Automatically Locating Results to Support Systematic Reviews in Software Engineering

José Alberto S. Torres[1], Daniela S. Cruzes[2], Laís do N. Salvador[3]

[1] Regional Center of Telematics, DPRF, Salvador/BA, Brazil
alberto@torres.eti.br
[2] Dept. of Computer and Information Science (IDI), NTNU, Trondheim, Norway
dcruzes@idi.ntnu.no
[3] Dept. of Computer Science, UFBA, Salvador/ BA, Brazil
laisns@dcc.ufba.br

Abstract. Background: Systematic Reviews are extremely dependent on human effort and, therefore, costly and time consuming. Some authors in Software Engineering are starting to research the use of computer support to reduce human labor in some tasks of the process. Aim: Define a method for automatic location of sentences that describe the results in an unstructured scientific paper aiming to reduce the human effort. Method: Three sentence classification methods were analyzed and a new sentence classification method was proposed and tested with the same input set used in the other methods. Results: The method proposed in this work reached rates ranged between 60% and 72% of recall of the sentences describing results of the papers. Conclusions: The proximity between the recall rates found in automatic tests conducted with the proposed method and in a test with humans confirms the feasibility of this technique for automating part of the process.

Keywords: Evidence-Based Software Engineering; Systematic Review; Information Retrieval.

1 Introduction

For decades people have known the gaps between research evidence and clinical practice, and the consequences in terms of ineffective or even harmful decision making. Evidence-based medicine came to help in overcoming this problem in decision-making. This methodology is about asking questions, finding and evaluating relevant data and use such information in clinical practice in order to assist the work of doctors [21]. The success achieved by evidence-based medicine, especially after the 80's, has led other areas that provide service to the general public to adopt this paradigm, such as psychiatry, nursing and, more recently, the Software Engineering area [3].

Evidence-Based Software Engineering (EBSE) aims to improve decision-making on the development and maintenance of software through the integration of current best research evidence with practical experience [13][6]. EBSE also aims to provide

knowledge about when, how and in what context technologies, processes, tools and methods are most appropriate for the practice of software engineering. In this context, the Systematic Review (SR) has provided mechanisms to identify and aggregate research evidence providing a full and fair assessment of the state of evidence related to a particular topic of interest [13][6]. The process of conducting this type of study is guided by a rigid and well-defined sequence of methodological steps, which follow a strict protocol defined before the beginning of activities [12].

In the last years, most strongly after the publication of the seminal papers on EBSE [13][6] and the procedures for undertaking systematic reviews [12], we have noticed an improvement from the use of Systematic Reviews by researchers in Software Engineering. Zhang and Babar [29] investigated the adoption and use of the Systematic Review in Software Engineering and discovered that the vast majority of researchers in SE consulted were convinced of the value of using a systematic methodology and rigorous literature reviews. In this interview, study interviewees showed concerns on the amount of time and resources required to run a SR. Cruzes and Dybå [4] performed a tertiary review of the types and methods of synthesis and concluded that synthesis of empirical research is at the heart of systematic reviews, and future attention must be directed toward synthesis methods that increase our ability to find ways of comparing and combining what is seemingly incomparable and hard to combine.

One of the challenges in conducting Systematic Reviews of Literature is to maintain a balance between methodological rigor and the required effort. Felizardo [7], Malheiros [16] and Silva Rocha [24] have applied the techniques of text mining and machine learning in order to reduce the effort and time required for construction of Systematic Review of Literature (SRL), acting, respectively, in steps of selecting papers and automatic identification of contextual information.

Thus, we propose in this paper a way to perform the automation of an important step in performing synthesis in Systematic Reviews, the results extraction activity. A previous study [27] analyzed the performance of some existing algorithms and methods in sentence classification and demonstrated the feasibility of deploying a technique to locate result sentences in unstructured Software Engineering papers automatically. The new method proposed in this paper, called Textum, has as main objective to automate part of the process that deals with the location of the sentences that represent papers results. This way, it would be possible to analyze specific parts of the paper according to rules executed by the proposed algorithm instead of manually analyzing the full paper.

This paper is organized as follows: in Section 2, information regarding related works is provided to contextualize this work; Section 3 presents details about the main methods for classifying automatically sentences in scientific papers; in Section 4 is featured details about proposed method; Section 5 shows the feasibility study of the proposed approach and defines its application viability, as well as, presents a comparison with others methods; finally, Section 6 concludes this work.

