Filtering Word Senses Using the Least Squares Technique for Word Sense Disambiguation

Jaime Alberto Guzmán-Luna [#1], Sebastián Alonso Gómez Arias [#2]

[#] Department of Computing and Decision Sciences, Faculty of Mines, Universidad Nacional de Colombia
Bl M8A office 308, Medellín, Antioquia, Colombia
[1] jaguzman@unal.edu.co
[2] seagomezar@unal.edu.co

Abstract— the problem of word sense ambiguity (also known as polysemous semantic ambiguity) affects a number of fields of knowledge. Consequently, and because of its relevance, several branches of science have tried to solve it. Polysemous semantic ambiguity occurs when one word has more than one meaning or sense in a sentence. In this paper, the authors propose a coefficient filtering model which uses knowledge-based least squares estimation methods in order to address the polysemous semantic ambiguity of words.

Polysemous semantic ambiguity, word sense filtering, knowledge-based disambiguation techniques, word sense disambiguation.

I. Introduction

Natural language processing is the part of artificial intelligence and computer linguistics that aims to represent human language in a machine-readable format.

Natural language is full of ambiguities (be it lexical, syntactic, semantic, phonetic). Many of these are caused by the vagueness of language itself and the lack of precision with which it is used daily. Furthermore, this problem is magnified by the high amount of possible meanings associated with one single word in a specific language [1].

The author of [2] defines three categories of ambiguity in language: (a) polysemous ambiguity. In this case, one word may have several different meanings depending on the way in which it is used at in a given moment. (b) Syntactic ambiguity, which occurs when the same syntactic structure has different meanings. (c) Referential ambiguity, which makes it necessary for the analysis to go beyond the boundaries of the sentence in order to determine the referential antecedents of the pronouns.

A word that has more than one sense is called polysemous, i.e. a word having several senses, the most appropriate of which can only be inferred by looking at the particular context in which it is being used. It is worth noting, however, that in many cases it is difficult even for humans to discern the appropriate sense [3]. Polysemous ambiguity is most common with verbs, adjectives and nouns. In this paper, the authors propose a coefficient filtering model which uses knowledge-based disambiguation methods to deal with the polysemous semantic ambiguity found in the senses of the words.

This paper is structured as follows: the next section introduces the theoretical framework constituting the basis of this study. Section three introduces the knowledge-based disambiguation methods used in the model proposed. Section four shows the proposed disambiguation model in detail. Section five shows the experiments and results with which the model was assessed. Finally, section six presents the conclusions and future work arising from this study.

II. Theoretical Framework

There are two variants in the field of word sense disambiguation (WSD): (i) One Word. In this approach the system must disambiguate one word per sentence. This variant is used mainly for supervised methods in which all the context is used as a trained corpus and the system is tested using one word per sentence. (ii) All Words: in this approach the system must disambiguate all the types of words in the text (nouns, adjectives, verbs and adverbs). This variant uses knowledge-based methods.

Knowledge is an essential component of WSD. Knowledge bases provide information relevant for associating a sense with words. These knowledge sources may be text corpora, dictionaries, thesauri, glossaries and ontologies. These sources can be classified as structured or unstructured sources. (i) Some structured sources include: thesauri, machine-readable dictionaries and ontologies. (ii) Some unstructured sources include: corpora, common sense lists, annotated corpora, RAW corpora. It is worth noting that a corpus is large and structured set of texts. They are useful for performing statistical analyses, testing hypotheses, checking occurrences and validating linguistic rules belonging to a specific area of language.

According to [4], word sense disambiguation methods may fall into one of the following three categories.

A. *Supervised Methods*

Supervised disambiguation methods utilize machine learning techniques in order to learn to infer rules based on previously trained corpora (corpora are previously tagged and disambiguated texts). The goal being processing and disambiguating new texts based on the rules learned.

B. *Unsupervised Methods*

These disambiguation methods use only the context of a word in order to disambiguate the word in question. They are based on the idea that the same sense of a word will have similar words in its vicinity.

C. *Knowledge-based Methods*

These methods seek to exploit the knowledge contained in the thesauri, ontologies, dictionaries, etc. in order to infer the sense of a word in a given context.

Knowledge-based methods employ different sources of knowledge (e.g. WordNet, Multilingual Central Repository, etc.). These knowledge bases are used as lexical databases which group words into sets of synonyms called synsets and provide short, general definitions of each of them. These also store semantic relations between the synonym sets. Their purpose is twofold: they seek to produce what would be a combination of a dictionary and a thesaurus whose usage is more intuitive while supporting automatic text analysis through artificial intelligence applications. The hypernym/hyponym relationship between synsets could be interpreted as a specialization/generalization relationship between conceptual categories. This makes it possible to interpret and use WordNet as an ontology in computer science [5].

TABLE I summarizes some of the most widely known techniques for solving word sense ambiguity along with the type of disambiguation method to which they belong.

TABLE I
Summary of the approaches and techniques used to attack word sense ambiguity.

Authors	Technique	Disambiguation Method
[6]	Decision lists	Supervised
[7]	Decision trees	Supervised
[8]	Bayesian networks	Supervised
[9]	Neural networks	Supervised
[10]	Vector machines	Supervised
[11]	Context grouping	Unsupervised
[12]	Word grouping	Unsupervised
[13]	Concurrence graphs	Unsupervised
[14]	Simplified Lesk	Knowledge-based
[15]	Adapted Lesk	Knowledge-based
[16]	MFS	Knowledge-based
[17]	UKB	Knowledge-based

