

# Attribute Analysis in Biomedical Text Classification

**Francisco Carrero García<sup>1</sup>**  
francisco.carrero@uem.es  
**José María Gómez Hidalgo<sup>1</sup>**  
jmgomez@uem.es

**Enrique Puertas<sup>1</sup>**  
enrique.puertas@uem.es  
**Manuel Maña<sup>2</sup>**  
manuel.mana@diesia.uhu.es

**Jacinto Mata<sup>2</sup>**  
mata@uhu.es

<sup>1</sup> Universidad Europea de Madrid, Tajo S/N, 28670 Villaviciosa de Odón, Madrid, Spain

<sup>2</sup> Universidad de Huelva, Dr. Cantero Cuadrado 6, 21071 Huelva, Spain

## Abstract

Text Classification tasks are becoming increasingly popular in the field of Information Access. Being approached as Machine Learning problems, the definition of suitable attributes for each task is approached in an ad-hoc way. We believe that a more principled framework is required, and we present initial insights on attribute engineering for Text Classification, along with a software library that allows experiment definition and fast prototyping of classification systems. The library is currently being used and evaluated in Information Access projects in the biomedical domain.

**Keywords:** text classification, machine learning, attribute engineering.

## 1 Introduction

Text Classification is a subtask of Information Retrieval that involves the automatic labeling of textual elements (such as documents, words or groups of words) with categories. Within the latest years, Machine Learning (ML) techniques have become the main approach to build classification systems. One reason for this is related to the fact that this approach has proved to be very effective for some classification tasks and knowledge domains. However, it should not be forgotten that there is an increasing number of resources available to researchers, and this fact has contributed to reduce the development costs of classification systems with a ML approach.

An example can be found Text Categorization, a subtask of Text Classification in which categories are predefined. Sebastiani [4] states that the advantages of using ML with Text Categorization “are an accuracy comparable to that achieved by human experts”, and “considerable savings in terms of expert manpower, since no intervention from either knowledge engineers or domain experts is needed for the construction of the classifier or for its porting to a different set of categories.”

Domain experts now have become responsible for creating the resources that will be used as data sets. For instance: when working with web pages, we can use web directories such as Yahoo!, which have millions of web pages classified by human experts; Ohsumed, a set of references from the on-line medical information database MEDLINE; and Reuters Corpus Volume 1 is currently the most widely used collection for news categorization research.

The selection of attributes that will be used to represent the instances is currently one of the most critical issues in Text Categorization, especially because other stages in the Text Categorization cycle have available libraries and tools that simplify the process. For instance, WEKA [6] is a suite of ML software developed at the University of Waikato with Java technology, which implements several algorithms from various learning paradigms.

Our work is focused on the analysis of common elements in the multiple Text Classification tasks. We have developed a framework and software library that allow together the analysis, modeling and fast prototyping of classification systems, supporting both the experimentation phase and the development of functional system prototypes.

This library is being used in two R&D projects, Isis and Sinamed [5], whose objective is to enhance Information

Access in the medical domain through the improvement and utilization of Text Classification tasks, like Text Categorization, Automated Text Summarization, and Biological Entity Recognition. While Sinamed is focused in basic research, Isis involves dealing with actual patient medical records. We are currently using our library for tasks addressed in these projects.

## 2 Organization of Text Classification Tasks

A possible organization of Text Classification tasks can be done on the basis of the granularity of the text elements. Therefore we can either consider words, phrases or documents as atomic elements. Another way to categorize Text Classification is to determine whether the learning process is supervised or unsupervised. These two organizations are often combined to characterize a Text Classification task [3]. Examples of this are the following ones:

- Text Categorization works with documents and uses supervised learning.
- Document clustering considers documents as basic elements, but the learning process is unsupervised.
- Name Entity Recognition is a task that assigns predefined labels (supervised learning) to sequences of words that represent a certain kind of entity.
- Key Phrase Extraction also works with words, but learning algorithms are unsupervised.
- Text Summarization addresses both the problem of selecting the most important portions of text and the problem of generating coherent summaries, using supervised learning.

Focusing on an actual task addressed by our framework, we can distinguish three different phases in the life cycle of a Text Categorization system: document indexing, classifier learning and classifier evaluation [1].

- Document indexing involves mapping a document into a compact representation of its content that can be directly interpreted both by a classifier-building algorithm and by a built classifier. In this stage the decisions on attribute representation and attribute selection become critical.
- Classifier learning. A text classifier is automatically generated by a general inductive process. This process infers the characteristics that any document should have to be classified under each category by observing the characteristics of a set of pre-classified documents
- Classifier evaluation. Different measures to evaluate a classifier are: effectiveness (success rate), training efficiency (required time in average to train a classifier) and classification efficiency (required time in average to classify a document).

There is a number of software libraries that provide support to the latest phases. However, document indexing is most often approached in an ad-hoc fashion. Therefore, we believe that a framework is required to better understand the value of potential representation elements (attributes), not only in text Categorization, but in general, in all the Text Classification tasks.