2 Background and Related Work

The effort required to conduct a Systematic Review is one of the obstacles to a greater use of this type of study. Zhang and Ali Babar [29] performed a survey asking Software Engineering researchers that had never executed SRLs before why they had never performed this kind of study. The results showed that 50% of stakeholders have taken this attitude because they did not know about SRLs at the time of the research and at time of writing the papers. In addition, 37% of them had never used the technique because of the amount time required to perform a SRL.

There are no studies to define a formula to estimate the average time taken to perform a systematic review in Software Engineering area. In medicine, Allen and Olkin [2] presented a formula (1) to determine the amount of hours spent as a function of the number of returned references (x) produced based on empirical observations made in SRLs.

$$\text{Hours} = 721 + 0.243x - 0.0000123x2 \quad (1)$$

Some authors have published, especially in recent years, papers about tools to automate parts of the Systematic Review as a way to reduce time and cost required for its realization. Felizardo et al [7] created a tool to support the primary study selection activity using visual text mining (VTM) techniques and discovered that this approach was useful in accelerating the selection task. Furthermore, the results showed that the method helped to increase the inclusion of relevant papers and the exclusion of irrelevant papers. Malheiros et al. [16] proposed an automated tool, called pexExplorer, to help researchers in the initial selection of items to be used in the SRL. This work showed that the use of visualization allowed for more information to be processed at once. This work still showed that during the selection of studies the use of this technique is valuable for data cleaning. Silva Rocha [24] developed a tool for automatic information extraction from the scientific papers context – tool called ContextExtractor. The results of the context extraction achieved by this tool were similar to the results of manual context extraction achieved by Junior Software Engineers.

These tools have been built based on text mining techniques and aim to reduce the time required for construct reviews by automating part of the process. The prototypes proposed by Felizardo and Malheiros focus on the information quality assessment and selection of the papers, in its turn, Silva Rocha´s tool focuses on the extraction of context information from studies performed and described in the paper.

However, we did not found studies to automatize the results identification activity, one of the main steps of the systematic review process. The reason is probably due to the complexity of this activity. Cruzes el al. [5] performed an experiment with graduate students at the University of Maryland enrolled in a class of Experimental Software Engineering. In this study, they assigned a set of papers for students to identify in the text the sentences that represented the results of the studies analyzed. The sentences would be identified and then compared with the oracle developed by experienced researchers in order to identify the percentage of accuracy in the results of manual selection of items and time taken for the analysis of texts. The experiment showed that the number of results found by the participants was below the expected,

since they located, on average, only 53% of existing results. It was also observed in the experiment that the students spent between 1.5 and 3 hours to read the papers, with each participant consumed, on average, 8.1 minutes to read each page. The analysis of the experiment results confirms the complexity of performing this activity, both by the low success rate and the high amount of time required for analysis.

Our study proposes a method for automatic identification of the results on Software Engineering scientific papers, reducing the time spent to perform the activity and improving rates of success in identifying of study results.

3 Sentence Classification

The problem of making the machine to understand and differentiate between different categories of sentences is usually treated in Computer Science as a task of automatic classification of text. Hachey and Grover [8] use various techniques based on machine learning to set the rhetorical status of sentences from a corpus of the legal documents. Khoo et al. [11] proposes the classification of sentences of a corpus of helpdesk e-mails into categories related to domains, such as education, answer and questions.

More focused on the topic covered in this study, there were also works with emphasis on the analysis of sentences of scientific texts. These works were divided in two categories: those that classify just the sentences found in the "abstracts" of papers [17][22][25]; and those that work on the full text [26][10][1]. Just the full text studies will be summarized below because this is the scope of this study. We can find the full discussion about these studies in [27].

The first study analyzed was developed by Teufel [26] and proposes to define a set of attributes that characterize the types of sentences. With this attributes would be possible to classify the sentences in different categories. Teufel conducted an experiment where a set of sentences extracted from a series of computational linguistics papers were classified into seven different categories according to their purpose.

The second study was written by Ibekwe-SanJuan [10]. Its main purpose was to help users on the identification of the key scientific information in the text, from elements associated with specific sentences. In her work, Ibekwe-SanJuan observed in previous studies that scientific writing is not a neutral act, it is indeed a social act because the authors need to convince the community of the validity of their research. Hence they make use of rhetorical cues and a few recurrent patterns. The author argues that, in theory, this behavior would allow the automatic identification of sentences bearing these patterns through the use of templates or regular expressions.

