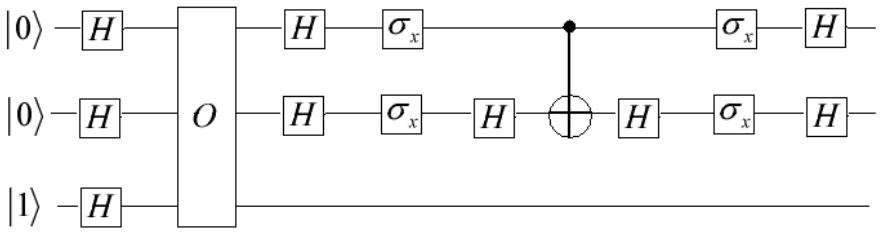


Fig. 1. A quantum circuit that implements Grover's algorithm for N = 4 elements. Pauli matrix σ_x behaves as a NOT gate. Block with letter O denotes a query to the oracle.

Another representation of Grover's algorithm can be implemented using optical approaches, as described in [16] for a system of 2 qubit elements. An implementation using two trapped atomic ion qubits for a system of 2 qubit elements is also proposed [17]. Other representations such as nuclear magnetic resonance are described in [5].

As it will be shown on the next sections, we will build a model that will also represent Grover's algorithm, but using only classical (non-quantum) elementary gates (with its limitations, see Conclusions section).

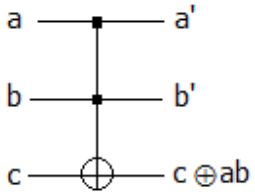
3 Toffoli gates

Toffoli gates were invented by Tommaso Toffoli [15]. Its main characteristic is that it is a universal reversible logic gate. It is a universal gate because any logic gate can be constructed by means of several Toffoli gates interconnected. It is a reversible logic gate because, given a certain output, we can obtain its corresponding input. Toffoli gates can be modeled using the billiard ball model [2]. This gate is also known as a controlled-controlled-NOT gate, because it flips the third bit on a 3-bit gate if and only if the first two bits are 1. Fig. 2a) shows Toffoli gate truth table when applied to three bits, and Fig. 2b) shows its circuit representation.

Toffoli gates are crucial for our proposed model, since it will aid us in the construction of the inversion operator C because it can detect if the marked state was found or not (see Section 4.2). The output of the inversion operator C will be zero if the element is not the marked state, and it will be 1 if the marked state was found. These outputs will be used then for further calculations.

Input			Output		
a	**b**	**c**	**a'**	**b'**	**c'**
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

a)



b)

Fig. 2. a) Toffoli gate truth table when applied to three bits. b) Circuit implementation.

4 Realization of Grover's Algorithm using Toffoli gates

Now that we have obtained the neccesary equations from the preceeding section, we now are able to model Grover's algorithm. If we look closely to Eq. 3 and 4, we can see that the operations involved are elementary additions, substractions, multiplications and divisions, such as $\alpha - 4\alpha/N - 2\beta/\sqrt{N}$. Thus, we need basic gates that perform these operations, such that this model will be constructed interconnecting classical logic gates. The elements involved in these basic operations are needed for the model, so we must supply them at the beginning of the execution. We call these elements as the **control data** of the model. Also, the **input data** will consist of the initial element list, which contains the superposition of states. Fig. 3 shows a general diagram for the proposed model.

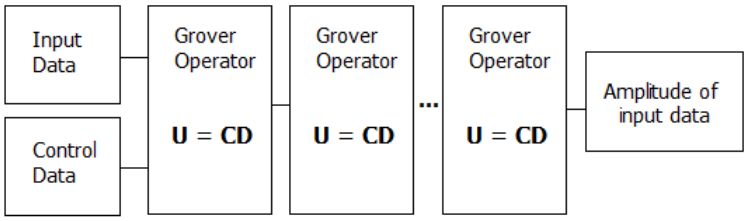


Fig. 3. General shematic of Grover's algorithm procedure. Grover operator is applied the optimum number of iterations in order to increase the amplitude of the marked state, and thus increasing its probability.

Elements from Figure 3 are described below.

Input data: It consists of the element list, which stores the marked state and the colective state.

Control data: These are fixed data that must be supplied to the algorithm before its execution. Control data consists of α, which represents the amplitude of the marked state; β, which stores the amplitude of all the collective set; and finally $1/N$ and $1/\sqrt{N}$, where N represents the total number of states on the database.

Grover operator: This operator was defined in section 2 as $U \equiv DC$.

Input data amplitudes. This is the set of the final amplitudes of the marked state and the collective state. After applying Grover operator a total of $\lfloor \pi\sqrt{N}/4 \rfloor$ iterations to the superposition of states, we expect that the amplitude of the marked state is almost 1.