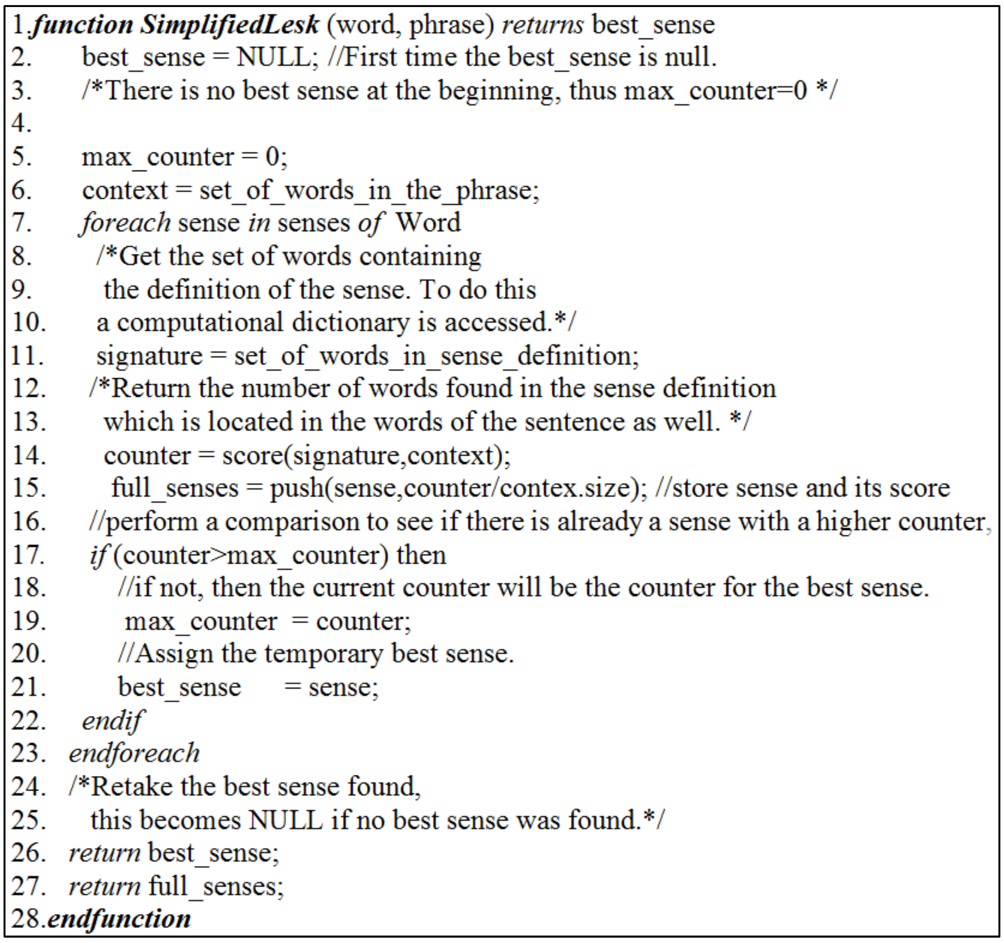
From [4] it can be deduced that knowledge-based methods are more frequent in many cases because they are versatile, easy to adapt and implement for virtually any case in any field of knowledge.

III. KNOWLEDGE-BASED DISAMBIGUATION METHODS

The following are detailed descriptions of the algorithms shown in table I. These algorithms are used in the proposed coefficient filtering model.

A. *Simplified Lesk*

The simplified version of this algorithm compares the dictionary definition or examples for an ambiguous word with the other terms contained in its vicinity [14]. Fig. 1 shows the pseudocode for the *Simplified Lesk* algorithm used to disambiguate words in a sentence in this study's model.



```
function SimplifiedLesk (word, phrase) returns best_sense
    best_sense = NULL; //First time the best_sense is null.
    /*There is no best sense at the beginning, thus max_counter=0 */

    max_counter = 0;
    context = set_of_words_in_the_phrase;
    foreach sense in senses of Word
      /*Get the set of words containing
       the definition of the sense. To do this
      a computational dictionary is accessed.*/
      signature = set_of_words_in_sense_definition;
     /*Return the number of words found in the sense definition
       which is located in the words of the sentence as well. */
       counter = score(signature,context);
        full_senses = push(sense,counter/contex.size); //store sense and its score
     //perform a comparison to see if there is already a sense with a higher counter,
     if (counter>max_counter) then
         //if not, then the current counter will be the counter for the best sense.
          max_counter  = counter;
         //Assign the temporary best sense.
         best_sense      = sense;
    endif
  endforeach
  /*Retake the best sense found,
     this becomes NULL if no best sense was found.*/
  return best_sense;
  return full_senses;
endfunction
```

Fig. 1. The Simplified Lesk algorithm used in this study. Source: the authors.

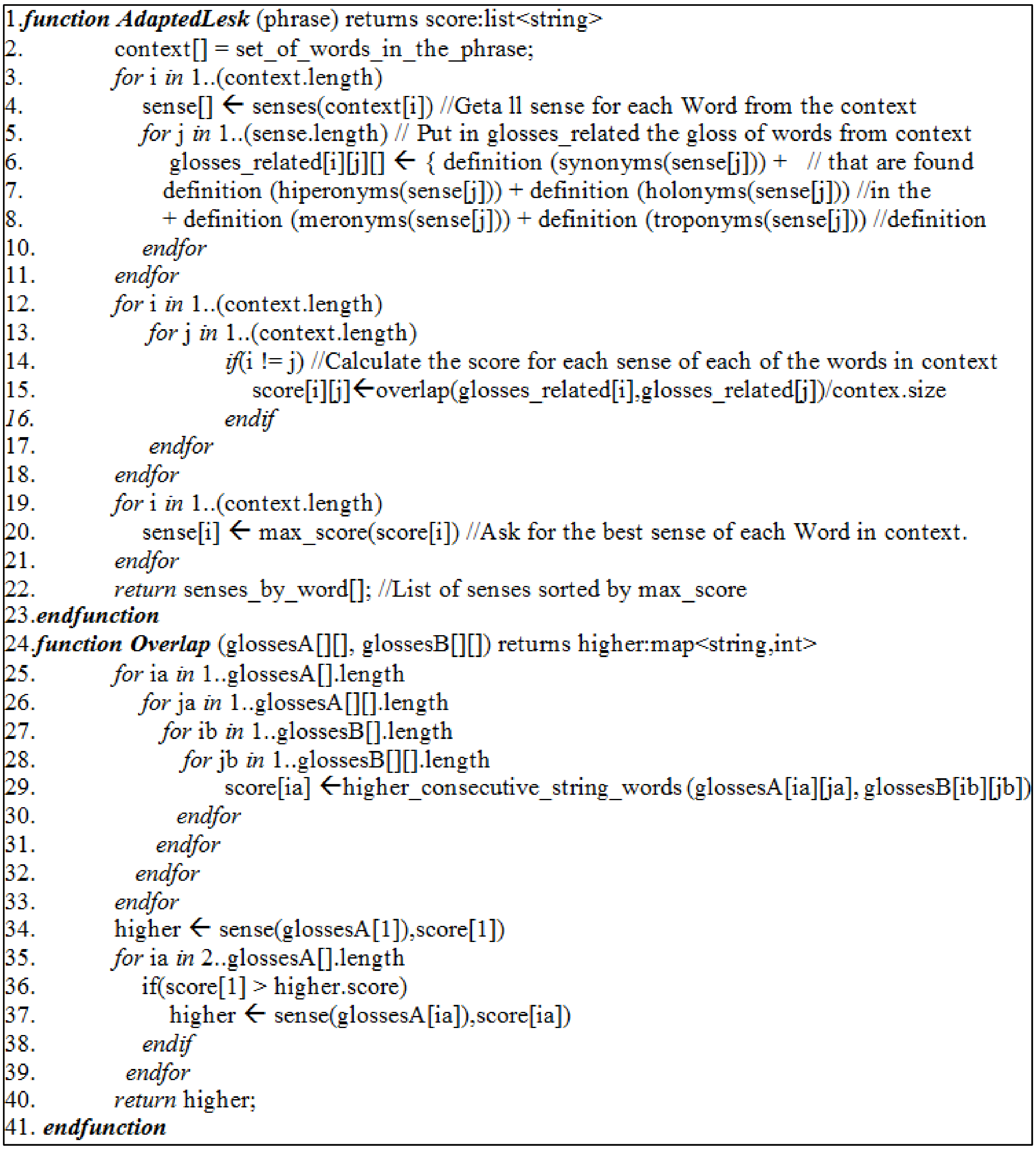
In this algorithm, *word* is the word to be disambiguated and *phrase* are the words making up the sentence where the ambiguous *word* is located. *Context* contains the functional words (adjectives, verbs, adverbs and nouns) which accompany the *word* contained in *phrase*. *senses* is the set of possible meanings for a Word. These are provided by a knowledge base. *best_sense* is the sense with the best *counter* and *full_senses* is the array containing the (*sense*, *counter*) tuple along with the sense and the coefficient for that sense. The coefficient, *counter*, is calculated via the *score* function in the following way:

$$score = \ |\, \#\, words\ present\ in\ the\ word\ definition \cap \#\, words\ in\ the\ context\, | \quad (1)$$

Finally, the algorithm returns the *best_sense* and *full_senses* variables.

B. Adapted Lesk.

Authors such as [15] propose a modification to the original *Simplified Lesk* algorithm because it has limitations since it only compares words in the vicinity (*context*) of the word to be disambiguated (*word*) against the words in the definitions of *word* (*senses*). This frequently causes poor disambiguation when the context or definitions (*senses*) of the word are limited [18].



```
function AdaptedLesk (phrase) returns score:list<string>
    context[] = set_of_words_in_the_phrase;
    for i in 1..(context.length)
        sense[] ← senses(context[i]) //Geta ll sense for each Word from the context
        for j in 1..(sense.length) // Put in glosses_related the gloss of words from context
            glosses_related[i][j][] ← { definition (synonyms(sense[j])) +   // that are found
            definition (hiperonyms(sense[j])) + definition (holonyms(sense[j])) //in the
            + definition (meronyms(sense[j])) + definition (troponyms(sense[j])) //definition
        endfor
    endfor
    for i in 1..(context.length)
        for j in 1..(context.length)
            if(i != j) //Calculate the score for each sense of each of the words in context
                score[i][j]←overlap(glosses_related[i],glosses_related[j])/contex.size
            endif
        endfor
    endfor
    for i in 1..(context.length)
        sense[i] ← max_score(score[i]) //Ask for the best sense of each Word in context.
    endfor
    return senses_by_word[]; //List of senses sorted by max_score
endfunction
function Overlap (glossesA[][], glossesB[][]) returns higher:map<string,int>
    for ia in 1..glossesA[].length
        for ja in 1..glossesA[][].length
            for ib in 1..glossesB[].length
                for jb in 1..glossesB[][].length
                    score[ia] ←higher_consecutive_string_words (glossesA[ia][ja], glossesB[ib][jb])
                endfor
            endfor
        endfor
    endfor
    higher ← sense(glossesA[1]),score[1])
    for ia in 2..glossesA[].length
        if(score[1] > higher.score)
            higher ← sense(glossesA[ia]),score[ia])
        endif
    endfor
    return higher;
endfunction
```

Fig. 2. The adapted Lesk algorithm using the proposed model. Source: the authors.

Given this [15], the *adapted Lesk* algorithm is proposed. This algorithm takes the definitions of the words (*senses*) stored in *context*, - which is composed of the functional words (adjectives, verbs, adverbs and nouns) in the sentence (*phrase*) - and compares them (usually the comparison is done using pairs of words) with the definitions of the synonyms (similar senses), hypernyms (more general senses), hyponyms (more specific senses), meronyms (senses that are part of other senses), holonyms (senses that act as the whole of other senses) and troponyms (verbal hyponyms)of each word in the context. Fig. 2 shows the *adapted Lesk* algorithm in detail.