The last work analyzed was developed by Agarwal and Yu [1] and focused on the classification of full-text biomedical journal papers. Four different methods for sentence classification were tested, two rule-based and two machine-learning based. The first rule-based method had just one rule, the category was assigned to a sentence based on which section the sentence occurred. In the second rule-based method, the author identified 603 rules to classify the sentences into four different IMRaD (Introduction, Method, Results and Discussion) categories. The machine learning methods

were also split on two: supervised machine- learning system trained on non-annotated corpus and trained on manually annotated full-text sentences.

4 An Approach For Locating Results (Textum)

The main objective of this paper is to define a method to automate the task of locating result sentences in unstructured Software Engineering papers. Papers that do not follow the model IMRaD, i.e., their texts are not organized using the standard sections - **I**ntroduction, **M**ethods, **R**esults and **D**iscussion, will be treated in this work as unstructured papers. The expected result from the application of this method is to reduce the human effort required for performing this activity, thus reducing time spent on execution.

The proposed algorithm was built based in text mining techniques associated with classification strategies. The method was divided in four main activities, going from text importing to the identification and selection the sentences that represent the results of the scientific papers in analysis (Fig. 1).

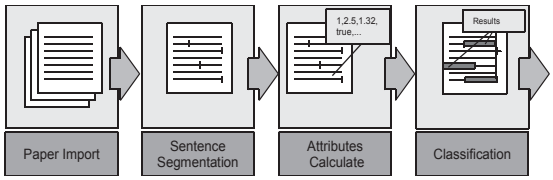


Fig. 1. - Textum Method Schema

4.1 Paper Import and Sentence Segmentation

The first step is to convert the papers into plain text format documents. Tags and undesirable characters are removed in this stage. The next step is to segment the sentences, pieces of text with full meaning. This activity is performed with a pattern recognition algorithm based on the use of punctuation, capitalization, names, acronyms and grammatical particle position in the text. In general, Pattern recognition algorithms try to identify the "most likely" answer for possible inputs based on statistical parameters and characteristic which differ from the pattern matching algorithms and look for exact matches.

4.2 Attributes Calculation

There are two main activities for defining the new Sentence Classification method: the definition of sentence attributes and the classification strategy. Text classification strategies use a number of predefined attributes to automatically infer the category and classify the pieces of text.

To define the final attribute set, we used elements defined in [26][10][15][14][20][19][23]. To select the best attributes from the whole set, we used

a genetic algorithm based technique. Classification tests were run from sets of attributes mounted from the initial group. The sets of attributes with the best classification results were selected and mixed to form new subgroups, used as the basis for new classification process, and so on, until define the best performing attributes to compose the final set.

Seventeen attributes were selected to compose the final set for each one of the sentences:

- Keyword Frequency – this attribute follows Luhn [15] model, composed by the sum of the sentence word frequencies multiplied by the distance between words.
- Cue Method – a weight was assigned to each word in the list based on the presence of this word in one of the four previously created word lists: Bonus A - weight 3; Bonus B– weight 2; Bonus C - weight 1 and Stigma – weight -1. [26]
- Paragraph Sector – this attribute was proposed by Teufel and Moens [26] and considers the sentence position in the paragraph to which it belongs. The total number of sentences in the paragraph is divided by four to create equal number of paragraph sections, in which the sentences will be allocated.
- TF-IDF - is the product of term frequency (TF) by the inverse document frequency (IDF). The calculation method is the same showed in Salton [23].
- TF-ISF – The Inverse Sentence Frequency is defined based in the word frequency in the sentence - "F(w)", in the total number of words in the sentence - "n", and in the number of sentences in which the word appears– "S(w)". The TF-ISF equation is showed in (2):

$$TF\text{-}ISF(w) = F(w) * (\log n / S(w)) \quad (2)$$

- Sentence length – number of words in the sentence.
- Gist Sentence – It uses the methods of keyword frequency and TS-ISF to select the sentence with highest score in the text, which, at least theoretically, represent the main idea. This sentence is called gist sentence of the text and become the basis to calculate the weight of the other sentences [19].
- Lexical Connectivity –the weight is set based in the number of words shared between the sentences divided by the total number of words in the sentence.
- Section Position –this attribute is loaded with the sequential position of each sentence in the section.
- Segment Position –this is a variation of the method proposed by Teufel and Moens [26]. The text is divided in ten parts and to each one is given an identifier – a letter from A to J. The sentences belonging to each part of text are defined with the letter assigned to the piece of text.
- Verbal Tense– The first verbal occurrence in the sentence. The possible values are: present in 3th singular person, present in other persons, infinitive, past participle, present participle, gerund and sentences without verb.
- Citations – IF the sentence has a citation the value is 1, else the value is set to 0.
- Header –The definition of the relevance of the header is given if the section title has at last one of these words: "result", "introduction", "conclusion", "date", "implementation" and "discussion".