4.1 Elementary gates

In order to implement Grover's algorithm using non-quantum operations through classical gates, we need to define them first. Such gates constitute the set of basic operators of the model.

Π GATE

This gate requires two input elements. It returns the product of both elements. (See Fig. 4)

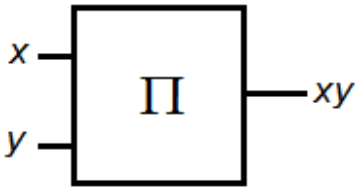


Fig. 4. π-gate: it returns the product of x times *y*

$\sum$ GATE

This gate also requires two input elements. It returns the sum of both elements. (See Fig. 5)

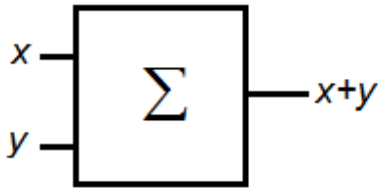


Fig. 5: ∑-gate: it returns the sum of x and *y*.

σ GATE

For this gate, two input data are needed. If the second element is set to 1, the first element will suffer a change of its algebraic sign, otherwise, it will remain unaltered. (See Fig. 6)

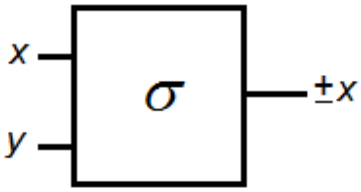


Fig. 6: σ-gate: if *y* is set to 1, this gate will change x's sign. Oterwhise, x will keep unchanged.

4.2 Modeling of Grover operator using Toffoli and elementary gates.

As stated before, Grover operator consists of the application of C operator, followed by D operator, that is, $U \equiv DC$.

Inversion operator **C** is constructed according to the element we are searching for, that is, the marked state. We constructed inversion operator using the binary representation of the element as follows: if the marked state contains zeroes, two NOT gates are put sequentially, otherwise, no gate is needed. These gates are connected by means of a **Toffoli gate**; this gate acts as follows: if every bit is set to 1, it means that the marked state was found, and it will return a 1 as an output, otherwise, it will return a zero, (that is, the marked state was not found).Suppose we want to search for the element 10 on the superposition of states (whose binary representation is 1010 for a 4-bit system). Construction for the inversion operator C using elementary and Toffoli gates is shown in Fig 7.

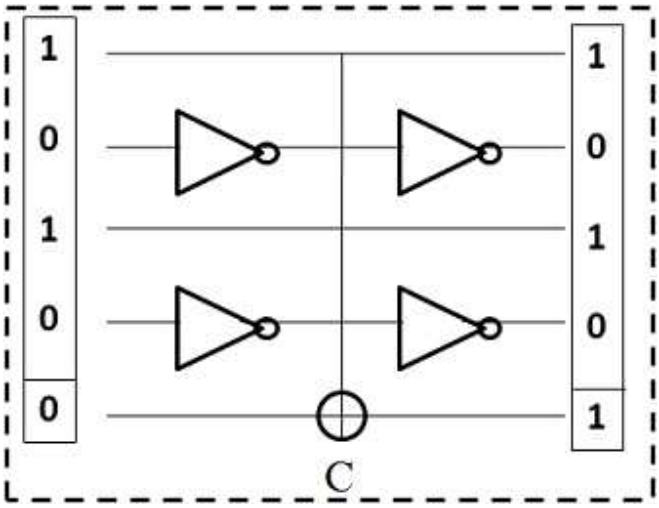


Fig. 7: Construction of inversion operator C for element 10 (which binary representation is 1010). Fifth bit is set to 0, so if every remaining bit at the end was set to 1, this bit will be also switched to 1, meaning that the marked element was found.

Diffusion operator D is needed to increase the amplitude of the marked state. According to Equation 3, this operator can be constructed using elementary gates as shown in Fig 8.

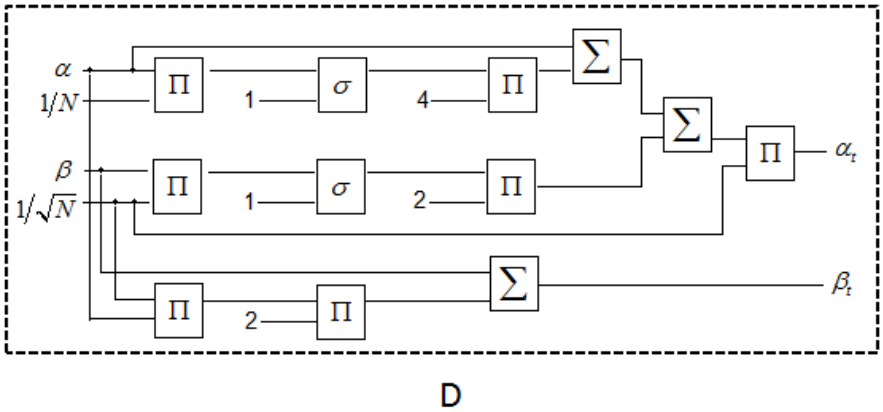


Fig. 8: Diffusion operator. Interconnection of classical gates allows to increase the amplitude of the marked state. This operator along with the inversion operator succesfully simulates Grover operator U = CD. α_t and β_t are variables that will be used on further calculations.