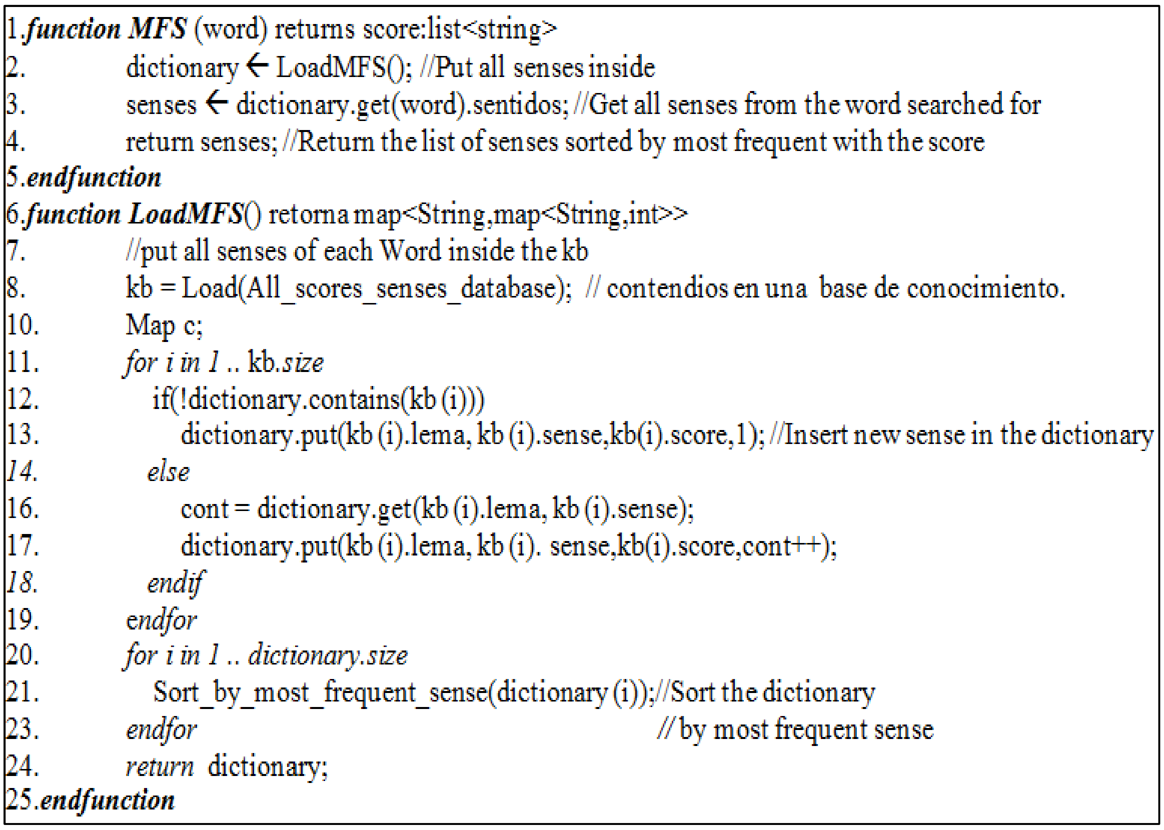
The comparison is performed using the *overlap* function, which finds the length of the largest string of consecutive functional words (adjectives, verbs, adverbs and nouns) found in the pairs of definitions.

Finally, the algorithm will return the *senses_by_word* which contains the senses for each word in the *phrase* with its respective score.

C. Most Frequent Sense (MFS)

The most frequent sense (MFS) is a semantic polysemy disambiguation algorithm which finds the most used sense for each word. This method uses an annotated knowledge base which stores the senses for each word and a frequency coefficient that determines which *sense* is the most commonly used for a given *word* [16]. Fig. 3 shows the Most Frequent Sense algorithm in detail.

This algorithm first preloads the knowledge base *kb*. Then, it takes a *word*, and queries the dictionary variable in search of its corresponding *senses*, which are sorted by *score*. Finally, the Most Frequent Sense algorithm returns, for one *word*, the list of senses and its *score* associated by means of the *senses* array.



```
1.function MFS (word) returns score:list<string>
2.        dictionary ← LoadMFS(); //Put all senses inside
3.        senses ← dictionary.get(word).sentidos; //Get all senses from the word searched for
4.        return senses; //Return the list of senses sorted by most frequent with the score
5.endfunction
6.function LoadMFS() retorna map<String,map<String,int>>
7.        //put all senses of each Word inside the kb
8.        kb = Load(All_scores_senses_database);  // contendios en una  base de conocimiento.
10.       Map c;
11.       for i in 1 .. kb.size
12.          if(!dictionary.contains(kb (i)))
13.             dictionary.put(kb (i).lema, kb (i).sense,kb(i).score,1); //Insert new sense in the dictionary
14.          else
16.             cont = dictionary.get(kb (i).lema, kb (i).sense);
17.             dictionary.put(kb (i).lema, kb (i). sense,kb(i).score,cont++);
18.          endif
19.       endfor
20.       for i in 1 .. dictionary.size
21.          Sort_by_most_frequent_sense(dictionary (i));//Sort the dictionary
23.       endfor                                        // by most frequent sense
24.       return  dictionary;
25.endfunction
```

Fig. 3. Most Frequent Sense, the algorithm used in this study's model. Source: the authors.

D. UKB

The method consists of covering the length of any lexical knowledge base's graph (LKB). Given a lexical knowledge base (LKB), a non-directed graph is built *$G=(V,E)$* , where the nodes are the LKB's concepts, *vi*, and each relation between concepts v_i and v_j is represented by a non-directed vertex $e_{i,j}$. Given an input text, the list W_i *(i=1…m)* is extracted from functional words (nouns, verbs, adjectives, and adverbs), which have an entry in the dictionary and, thereby, can be related to concepts in the LKB. Let $senses_i$ = *{v1,…,vi$_m$}* be the i_m senses related to the word Wi in the knowledge base graph. Thus, polysemous words are associated with various concepts; while monosemous words are associated with a single concept. The result of the disambiguation process consists in that each sense within the list called $senses_i$ is given a score. Thus, for each word to be disambiguated, the concept that has the highest score in *G* is select. Fig. 4 shows in detail the implementation of the UKB algorithm.

```
1.function loadUKB(phrase) return Map<String,Map<String,int>>
2.      read kb; //tagged knowledge base with the sense for each words
3.      Map dictionary;
4.      For i in 1 .. kb.size {
5.        For j in kb(j) - context.. kb(j)+ context {
6.              sensesj [] ← synsets(kb(j));
7.              Gkb ← relevant_concepts(kb(j)- context..kb(j)+ context);
8.              For k in 1 .. sensesj.size{
9.                    shortest_path ← BFS(senses(k),Gkb);
10.                   Gd.add(shortest_path); //Disambiguation graph
11.             endfor
12.       endfor
13.       score = PageRank(Gd);
14.       dictionary.add(kb(i),score); //Add the sense and the score for each word in the sentence
15.     endfor
16.     return dictionary;
17.endfunction
18.function BFS(Graph G, Node a): List<Node>
19.     foreach (vertex u: u.explorado = falso);
20.     a. explored = falso;
21.     q = new list;
22.     q.push(a);
23.     while (q ≠ ∅)
24.       u = q.pop();
25.       foreach vertex (u, v)
26.         if (v.explored = false) then
27.           v. explored = true;
28.           q.push(v);
29.         endif
30.       endforeach
31.     endwhile
31.endfunction
```