- Self-indicative phrases– this approach follows the Ibekwe-SanJuan [10] method. It uses a Grammar to recognize the sentences that represent results from papers. The sentences selected are assigned with weight 1 and the others are assigned with value 0.
- Comparison – Check, through the grammar rules, if there are comparatives or superlatives in the sentence. The “er” suffix and the “more … than” structure, for example, are comparative indicators. The “est” suffix is a superlative indicator.
- Number Presence – Check if there are numbers in the sentence, if so, “Yes” is assigned, otherwise "No".
- Percentage Indicator – if there is the signal “%”, or the word “percent” (and variations) in the sentence, the value is “Yes”, else the value is “Not”.

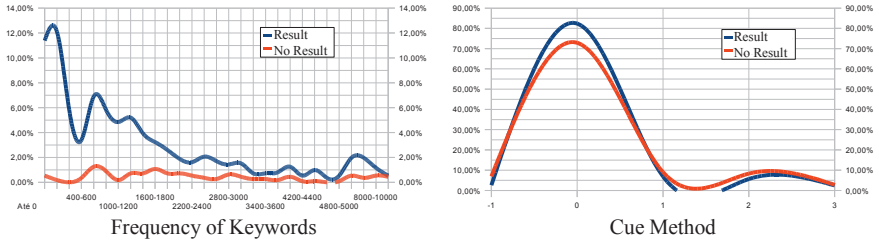
4.3 Classification

The classification task is performed in two steps. The first step uses a rule-based algorithm and the second a machine learning algorithm.

The process of choosing the classification rules set is based on the analysis of the distribution of each sentence attribute value. To perform this task, a Software Testing Papers corpus was divided into three distinct subsets. The first set was used to analyze the values of the attributes of a sentence in order to identify attributes that were causing noise in the classification task. The second and third were used in the experiment to validate the method, to perform the training and to execute tests.

To build the rule-based classification model it was necessary to discover the attributes that showed wide divergence to the values taking into account the two classification classes: “result sentence” and “no result sentence”. To understand the concept, note the analysis of values defined to the attributes "Frequency of Keywords” and "Cue Method" in the graphs showed in Fig. 2. The first one shows that it is possible to observe a predominance of a particular kind of sentence for certain attribute values, making this an attribute relevant to the classification process.

Unlike Frequency of Keywords, Cue Method shows the inverse behavior, the curve remains similar between the two categories, making this feature very ineffective in the sentence classification. Because of this, all attributes with similar distribution values to both categories were removed from the classification step.



Frequency of Keywords Cue Method

Fig. 2. – Frequency of Keywords and Cue Method Graphs

The classification policy rules were developed based on the observation of the values distributions in each analyzed attribute. Looking at the attributes values was possible to discover different patterns in the distributions between the two categories and create rules to classify the sentences. We discovered, for example, that all results have a verb in their sentence composition; hence it was created the following rule "If the sentence has no verb, must be classified in the no result category". Some other rules are described in Table 1.

Table 1. - Rule Set Example

- If the sentence has no verb, must be set to "no result";
- If the sentence, excluding "stop words", having size less than or equal to 3, must be defined as "no result"
- The sentences that have value less than or equal to 600 for the attribute "frequency of key words" must be defined as "no result";

The second classification stage is based on machine learning algorithm. Six algorithms were tested to select the heuristic to be utilized in our method: Nayve Bayes (NB); Tree J 4.8 (TJ48), Decision Table (DT), Support Vector Machines (SVM), Nearest Neighbor (IBK) and Multilayer Perceptron algorithm (MP). The tests were executed with a Software Testing Corpus. The results are showed in Table 2.

