Using Text Mining to Locate and Classify Research Papers

EDGAR ALAN CALVILLO MORENO
Instituto Tecnologico de Aguascalientes
Av. Adolfo Lopez Mateos 1801 Ote. CP 20256, Aguascalientes, Mexico
MEXICO
alancalvillo@yahoo.com

RICARDO MENDOZA-GONZÁLEZ
1Postgraduate Studies
Universidad Politecnica de Aguascalientes
Calle Paseo San Gerardo 207, Fracc. San Gerardo, CP 20342, Aguascalientes
Mexico

JAIME MUÑOZ-ARTEAGA
Universidad Autonoma de Aguascalientes
Av. Universidad 940, CD Universitaria, CP 20131, Aguascalientes, Mexico
MEXICO
jmaauaa@gmail.com

JULIO CESAR MARTINEZ-ROMO
Instituto Tecnologico de Aguascalientes
Av. Adolfo Lopez Mateos 1801 Ote. CP 20256, Aguascalientes, Mexico
MEXICO
jucemaro@yahoo.com

SANDRA JIMENEZ-GONZÁLEZ
Universidad Politécnica de Aguascalientes
Calle Paseo San Gerardo, Fracc. San Gerardo, CP 20342, Aguascalientes, Mexico
MEXICO
mc140003@alumnos.upa.edu.mx

Abstract: - The use of search engines to get research papers has grown exponentially. Natural language give us a lot of information without be classified there's a lot of key elements ignored on documents because of complexity from natural language. There are multiple search engines with the capacity of search and classify information but almost all of them takes all the information stored on a previous format with key fields like title, description, abstract, authors, theme and according to this information we got results from that engines, we could be able to use that information lost to classify and search papers and according to this we are going to be able to use that information to give better results on any search. In this work we provide a semi-automatic engine for classification and search based on Naive Bayes and text mining, this techniques will provide us a better approach when we took research papers and classify to use on any search engine. We use an illustrative example as a proof of concept following a methodological approach to specify the workflow, together with a preliminary study comparing our proposal against a wrapper, in this case we develop a wrapper to compare against our Naive Bayes algorithm.

Key-Words: - Naive Bayes, text mining, Crisp-DM, database, classifier, algorithm.
1 Introduction

The use of research papers repositories that allows researchers, academics, and students around the world to get information in many areas of Science grows up daily. Services such as Springer, ACM-Digital Library, Science Direct, Scopus, IEEE among others available online, provide access-alternatives to content from journals, conferences, book chapters, and other scientific forums, by Nevertheless, many of these services allows searching from general information such as title; description; abstract; authors; institution; area; and theme. This situation neglects very useful information that could enhance papers searching[1].

The researchers usually go into a digital library and write their own key words selected to locate works but we don't have a perfect feedback from our search because we could have important information on a title from any image or any other place from the document, this is because when we upload any document into any digital library we just fill a few fields and put important information related with that paper but we are not sure if we put all important information or keywords and this kind of behavior could cause that we don't have a good information related with our paper.

There's a certain number of techniques used on data mining to identify and classify information, but on data mining we used to have an specific structure of information with predefined fields and this let us apply these techniques because we know what kind of information we are expecting but this doesn't occur on text-mining because in this case we don't have an specific structure of information we are going to work with a non-structured document. We could define text mining[2] as a process to extract information looking for statistical patterns to be used to classify or locate information.

This is a usual issue when a repository takes information from a paper it only includes the indexed information on previous upload but this could be solved using text-mining techniques[2], to go through any paper and get important information from any paper uploaded into the digital library.

We start this work looking for any methodology like figure[3] that could help us to generate a proper architecture to our work, in this case we select Crisp-DM[4] to adapt into a text-mining architecture.

The first phase consists in give a review of the problem that we are going to solve and have clear our objective to be solved; in this case we are looking for an algorithm to let us classify and search information extracted from research papers.

The second phase of this methodology consists in take a deep analysis of the data to be used in the methodology; this analysis must include a previous limit of the information to be used. If we are going to work with a semi-automatic classification/search we must select a significant number of clusters to be classified because according to the kind of information used on this phase is the level of significant algorithm that we could use on deployment phase.