Fig. 4. Most Frequent Sense, the algorithm used in this study's model. Source: the authors

The score is estimated on the basis of adaptations done by [19] to the PageRank . The main idea behind this method is to draw a subgraph, GKB, of the LKB whose vertices and relations are particularly relevant for that specific context. Such a graph is called disambiguation subgraph GD, and is built as follows: for each word Wi in the input context, and for each concept vi ∈ Conceptsi, a breadth-first search (also known as BFS) is conducted on GKB, beginning with node vi. Each execution of the BFS calculates the minimum distance of paths between vi and the remaining concepts in GKB. Specifically, the interest is to find out the minimum path distance between vi and the concepts related to the remaining words in the context. Let vdpvi be the set of those shorter paths.

$$v_j \epsilon \bigcup_{j \neq i} senses_j \quad (2)$$

The calculation of the shortest path using BFS is repeated for each concept belonging to each word within the context while saving the respective vdp_{vi}. At the end, a set of minimum distance paths is obtained, each with different concepts as their starting point. Thus, the G_D disambiguation graph is the union of the vertices and edges of the shorter paths:

$$G_D = \bigcup_{i=1}^{m} \left\{ mdp_{v_j} / v_j \epsilon senses_i \right\} \quad (3)$$

The G_D disambiguation graph is a subgraph of the original graph G_{KB}. In this way, one can say that it catches the more relevant concepts and relations in the knowledge base for a particular input context. Based on this graph, the PageRank can be calculated. The main idea of the PageRank is that provided that there is always a link between vi and vj within a graph, a vote of node i to node j is produced, thereby increasing node j's score. Additionally, the magnitude of this vote depends on node i's score: the more important node i, the higher the score for node j.

Eventually, the UKB algorithm will give back the dictionary variable, which contains the word senses for each word in *phrase* and their respective associated score.

The disambiguation methods presented in this section return, for each word W, in a phrase F, an array containing the senses for W along with their respective Score. For instance, when providing the four algorithms presented in this study with the sentence: "Los conos de pino cuelgan en un árbol", where we are trying to disambiguate the word "Pino", we would get the following output. See TABLE II.

TABLE II. Disambiguation example using the word "Pino" in the sentence "Los conos de pino cuelgan en un árbol" using the methods mentioned in this paper.

Possible sense for "Pino"		**Disambiguation methods used**			
Definition	**WordNet Identifier**	**Simplified Lesk (score)**	**Adapted Lesk (score)**	**Most Frequent Sense (score)**	**UKB (score)**
a coniferous tree	11608250-n	0.125	0.375	0.332582	0.016422
any gymnospermous tree or shrub bearing cones	13108841-n	0.250	0.625	0	0
the act of supporting yourself by your hands alone in an upside down position	00436187-n	0	0.125	0.584651	0.013467
(gymnastics) an exercise designed to develop and display strength and agility and balance (usually performed with or on some gymnastic apparatus)	00435778-n	0	0.125	0	0
straight-grained durable and often resinous white to yellowish timber of any of numerous trees of the genus Pinus	11608885-n	0.125	0.375	0.082767	0.014204
the hard fibrous lignified substance under the bark of trees	15098161-n	0.125	0.375	0	0

IV. Model Proposed for Word Sense Filtering

The algorithm proposed for the process of polysemous disambiguation of a word is composed of the knowledge-based disambiguation techniques discussed in section three of this paper. This group of algorithms returns a list of senses with their respective *score* for a given *word* contained in a *phrase*.

Those coefficients are then analyzed by a superior mechanism that validates the vector of coefficients obtained by each algorithm and processes it using a coefficient filtering technique which employs a linear least squares regression in order to obtain a final score which guarantees a better sense by uniting the four techniques used. The following figure details the process:

The following is a detailed description of the stages of the natural language processing algorithm presented in Fig. 5.