Table 2. - Sentence Classification Result

Algorithm		N. Bayes	T. J.48	TRD	SVM	MP	IBK
Result Sentences	**Precision**	26,2%	55,6%	0,0%	31,6%	32,7%	28,1%
	Recall	17,5%	7,9%	0,0%	9,5%	28,6%	39,7%
No Result Sentences	**Precision**	93,0%	92,5%	91,9%	31,6%	93,8%	94,5%
	Recall	95,7%	99,4%	100,0%	9,5%	94,8%	91,1%
Average (Results and no results)	**Precision**	87,6%	89,5%	84,5%	87,6%	88,9%	89,1%
	Recall	89,4%	92,1%	91,9%	91,0%	89,5%	86,9%

Two metrics were used in these tests: precision – number of sentences correctly classified; and recall - percentage of sentences identified taking into consideration all the sentences of the text [18]. In our study, we tried to select the "result sentences" in order to facilitate the researcher work. In this case, if the algorithm classifies "no results" sentences incorrectly is better than the algorithm does not select the "result" sentences. With a more number of sentences in the final set, the researcher will spend more time to finish his work, but without the result sentences his work could generate an incorrect result. Because of this, the recall was prioritized in the selection of machine learning classification algorithm and because this Nearest Neighbor heuristic was defined as the core of the second stage of classification process.

5 Feasibility Study With Textum

A feasibility study was developed to evaluate the precision, recall and effectiveness of Textum algorithm in automatic identification of result sentences. Two questions were defined: Can the automatic method get better precision and recall rates in the sentence classification than the rates obtained by humans in the experiment conducted by Cruzes [5]? Is it possible to reduce the text size without removing the sentences with results in the same proportion?

In this feasibility study, the whole annotation, text preprocessing, calculation of the attributes and classification rules steps were executed using a specific tool developed to this work. The machine learning classification was performed using the Weka library [9].

The first step was to select the papers to be used in the experiment. A corpus of 17 papers randomly selected in software testing area was used in our research. Nine papers were used to define the classification rules and to select the machine learning algorithm and other eight papers were selected to perform the feasibility study. A list of these papers can be obtained in [28]. The papers were imported and the sentences that represent results were marked in the tool. It was necessary to annotate the sentences only to measure the classification performance.

After that, the text mining processing was executed to index the sentences and create the matrix with them and their attributes values. The attributes described in the section 4.2. were calculated to each sentence in the set and stored in the matrix. This step finishes the pre-processing activity.

This matrix with sentences and attributes were the input to classification activity, executed in two steps and described in the section 4.3. The rule-based classification was also performed as a filter to remove some sentences that were false-positives (not results) before next classification step. The final result from this processing is a document, which will be an entry file with sentences and attributes to be loaded in the Weka, where the machine learning classification will be executed.

The study showed a precision rate up to 56.5% and recall rate up to 60%, depending on the rule-based classification scheme used. These results showed precision rates close to those described in an experiment conducted with students at the University of Maryland, in which students achieved rates close to 50% [5]. The main difference between manual and automatic tests realized was the time spent to perform these procedures. While the computational processing finished in minutes, the humans took hours to finish reading the texts.

After locate the result sentences, the tool highlight only them in the original text. Looking the highlight screen (Fig 3), the researcher could find easily the results in the full text. The main problem in this experiment was that despite the precision rates were close the numbers found in the human experiment, the recall rates were too low.