The third phase consists in have an analysis of how we are going to clean our papers used to generate our knowledge database. In this particular phase we must take a review of which information is not useful to our algorithm basically this phase will leave us a clean paper to be used in our extraction of significant information to our database.

The modeling phase consists in select a proper algorithm to be used in our classification/search at this point we already have a proper analysis of the problem to be solved and a proper information with their own cleaning. We are going to be able to execute a proper algorithm to extract the significant information from our papers.
We must execute a proper evaluation from our data received on modeling phase, this must be validated using a statistic tool and have a proper analysis if our information obtained from our papers is significant. In case of a not significant result we must start the process again from problem analysis.

If the information is significant we are going to be able to go into a deployment and use our algorithm to classify and search papers.

This work propose the implementation of this architecture with our first steps focus on clean text removing unnecessary information like images, short words, code if it is presented and common repeated words after we have our document cleaned we ran our algorithm in two phases. The first phase consists in take all our papers related with three specific areas to generate a database used to classify / locate papers once we have our database with phrases and their own values obtained with Naive Bayes, we could go through our application and generate multiple searches using big phrases to be analyzed with the same algorithm used to generate our database.

The main reason of choose Naive Bayes was related with it takes as an assumption that all the elements are independent between them it makes useful if we are working with natural language, Naive Bayes[5] takes an average of all the events and they’re stored separated from other events.

2 Problem Outline
The use of search engines to get research papers has grown exponentially; this is because of all the multiple users sending information to different research repositories with a lot of information creating a snowflake effect including important information[6], but almost all the repositories only got information from the form created at the beginning when the user goes into the site and upload all their own information but if the user don't put some specific keywords on that process, these keywords are not going to be included on any search no matter which search engine use , it will take only the information filled at the first form[7], this is a common scenario on almost all the repositories with papers and bearing in mind these aspects we realized the following objectives for our proposals:

- Create a “cleaner algorithm” to prepare all the papers to be used in our database of terms[8]
- Generate a classifier algorithm based on Naive Bayes oriented to search and classify papers using as entry the terms database[5].
- Generate wrapper to compare against Naive Bayes.
- Establish a database using Naive Bayes to categories on first bullet point.

3 Contribution
This set of bullet points will let us start working and in the first place have an algorithm to get information from natural language, according to this we are going to generate a database to start our classify algorithm, once we have our classify algorithm it will use the same logic instead of read a full paper with a big number of phrases it will be run over a less number of phrases introduced by final user[9].

3.1 Database Validation
The first step to validate if that database is significant, we are going to need to follow the next steps to validate:

1. The database must be filled with all the words obtained from papers of three categories.
2. Take paper from list of papers used to create database and remove all these words found on database.
3. Run algorithm to classify and get category.
4. Go into step 2 and take a different paper until finish with all the papers.

We are going to get a first set of numbers to validate how many papers were classified on the right category, according to this we are going to know if our database is significant.

3.2 Database Use
The second test is once we know our database is significant and also we have a good algorithm to classify and locate is go with some tests out of the box, this case consist in take a number of papers different from our papers used to generate our database and run our algorithm to classify and get
the numbers with this process we are going to be able to know how useful is our database and algorithm with some papers out of the box.

After review current literature on Crisp-DM, we could extract an approach focused on text-mining structured into the following steps[3]:

\[
\text{Begin} \hfill \\
\text{paper} = \text{getPaper} \hfill \\
\text{paperCleaned} = \text{cleanAlgorithm} (\text{paper}) \hfill \\
\text{dataProcessed} = \text{textMiningAlgorithm} (\text{paperCleaned}) \hfill \\
\text{evaluateInformationProcessed} = \text{applyTechniqueValidity}(\text{dataProcessed}) \hfill \\
\text{if} (\text{evaluateInformationProcessed} == \text{`Significant'}) \text{then} \hfill \\
\quad \text{use textMiningAlgorithm} \hfill \\
\text{else} \hfill \\
\quad \text{change textMiningAlgorithm and cleanAlgorithm} \hfill \\
\text{end if} \hfill \\
\text{End} \hfill \\
\]

Since papers have non-structured information and natural language, we adapt the above steps in order to address them to text mining providing a specific structure as follows[10]:

- Remove words with less than 3 characters.
- Remove generic words.
- Remove adverbs and adjectives.
- Remove unnecessary verbs.
- Remove images.