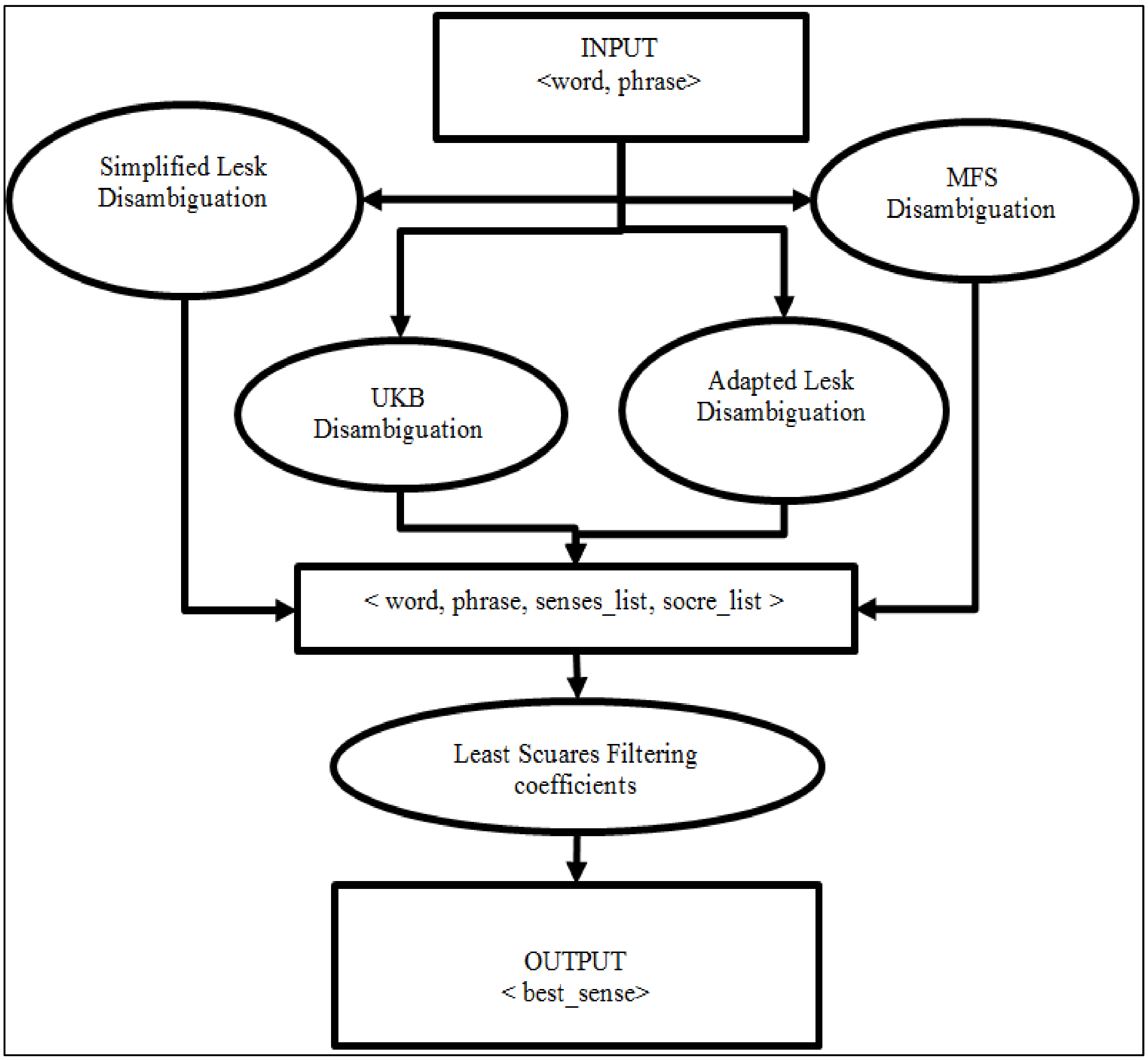

INPUT
<word, phrase>
Simplified Lesk Disambiguation
MFS Disambiguation
UKB Disambiguation
Adapted Lesk Disambiguation
< word, phrase, senses_list, socre_list >
Least Scuares Filtering coefficients
OUTPUT
< best_sense>

Fig 5. Stages of the disambiguation algorithm. Source: the authors.

A. *Coefficient Filtering*

The first step in coefficient filtering is reading the first sense returned by each of the methods described in the previous section: MFS, UKB, Simplified Lesk and Adapted Lesk. Then, and since the goal is to obtain the best sense, *i*, a coefficient filtering model using the least squares method shall be used which integrates both methods. For coefficient filtering, a least square filter was implemented for the estimation of a scalar value [20]; thus, the model finds the coefficient $\partial Score_i$ for each sense, *i*, of a word, *w*. Four approaches shall be used (one per disambiguation method) for each *sense i* of a *word*. The equation used is:

$$\partial Score_i = \frac{\sum_{j=1}^{n=4}(x_{ij}-\overline{w}_i)^2}{n=4} \quad (4)$$

Where $\partial Score_i$ will show the coefficient resulting from the least squares filtering for sense *i*, x_{ij} represents the coefficient for sense *i* in disambiguation method *j*, and n represents the amount of coefficients obtained for our model, which will always be 4, as four disambiguation methods were deployed. Additionally, $\overline{w_i}$ represents the mean of the coefficients from disambiguation methods *j*, which is given by the following equation:

$$\overline{w}_i = \frac{\sum_{j=0}^{n=4} x_{ij}}{n=4} \quad (5)$$

Upon obtaining the $\partial Score_i$ for each sense *i*, the next step is to compare those coefficients in order to establish each sense's definite and integrated score. Therefore, the best sense *i* of a *word* should be the sense having the highest $\partial Score$. By using table III below, an example of this coefficient filtering for the word "pino" in the sentence discussed in table II is shown.

TABLE III.
Coefficient filtering for one sense i.

		Disambiguation method j					
		1	2	3	4		
Sense i	**WordNet identifier**	**Simplified Lesk**	**Adapted Lesk**	**MFS**	**UKB**	$\bar{w}_i$	$\partial Score_i$
1	11608250-n	0.125	0.375	0.332582	0.016422	0.212251	0.02173213
2	13108841-n	0.125	0.625	0	0	0.1875	<u>0.06640625</u>
3	00436187-n	0	0.125	0.584651	0.013467	0.1807795	0.05672456
4	00435778-n	0	0.125	0	0	0.03125	0.00292968
5	11608885-n	0.125	0.375	0.082767	0.014204	0.1492427	0.01855213
6	15098161-n	0.125	0.375	0	0	0.125	0.0234375

As one can see in TABLE III, the best sense for the word “pino” in the sentence “Los conos de pino cuelgan en un árbol” would be sense 2 (i=2) which has a $\partial Score_2 = 0{,}06640625$, which is the maximum $\partial Score$.

V. Experiments and Results

Tests were conducted on the official text [21] used for the SENSEVAL-3 competition [22]. This text consists of a tagged corpus on which words must be disambiguated; each word having a set of defined senses. The results shown in this chapter have been obtained using, as client-server, a computer with an Intel Core 2 Quad Q8200 (2.33 GHz) processor and DDR2 3544 MB 800 MHz. The OS was Ubuntu 12 LTS 32 Bits.

For the assessment, the following metrics were used: Precision, recall and coverage. TABLE IV describes these three metrics, where T is the total amount examples in the set, V is the number of instances accurately classified, and U is the number of classified examples whose list of potential senses is not empty (no matter if they are correctly classified or not).

TABLE IV.
Metrics used for results assessment

Metrics	Equation	Definition
Coverage (C)	$C = \frac{U}{T}$	Coverage is the percentage of words to which the WSD system has responded (number of instances for which the WSD system proposed an answer / total number of instances in the testset)
Recall (R)	$V = \frac{V}{T}$	Recall is the percentage of words being accurately disambiguated within the set of all test words (number of instances accurately disambiguated / total number of instances in the test set).
Precision (P)	$P = \frac{V}{U}$	Precision is the percentage of words accurately disambiguated by the WSD system, given a reference corpus (number of instances accurately disambiguated by the WSD system/number of instances for which the WSD system proposed an answer).
F1	$F1 = \frac{2 * P * R}{P + R}$	The F1 coefficient is a value between 0 and 1 that indicates task effectiveness; 1 being the value for the perfect system. This metric groups precision and scope in order to determine how the model's general process was.
Performance	$\gamma = \frac{T}{Execution\ time\ (seconds)}$	Execution time of the given task in seconds.