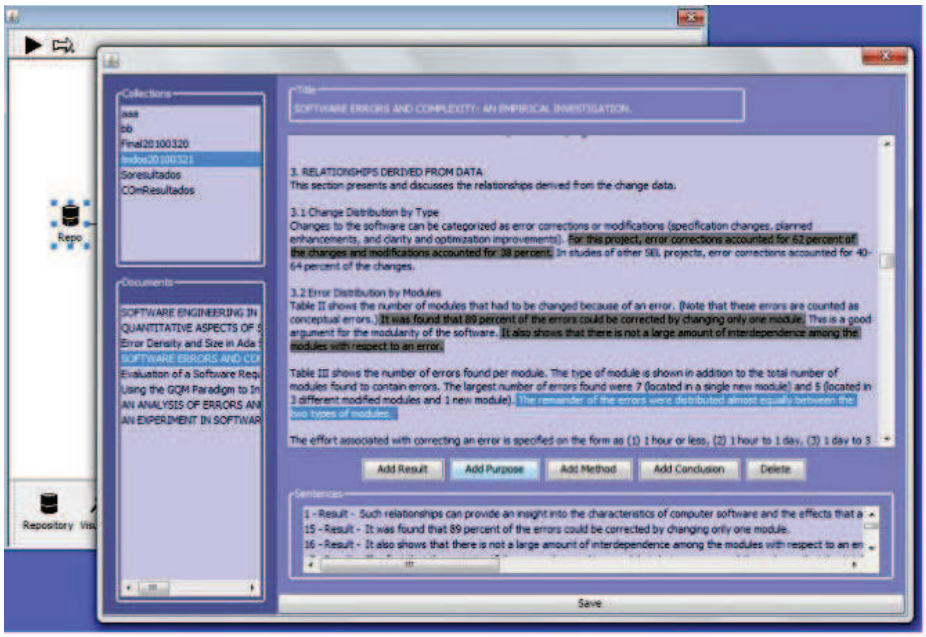


Fig. 3. - Highlight screen

However, in practice, we have noted that a researcher needs to read the whole paragraph to understand the context of a sentence highlighted, because sentences that are in the same paragraph help to explain the results. When these paragraphs, where the sentences highlighted were found, were completely analyzed, it was discovered that there were several other results that have not been automatically highlighted. The observation showed that if a tool is used to highlight the paragraphs instead of the sentences the recall rate reaches 72%, an increase of 12% compared with the original test.

Many paragraphs in a paper bring information about background or correlated studies. This information may be important to the reader to validate the study confidence and validity in a previous step of systematic review, but these aspects are not evaluated in Results Extraction step, such that reading these paragraphs imply unnecessary work for the researcher. We observed that most of the paragraphs that contain result sentences try to explain for themselves the context related with the outcomes obtained. Furthermore, although they might appear distributed in the text, these paragraphs are usually clustered and have cross-references among the elements in each cluster, which also facilitates understanding of the context.

Thereby, we changed the feasibility study to reach a new objective, instead of identifying only the result sentences, the algorithm tried to localize the paragraphs that contained at least one sentence classified as a result of the study. With this new paradigm, the Textum method achieved a precision rate of 74% in classifying paragraphs in which there are results. Even with the change, the main goal that was to reduce the effort needed to create the review has been maintained. In the end of the process, the

set formed by selected paragraphs represented approximately 20% of the initial text, a considerable reduction in the amount of information to be read by the researcher, with consequent reduction of time.

5.1 Comparison among Textum and other algorithms

The methods described in section III were applied on Health area papers. This type of papers, as already discussed, is structured, so that the sentences of common interest are grouped into standard sections. The results of these papers, for example, are concentrated in a section called "results" and the sentences that describe the methodology are in a section named “method”. The other two sections, “introduction” and “discussion”, also congregate sentences with purpose consistent with the section title.

However, as the area of interest in our work is Software Engineering (SE) and not Health, we needed to test the performance of these algorithms in this domain. Unlike health papers that use the IMRaD model, scientific papers in SE do not usually follow a fixed pattern to organize their content, so that the sentences of similar purpose are dispersed in the full text paper. In order to assess the impact of this feature in the automatic classification of sentences of papers, we tested Agarwal [1] and Ibekwe-SanJuan [10] algorithms using a set of papers from a Corpus developed by Cruzes et al. [5] in the Software Testing area. We could not test Teufel´s algorithm because we did not have access to the code used by the researcher.

To evaluate the performance of Ibekwe-SanJuan´s method to classify unstructured Software Engineering papers, we used a tool and a file with the grammar definition provided by the author. For an input set with little more than two thousand sentences belonging to Software Testing Corpus, the algorithm achieved an accuracy rate of 9.47%.

The Agarwal’s algorithm was tested from software provided by the author and the same input set used in previous method. Despite the better results, when compared to Ibekwe –SanJuan, the rates were still not good. The algorithm test resulted in 25% of accuracy in the results classification.

The main feature observed, in both trials, was the low performance of classifiers, probably due to the use of unstructured papers as input set, in contrast to the fact that originally the researchers used as input set of structured papers from health area. A detailed description of this experiment is presented in [27].

In Table 3 are listed seven sentences, extracted from unstructured papers belonging to Software Testing Corpus with their results of automatic classification experiment using three different algorithms, Agarwal´s and Ibkwe-SanJuan algorithms, described previously, and our proposal algorithm, Textum.