After this, we would obtain a clean set of meaningful terms enable classification by finding patterns which will be grouped considering repeated-phrases throughout the analyzed paper:

\[
\text{Begin} \hfill \\
\text{Set Directory in Papers FolderCategory} \hfill \\
\text{Get numberOfPapers} \hfill \\
\text{Get arrayPapers} \hfill \\
\text{Do i=0 to numberOfPapers} \hfill \\
\quad \text{paperCleaned[FolderCategory][i]} = \text{cleanAlgorithm} (\text{arrayPapers[i]}) ; \hfill \\
\quad \text{paperUniqueWords[FolderCategory][i]} = \text{uniqueWords(} \text{paperCleaned[FolderCategory][i]} \text{)} ; \hfill \\
\quad \text{frequencyWords[FolderCategory][i]} = \text{frequencyAlgorithm} (\text{paperCleaned[FolderCategory][i]} , \text{paperUniqueWords[FolderCategory][i]}) \hfill \\
\text{end do} \hfill \\
\text{Do i=0 to count(frequencyWords)} \hfill \\
\quad \text{averagePerCategoryWord[FolderCategory][i]} = \text{getAverage(frequencyWords[FolderCategory][i])} ; \hfill \\
\text{end do} \hfill \\
\text{insertDataIntoAveragePerPhrases(averagePerCategoryWord[FolderCategory], FolderCategory)} ; \hfill \\
\text{End} \hfill \\
\]

3.3 Naïve Bayes

The first step is get an average of every word on their different categories (Development, Database and operating systems), it takes the values generated per category and these values are going to be stored to use on the next step.

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Database</th>
<th>Development</th>
<th>OS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Program</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1. Average per phrases.

The table 2 represents every word used to fill our database with their own probability that occur on every category, this information represents our Naïve Bayes table.

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Database</th>
<th>Development</th>
<th>OS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
</tr>
<tr>
<td>Program</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. Naïve Bayes Table.

The table 2 represents every word used to fill our database with their own probability that occur on every category, this information represents our Naïve Bayes table.

The next step could be split in two parts, if we need to classify our paper we just need to read our paper and according to phrases found we just select that phrases from our database and indicate at our algorithm that we found that word to be considered and we pull the values per category, at the end we
are going to have a list of phrases found in our document and we just need to get a sum per category and according to our final result we are going to be able to determinate of which category belongs.

Classifier reveals the category, which the paper classified, belongs, “Database” in this case.

![Diagram of flow on classification algorithm]

The Figure 2[3] shows how the paper was treated by algorithm to be classified into our database knowledge.

In this case we run in the first step our reader looking for patterns using our “Pattern Locator”, in this place we pull similar terms located into our Knowledge DB to extract a possible category and according this information we insert the paper into our repository of papers.

This particular example was using a paper to classify information, but if we have a search engine with a field to give the user an open search to locate the paper that he needs, we took that words introduced by user and according to this information we ran the same algorithm used to classify papers but with the difference instead of have a paper to classify and locate, we are going to have a long string from user to classify and instead of locate we are going to show results according to the category that belongs.

If we have a knowledge database created it would be necessary to have a deep review on statistic terms we could have two options we could have a deep review testing with a big number of papers to watch the behavior or we could use a leave one out technique to get our deep review.

We are going to use leave one out because we are going to be able of test the data used to create the database, in this case leave one out gives to us the opportunity of remove every word used for every paper to be removed from database knowledge and then with the rest of data contained on knowledge database we try to classify our paper removed and this test at the end will throws a result if every paper is classified in the right way we are going to have a solid information in our knowledge database.

This is the best option to have a deep review it is fast and it would be done by an automated script and we are going to get our results in a short time instead of the time that could be costs if we go through the other option and locate a big number of papers to have a result related with the significant of our database knowledge.

### 4 Case of Study

We start working on construction of our knowledge database, the first step was determinate the number of papers to be analyzed. We select three categories database, operating systems and development, this particular selection were focus on similar categories to achieve a better algorithm to clean and classify/search. We select 30 papers from Springer, IEEE and Elsevier using the search engine to get 10 random papers per library per category. The papers selected are between 10-15 pages with double column.