For illustrative purposes, let's take a closer look at the first part of Fig. 8. Eq. 4 shows that the partial chain of operations $\alpha - 4\alpha/N$ is performed. This is done by the section described in Fig. 9, taken from the diagram in Fig. 8

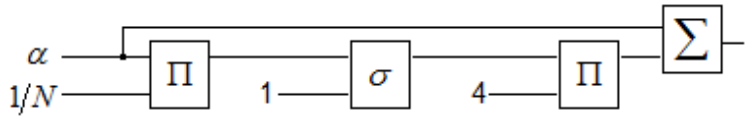


Fig. 9: Carefully following each element on the diagram, we see that the operation $\alpha - 4\alpha/N$ is successfully made.

4.3 Searching for a marked state using the model.

In order to show how the model works, let's look for the element labeled "4" stored in a list of 3 q-bit elements. First, we construct the Inversion operator C for element 4 using a Toffoli gate and NOT gates (see Fig. 10)

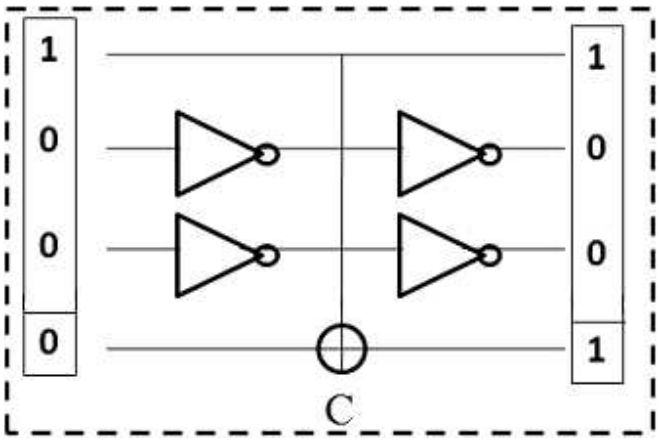


Fig. 10: Construction of inversion operator C for element 4 (which binary representation is 100). Fourth bit is set to 0, so if every remaining bit at the end was set to 1, this bit will be also switched to 1, meaning that the marked element was found.

Toffoli gates are useful on the task of finding the marked state, since it detects wherever the marked state is there or not. Since the entire list consists of 8 elements (from 0 to 7), we must apply the inversion operator for every single element belonging to the list. Fig. 11 shows the results of applying the inversion operator on element 3 and 4.

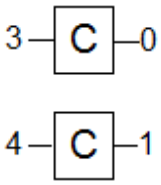


Fig. 11: When inversion operator is applied to element "3", it returns a 0 since it is not the marked state. On the other hand, C operator returns a 1 when applied to the element 4, that is, the marked state was found. This outputs are used on further calculations.

Next step is to increase the amplitude of the marked state using the diffusion operator. Fig. 12 shows how while combining the outputs from both in andversion diffusion operators we increase the amplitude of the marked state, while the collective state remains without change. Notice that only the element 3 and 4 are shown for simplicity, but this has to be done for every element on the list of elements.

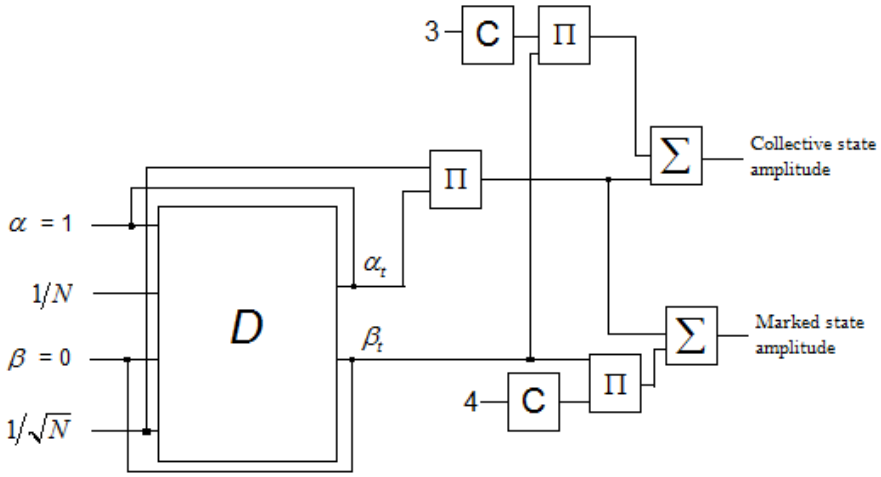


Fig. 12: With the ouput of inversion operator, it is now possible to increment the amplitude of the marked state. To accomplish this, we apply diffusion operator D and combine it with the output of inversion operator C, just like Eqs. 3 and 4 describe. Notice how the outputs from the inversion operator are multiplied by the auxiliary variable β_t and then added to the final result. Since the ouput from the inversion operator is always zero for elements of the collective state, the product is also zero and nothing is added, except from the marked state.