The results shows the difficulty to the algorithms automatically define the sentence types. The main problem is caused by the lack of a pattern to characterize the sentences. For example, in sentences as the first two in Table 3, we could think of a rule that says that the percent sign in a sentence indicates that this one is a result in a study. However, further analysis of the paper set shows that this sign is commonly found in sentences that describe the method in the paper. The sentence “In the exper-

iment, we used 30% of the original set as the training set", for example, is one example of a sentence appearing in the methods section.

Table 3. – Result Sentence Classification Test

Result Sentences	Classification Result (Algorithms)		
	Agarwal	Ibke-SanJuan	Textum
50% of the total effort required for error correction occurred in modified modules	✔		✔
18% of Errors have the source on Mistakes in control logic or computation of and Expression.			✔
Errors contained in modified modules were found to require more effort to correct than those in new modules, although the two classes contained approximately the same number of errors		✔	✔
Interfaces appear to be the major source of errors regardless of the module type.			
50% of the total effort required for error correction occurred in modified modules.	✔		✔
A major source of insight when analyzing a software development project is a record of the changes, including error corrections, made as the development progresses.			
The average effort to make a change was 5.0 man-hours, and the average to correct an error of any type was slightly higher, 5.4 man-hours.	✔		✔

6 Conclusion

In this paper we define and apply a method for automatically locate results from empirical studies written in unstructured published papers, called Textum method. The information in these papers is written in natural language, which is ambiguous even for humans. An experiment, in which some papers were provided to groups of students to identify the sentences in the papers that represented results, [5] showed that, on average, only 53% of all sentences in the existing text were correctly located.

The tool proposed in this work performed even better results than the students in this experiment. From the three approaches for automatic semantic annotation of sentences discussed in this paper, none of them outperformed the proposed method when the input set consisted of items that did not follow the IMRaD model. The precision and recall rates for these algorithms were below 30% while the proposed method exceeded the 47% level of precision in a recall of more than 60%. The precision rates obtained in the tests with Textum and in the experiment with humans [5] were similar.

Meanwhile, the recall rates in the automatic selection were lower than in the human selection. However, it was noticed that a text only composed of "result" sentences were not understood by the researches, they needed context information present in adjacent sentences to execute the systematic review process. Thus, instead of selecting only the "result" sentences in the text, we changed the paradigm and started to select the whole paragraphs in the text that contained at least one sentence classified as "result".

This change allowed an increasing the accuracy level to 74% and the recall level to 72%. Selecting only the paragraphs, the Textum method reduced the text to be analyzed by researchers to 20%, which, in theory, would probably reduce in 80% the time spent on traditional analysis of the paper. It is noteworthy that Textum method is focused on Software Engineering papers. The use of this method in other study areas, especially health, has not been evaluated; future works include the creation of an annotated corpus of papers from another area and evaluate the efficacy of this algorithm on these papers.

ACKNOWLEDGEMENTS

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq, grant 573964/2008-4.

REFERENCES

1. Agarwal, S.; Yu, H. Automatically Classifying Sentences in Full-Text Biomedical Articles into Introduction, Methods, Results and Discussion. In Proceedings of the AMIA Summit on Translational Bioinformatics, 2009.
2. Allen, I. E. and Olkin, I.. Estimating time to conduct a metaanalysis from number of citations received. Journal of the American Medical Association, 282(7): 634–635
3. Biolchini, J., Mian, P.G., Natali, A.C.C., and Travassos, G.H. (2005) Systematic Review in Software Engineering, Univ. Rio de Janeiro, TR, ES 679/05.
4. Cruzes, D. and Dybå, T., Research synthesis in software engineering: A tertiary study. Information & Software Technology 53(5): 440-455 (2011)
5. Cruzes, D., Mendonça, M. G., Basili, V.R., Shull, F., Jino, M.: Extracting Information from Experimental Software Engineering Papers. SCCC 2007: 105-114.
6. Dybå, T., Kitchenham, B.A., and Jørgensen, M. (2005) "Evidence-based Software Engineering for Practitioners," IEEE Software, 22(1): 58–65.
7. Felizardo, K. R.; Nakagawa, E. Y. ; Feitosa, D. ; Minghim, R. ; Maldonado, José Carlos . An Approach Based on Visual Text Mining to Support Categorization and Classification in the Systematic Mapping. In: 13th International Conference on Evaluation & Assessment in Software Engineering (EASE 2010), 2010.
8. Hachey, B.; Grover, C. Sequence modeling for sentence classification in a legal summarization system. Proceedings of the 2005 ACM symposium on Applied computing, 2005.
9. Hall, M.; Frank, E.; Holmes, G.; Pfahring-er, B.; Reutemann, P.; Witten, I. H. The WEKA Data Mining Software: An Update. SIGKDD Explorations, Volume 11, Issue 1, 2009.
10. Ibekwe-SanJuan, F.; Chen, C.; Pinho, R. Identifying Strategic Information from Scientific Articles through Sentence Classification. 6th International Conference on Language Resources and Evaluation Conference (LREC-08), Marrakesh, Morocco, 26 May -1st June, 2008.
11. Khoo, A.; Marom, Y.; Albrecht, D. Experiments with Sentence Classification. In Proc. of Australian Language Technology Workshop, pages 18—25, 2006.
12. Kitchenham, B.A. Procedures for Performing Systematic Reviews, Keele University, Technical Report TR/SE-0401 and NICTA Technical Report 0400011T.1, 2004.