In order to analyze the effectiveness of this proposal, we use two perspectives:

1) We were focus on review if our Naive Bayes is significant, using the concept “Leave-one-out”. The “Leave-one-out” consists on validate our resources used to build our database knowledge, in this case we took in the first a paper used to build the knowledge database and take the unique words found on that paper, once we have our unique words we remove these words from our knowledge database and we try to classify that paper using the rest of words in our knowledge database and we try to classify that paper using the rest of words in our knowledge database, this would let us know if the paper is classified in the right category. This was made following the next steps:

Begin
Set Directory in Papers Folder Category
Get numberOfPapers
Get arrayPapers
arrayDatabaseKnowledge=getNaiveBayes();
Do i=0 to numberOfPapers
paperCleaned[Foldere Category][i]=cleanAlg
algorithm(arrayPapers[i]);
paperUniqueWords[FolderCategory][i]=uniqueWords(paperCleaned[FolderCategory][i]);
databaseKnowledgeRemovedtmp=removeWordsfromDBKnowledge(paperUniqueWords[FolderCategory][i],arrayDatabaseKnowledge);
classify=naiveBayesAlgorithm(databaseKnowledgeRemovedtmp,paperUniqueWords[FolderCategory][i]);
insertClassifiedElement(classify,FolderCategory);
end do
End

The results from “Leave-one-out” will let us know revealed that algorithm could properly classify and locate all papers for database, development, and operating systems. If we are using a proper algorithm to classify and locate research papers, after this procedure we are going to be able to know if our algorithm is able to locate every paper used to build the database.

![Fig.3. These are Results after Leave-one-out.](image)

The leave-one-out figure 3 shows how our algorithm to clean paper and classify/search papers works properly with a good classification of every paper used to build our knowledge database. In this case we had a 100 percent of papers in a correct classification it means our knowledge database is significant.

2) The next test to our Naive Bayes database is take a 10 papers per category with the same steps used before, random articles per category and select them from IEEE, Elsevier and Springer.

We are going to classify these papers using our algorithm to clean papers and our Naive Bayes database. We are going to need to compare our Naive Bayes against another similar algorithm, in this case text mining use a Wrapper to classify/search.

A wrapper[10] is an algorithm used in text mining with a metaheuristic focus on an specific task like classify, search or apply another techniques on data mining or text mining.

The results obtained using this technique must be validated against a Wrapper in this case we are going to use ant colony to build our Wrapper.

The objective of this test is to get our performance in our Naive Bayes database with our algorithm and compare it against a wrapper. We use a value of 0.01 into our pheromone. The database is the same for wrapper and our Naive Bayes algorithm.

![Fig.3. The Bayes result.](image)

The figure 4 shows a good work classifying papers out of scope on development and database category, but it has a bad result on OS, this test as we said was using different papers that we use to create our database.

![Fig.4. The Wrapper result.](image)

The wrapper result shows a good result on OS, a 20 percent of error on development and 80 percent of error on database. We test with different values on pheromone but we got worst results this is the best approach using ant colony in our wrapper.
5 Conclusions

In this paper we provide a semi-automatical strategy based on data mining and Naive Bayes to classify and locate research papers in the web. This proposal is based on the CrispDM methodology that is oriented to ensure good results in software development.

We analyzed our proposal through the implementation of the statistical cross evaluation technique a “Leave-one-out” obtaining good results. Although this preliminary study reveals several strengths of our proposal, there are several venues for future work. By example, to perform another study using bigger databases. Other aspect to cover is get close into a full automatic classify and without any need of has a specific set of categories and grow our database according the papers used to build the Naive Bayes database.

We just found one similar work[8] but in this work they’re didn’t compare the algorithms generated against another similar tool or work, in this paper we compare our Naive Bayes algorithm against a wrapper.

Additionally, this paper was intended to provide a basic starting point to help developers in building databases, particularly for digital libraries repositories, using Naive Bayes algorithm and text mining. The strategy is focus into get important information from research papers which could be used into a search engine providing more details for classification and location instead of just getting information from a form in a digital repository.

References:

[10] Nicholas Kushmerick, “Wrapper introduction for information extraction.”.