TABLE V. shows the results obtained for each metric when the number of words introduced for disambiguation varies.

TABLE V
Results for the metrics used for assessing the results

# number of words entered	Precision %	Recall %	F1	Coverage %	Performance (seconds)	Performance per word (seconds)
9	66.67%	66.67%	0.6667	100.00%	3.23	0.35888889
19	55.56%	52.63%	0.5405	94.74%	7.84	0.41263158
35	60.61%	57.14%	0.5882	94.29%	16.05	0.45857143
37	55.88%	51.35%	0.5352	92.89%	15.31	0.41378378
39	55.88%	48.72%	0.5205	90.18%	16.98	0.43538462
47	58.14%	53.19%	0.5556	91.49%	20.03	0.42617021
81	51.43%	44.44%	0.4768	86.42%	43.67	0.5391358
113	59.78%	48.67%	0.5366	84.42%	63.31	0.56026549
Average number of words entered	**Average precision**	**Average recall**	**Average F1**	**Average coverage**	**Average performance**	**Average performance per word**
47.5	57.99%	52.85%	0.5530	91.80%	23.3025	0.45060397

Fig. 6 shows the results obtained when performing the tests with a different amount of entered words.

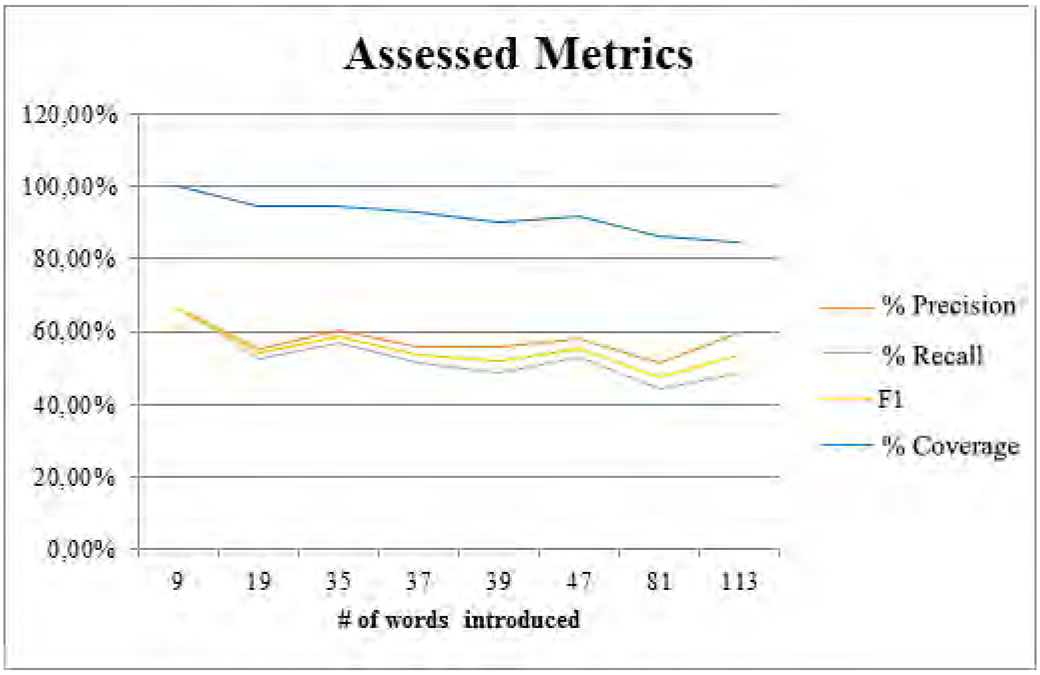


Fig 6. Stages of the disambiguation algorithm. Source: the authors.

The mean coverage of the disambiguation process was about 91%, which indicates that the system gives us a response close to the best answer almost invariably. When measuring recall with a different amount of words, Recall yields an average of 52%, thus suggesting that the system returned a correct answer in more than half of the cases. As for precision, the model properly disambiguated 57% of the answers given on average. The F1 coefficient is a value between 0 and 1 that indicates task effectiveness; 1 being the value for the perfect system. This metric groups precision and scope in order to determine how the model's general process was. Therefore, the model has a 55% effectiveness when performing the tasks assigned.

Fig. 7 shows the results obtained when conducting performance tests versus number of words.

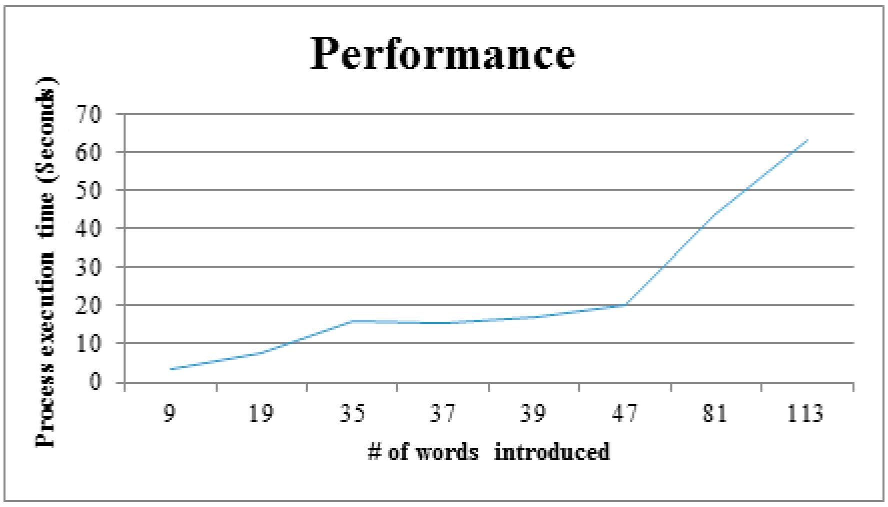


Fig 7. Results for performance per words, R $_{p.}$ Source: the authors.

For this model, we can see that the system speed is proportional to the number of entries it must process. However, we found, in a single case, an outlier indicating that performance increased in spite of entries increasing too. This might be due to connectivity issues or to the low complexity of the words to be disambiguated at the time.

Finally, TABLE VI shows a comparison against the unsupervised methods presented at the SENSEVAL 3 meeting [22]. Results are comparable, since the experiments were conducted using the same tagged corpus [21].