5 Simulation of the model using a high-level programming language.

At this stage, we can program an algorithm that simulates the processes of Grover's algorithm for quantum search. This algorithm will be programmed on a high-level programming language to run some tests, in order to validate our model.

5.1 General Algorithm

We present the general steps needed to simulate correctly Grover's Algorithm. This algorithm is based on Equations 3 and 4. Its advantage consists of the few steps needed to simulate Grover's algorithm.

```
Classic Grover Algorithm

Input
    N: the total number of elements on the system.
    B⃗ = (β_1, β_2, ..., β_{N-1}): the amplitude quoeficient vector
    of the collective state.
    α_M : the amplitude of the marked state.

Output
    B⃗ = (β'_1, β'_2, ..., β'_{N-1}): the changed amplitude quoefi-
    cient vector of collective state.
    α'_M : the new amplitude of the marked state.
Variables
    β:the value of the amplitude of the collective
    state. We can store it on a single variable since
    all the collective state will have the same ampli-
    tude through the entire algorithm.
    coef: an auxiliary variable used to store interme-
    diate values.

Begin
    //Initialize every element of the vector β_i ∈ B⃗ and
    //the amplitude of the marked state to α_M an
    //equally superposition of states
    β ← 1/√N
    α_M ← 1/√N
```

```
Repeat from i = 1 to ⌊π√N/4⌋
  //Apply the operations from Equation 3 to the
  //amplitude of the marked state and the
  //collective state.
  α'_M ← α_M − 4α_M/N − 2β/√N
  β ← 2α/√N + β

  //Store the new normalized marked state to the
  auxiliary variable.
  coef ← α'_M/√N

  //Assign each element β_i ∈ B⃗ its new amplitude.
  β_i ← coef
  //Assign the new amplitude of the marked state.
  α'_M ← coef + β

End Repeat
```

In order to valídate our proposed model, we implemented it using the programming language C++. Data input consists of the number of qubits *n* and the number of desired iterations. It let you choose the optimum number of iterations $\lfloor \pi\sqrt{N}/4 \rfloor$, or any other number of iterations.

5.2. Tests using the optimum number of iterations

Table 1 shows the results for databases built from 1 to 10 qubits, using the optimum number of iterations. The **probability of the marked state** is the probability of obtaining the marked state when a measure of the superposition of states is made. The **probability of the collective state** (or probability of failure) is the probability of obtaining one of the elements of the collective state. Since the probability of the marked state is always same, despite the element searched for, it is unnecessary to list the marked state. This means that if we are looking for the element numbered as 3 on a list of 16 elements, the probability of obtaining it after applying Grover's algorithm is 96.1319%, and that probability will not change if we are looking for the element numbered as 5 on that same list.

Table 1. Probabilities of obtaining the marked state and the collective state embedded into an *n*-qubit system, using the optimum number of iterations.

No. of qubits *n*	**Number of elements N**	**Number of iterations** $\lfloor \pi\sqrt{N}/4 \rfloor$	**Probability of the marked state**	**Probability of the collective sate**
1	2	1	50%	50%
2	4	1	100%	0%
3	8	2	94.5313%	5.4691%
4	16	3	96.1319%	3.8685%
5	32	4	99.9182%	0.0806%
6	64	6	99.6586%	0.3402%
7	128	8	99.5620%	0.4318%
8	256	12	99.9947%	0.0052%
9	512	17	99.9448%	0.0511%
10	1024	25	99.9461%	0.1023%

5.3 Tests using an arbitrary number of iterations

We made tests using an arbitrary number of iterations, instead of the optimum number. Table 2 shows the results for states from 1 to 10 qubit.

Table 2. Probabilities of obtaining the marked state and the collective state embedded into an *n*-qubit system, using an arbitrary number of iterations.

No. of qubits *n*	**Number of elements N**	**Number of iterations**	**Probability of the marked state**	**Probability of the collective sate**
1	2	3	50%	50%
2	4	3	25%	75%
3	8	6	99.9786%	0.0217%
4	16	7	36.4913%	63.5085%
5	32	7	20.9918%	79.0097%
6	64	12	0.0071%	99.9936%
7	128	10	91.9442%	8.0518%
8	256	18	54.1236%	45.7980%
9	512	20	94.2684%	5.7232%
10	1024	27	97.8187%	2.1483%