13. Kitchenham, B.A., Dybå, T., and Jørgensen, M. Evidence-based Software Engineering, Proc. ICSE'04, Edinburgh, Scotland, 23-28 May, pp. 273–281, 2004.
14. Larocca, J.; Santos, A.D.; Kaestner, A.A., Freitas, A. A. Generating Text Summaries through the Relative Importance of Topics. In: Proceedings of the International Joint Conference IBERAMIA/SBIA, Atibaia, SP, 2000.
15. Luhn, H. P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development, 2, 157-165, 1958
16. Malheiros, V. D. ; Hohn, E.; Pinho, R ; Mendonça Neto, M. G. de ; Maldonado, J. C. . A Visual Text Mining approach for Systematic Reviews. In: Empirical Software Engineering and Measurement, 2007. ESEM 2007: 245-254, 2007.
17. McKnight, L.; Srinivasan, P. Categorization of Sentence Types in Medical Abstracts. In AMIA Symposium, 2003.
18. Olson, D.; Delen, D.. Advanced data mining techniques. Springer Verlag, 2008.
19. Pardo, T.A.S; Rino, L.H.M.; Nunes, M.G.V. NeuralSumm: Uma Abordagem Conexionista para a Sumarização Automática de Textos. In Anais do IV Encontro Nacional de Inteligência Artificial – ENIA, pp. 1-10. Campinas-SP, Brasil, 2003.
20. Pardo, T.A.S. Gistsumm: um sumarizador automático baseado na idéia principal de textos. Série de Relatórios do Núcleo Interinstitucional de Linguística Computacional, São Paulo, 2002.
21. Rosenberg, W., & Donald A. (1995). Evidence based medicine: an approach to clinical problem-solving. British Medical Journal, 310 (6987), 1122-1126.
22. Ruch, P. A.; Geissbühler, A.; Gobeill, J.; Lisacek, F.; Tbahriti, I.; Veuthey, AL.;Aronson; AR. Using Discourse Analysis to Improve Text Categorization in MEDLINE. Medinfo, 2007.
23. Saltob, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. Information Processing & Management 24 (5): 513–523, 1988
24. Silva Rocha, M. C. Contextextractor: uma ferramenta de apoio para a extração de informações de contexto de artigos de engenharia de software experimental. Master Thesis, Universidade Salvador, 2009.
25. Tbahriti, I.; Chichester, C.; Lisacek, F.; e Ruch, P. Using argumentation to retrieve articles with similar citations: An inquiry into improving related articles search in the MEDLINE digital library. International Journal of Medical Informatics, 75(6):488–495, 2006.
26. Teufel, S.; Moens, M. Discourse-level argumentation in scientific articles: human and automatic annotation. In:Towards Standards and Tools for Discourse Tagging. ACL 1999 Workshop, 1999.
27. Torres, J. A. S.; Cruzes, D. S.; Salvador, L. N., Automatic Results Identification in Software Engineering Papers. Is it possible? Proceedings of the International Conference on Computational Science and Its Applications (ICCSA), 2012.
28. Torres, J. A. S., Automatic summarization of software engineering papers to support the systematic review process. Master Thesis, Salvador University, Graduate Program in Computer Science, Salvador,BA/Brazil,2011.
29. Zhang, H. and Ali Babar, M. An Empirical Investigation of Systematic Reviews in Software Engineering. Empirical Software Engineering and Measurement (ESEM/11): 87-96