TABLE VI.
Comparison with current approaches

	Proposed model	**Non-supervised disambiguation method used at SENSEVAL-3**
Coverage	91.80%	91.54%
Precision	57.99%	58.20%
Recall	52.85%	54.77%

VI. CONCLUSION AND FUTURE WORKS

This paper presented current *word sense disambiguation* techniques and focused on knowledge-based techniques, due to their ease of use, their performance and coverage in disambiguation tasks. There was a distinction between i) supervised methods, ii) unsupervised methods, and iii) knowledge-based methods. Also a theoretical model for coefficient filtering was presented, which shows knowledge-based disambiguation methods and gives room to a novel disambiguation technique which brings together knowledge-based techniques to generate enhanced semantic disambiguation by using coefficient filtering through least squares.

For the experiments conducted, an average precision of 57.99% was obtained, as well as an average coverage of 91.80%, and a recall of 52.85% for the proposed model. Generally, it can be stated that regarding coverage the model proposed shows better performance than the approaches advanced at SENSEVAL-3. However, precision is alike and recall is slightly lower. Thus, the combination of the MFS, UKB, Simplified and Adapted Lesk techniques linked together through a least square coefficient filtering helps enhance coverage.

As future works, the authors put forward the implementation of new coefficient filters that make it possible to obtain the best word sense more precisely. The authors also suggest incorporating new disambiguation algorithms allowing to improve the recall and precision of the coefficient filtering model. Finally, as a future work, the authors propose improving the process execution time of the proposed model in order to optimize it.

ACKNOWLEDGMENT

To Colciencias, through their call for young researchers 2013 “Jovenes Investigadores 2013” under the project: “a model for solving ambiguity in word senses in order to enhance quality in a software requirements elicitation architecture” “Un modelo de resolución de ambigüedad de sentidos de palabras para mejorar la calidad de resultados en una arquitectura de educción de requisitos de software”. Hermes code: 21481 Universidad Nacional de Colombia at Medellin.

REFERENCES

[1] A. S. Cueto, «Resolución de la Ambigüedad Semántica de las palabras mediante Modelos de Probabilidad de Máxima Entropıa», 2004.
[2] A. M. Guijarro, «Resolución de la ambigüedad semántica mediante métodos basados en conocimiento y su aportación a tareas de PLN», 2009.
[3] E. Agirre y G. Rigau, «Word sense disambiguation using conceptual density», en Proceedings of the 16th conference on Computational linguistics-Volume 1, 1996, pp. 16–22.
[4] R. Navigli, «Word sense disambiguation: A survey», ACM Computing Surveys (CSUR), vol. 41, n.o 2, p. 10, 2009.
[5] V. Snasel, P. Moravec, y J. Pokorny, «WordNet ontology based model for web retrieval», en Web Information Retrieval and Integration, 2005. WIRI'05. Proceedings. International Workshop on Challenges in, 2005, pp. 220–225.
[6] R. L. Rivest, «Learning decision lists», Machine learning, vol. 2, n.o 3, pp. 229–246, 1987.
[7] J. R. Quinlan, «Induction of decision trees», Machine learning, vol. 1, n.o 1, pp. 81–106, 1986.
[8] G. Escudero, L. Màrquez, G. Rigau, y J. G. Salgado, «On the portability and tuning of supervised word sense disambiguation systems», 2000.
[9] G. Cottrell y J. Allen, «A connectionist approach to word sense disambiguation», 1985.
[10] H. T. Ng, «Getting serious about word sense disambiguation», en Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How, 1997, pp. 1–7.
[11] H. Schutze, «Dimensions of meaning», en Supercomputing'92., Proceedings, 1992, pp. 787–796.
[12] D. Lin, «Automatic retrieval and clustering of similar words», en Proceedings of the 17th international conference on Computational linguistics-Volume 2, 1998, pp. 768–774.
[13] D. Widdows y B. Dorow, «A graph model for unsupervised lexical acquisition», en Proceedings of the 19th international conference on Computational linguistics-Volume 1, 2002, pp. 1–7.
[14] M. Lesk, «Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone», en Proceedings of the 5th annual international conference on Systems documentation, 1986, pp. 24–26.
[15] S. Banerjee y T. Pedersen, «An adapted Lesk algorithm for word sense disambiguation using WordNet», en Computational linguistics and intelligent text processing, Springer, 2002, pp. 136–145.
[16] J. Preiss, J. Dehdari, J. King, y D. Mehay, «Refining the most frequent sense baseline», en Proceedings of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions, 2009, n.o June, pp. 10-18.
[17] L. Padró, S. Reese, E. Agirre, y A. Soroa, «Semantic services in freeling 2.1: Wordnet and ukb», 2010.
[18] S. Patwardhan, S. Banerjee, y T. Pedersen, «Using measures of semantic relatedness for word sense disambiguation», en Computational linguistics and intelligent text processing, Springer, 2003, pp. 241–257.
[19] E. Agirre y A. Soroa, «Using the Multilingual Central Repository for Graph-Based Word Sense Disambiguation.», en LREC, 2008.
[20] C. L. Lawson y R. J. Hanson, Solving least squares problems, vol. 161. SIAM, 1974.
[21] ed Pedersen - Sense Tagged Text». [En línea]. Disponible en: http://www.d.umn.edu/~tpederse/data.html. [Accedido: 28-oct-2014].
[22] R. Mihalcea, T. Chklovski, y A. Kilgarriff, «The Senseval-3 English lexical sample task», en Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, 2004, pp. 25–28.

AUTHOR PROFILE

Jaime Alberto Guzmán-Luna: Civil Engineer, Universidad Nacional de Colombia, Medellín, Colombia. Specialist in Educational Communication, Universidad de Pamplona, Pamplona, Colombia. Master´s Degree in Systems Engineering, Universidad Nacional de Colombia. Doctor in Systems Enginnering, Universidad Nacional de Colombia. Associate Professor at Universidad Nacional de Colombia, Medellín, Colombia. E-mail: jaguzman@unal.edu.co.

Sebastián Alonso Gómez Arias: Systems and Infomation Technologies Engineer, Universidad Nacional de Colombia, Medellín, Colombia. Master´s Degree Candidate in Systems Engineering at Universidad Nacional de Colombia. E-mail: seagomezar@unal.edu.co.