## 3 A Taxonomy of Attribute Types for Text Classification

A first taxonomy of attributes, attending to its internal representation, can be found in [2]:

- Quantitative: e.g.
  - continuous values (e.g., weight);
  - discrete values (e.g., the number of computers);
  - interval values (e.g., the duration of an event).
- Qualitative:
  - nominal or unordered (e.g., color);
  - ordinal (e.g. military rank, qualitative evaluations of temperature (“cool” or “hot”) etc.).

The kind of attributes to use is an important aspect to be considered when experimenting with different ML algorithms. Nevertheless, when computing an attribute given a training instance, other criteria should be taken into account, related to the set of examples that must be processed to set a value for an attribute of a single example. We propose the following types:

- Intrinsic. When computing an attribute for a given example, only information from that example is used. E.g.: the length of a text in Text Categorization.
- Contextual extrinsic. The information is obtained from the processed example, but also from other examples that have a strong relation with it. E.g.: occurrence of a word in a text cited by the current text.
- Global extrinsic. The information comes from all the examples in the set. E.g.: occurrence of a word in the rest of the texts included in the set.

We can give some representative examples for another Text Classification task: Name Entity Recognition (NER). The main paradigm used to solve this problem involves reducing it to a word tagging problem and face it using ML

methods. The idea consists on identifying lexical, morphological, syntactical and orthographic attributes to characterize every word in the text, and applying ML algorithms to categorize the words as part of a named entity or not. Within this task, a common intrinsic attribute could analyze the current word to check if the first character is a capital letter. An example of contextual extrinsic features could be the part of speech for the current word. Finally, a global extrinsic attribute could pre-process all the words in the set to get a list of words likely to be part of an entity, and then check if the current word belongs to this list.

This organization of attributes has strongly motivated the organization of our software library, detailed in the next section.

## 4 Library Organization

As we have mentioned before, we have developed a framework and software library that allow the analysis, modeling and fast prototyping of classification systems. It runs part of the document indexing process, specifically the mapping of a document into a compact representation, but also helps to build fast functional prototypes to make experiments with different tasks and attribute sets.

With these considerations in mind, identified the following requirements for the library:

- Simple and flexible attribute definition.
- Simple and flexible definition of input and output formats.
- Easiness to build prototypes for diverse classification tasks.

The library is organized in the following packages:

- `jtLib.jtFormatter`. This is the core package of the library, and contains the classes that support all the mapping process. The main class is `JTFormatter`, which implement an abstract mapper by running the following tasks in the given order:
  1. Pre-processing of attributes.
  2. Processing of intrinsic attributes.
  3. Processing of contextual extrinsic attributes.
  4. Processing of global extrinsic attributes.
  5. Generation of data set in the object representation.
- `jtLib.attrs`. It contains some common pre-defined attributes to be used in diverse Text Classification tasks. We compute these attributes according to the following characteristics:
  1. The granularity of the text element to be classified.
  2. The internal representation.
  3. Its intrinsic or extrinsic nature.
- `jtlib.tasks`. It holds several sub-packages that implement different classifiers, like a text categorizer and an entity recognizer

To provide a flexible attribute selection we have developed an attribute configuration system based in XML. It means that a formatter can use any attribute defined in the `jtLib.attribute` package just by adding it to an XML configuration file. Thus, several parameters can be defined in the file, such as the name of the attribute, the class that will compute it, the size of the attribute window to be considered, the position of the current attribute within the window, the intrinsic or extrinsic nature (needed to decide if an attribute needs a pre-processing phase), etc. This helps to build different attributes vectors easily in a systematic way, therefore allowing to run multiple (and long) experiments to find the best vector with little human intervention.

The following code is a fragment example of an attribute configuration file. We define a contextual extrinsic attribute that computes if the words in a window of two words before to two words after begin with a capital letter.

```
<list>
  <attribute>
    <name type="string">initialCapitalLetter</name>
    <class type="string">jtLib.attrs.InitCaps</class>
    <domain type="string">{ 0, 1 }</domain>
    <size type="int">5</size>
    <prev_window type="int">2</prev_window>
    <next_window type="int">2</next_window>
    <scope type="string">contextual extrinsic</scope>
  </attribute>
</list>
```

The next two sections provide an example of the work we have developed for the 2006 Biocreative challenge.

## 5 Approach and performance on the Gene Mention Task

In this task we applied a simple process to build the classifier, with the aim to get a first working version with a low effort, and then concentrate on attribute analysis. During the first part of this process (get the working version) we used our JTLib library and the WEKA package for the following stages:

1. Document indexing. We used JTLib to develop an application that processes the training data (the 15,000 sentences preceded by a identifier) to obtain a representation based on the selected attributes and configured into the input WEKA format (ARFF).
2. Dimensionality reduction. Once the former ARFF file is generated, we used WEKA to process it aiming to find the attributes with best information gain. Then, we obtained a new and definitive training file to build the classifier. Table 1 shows the selected attributes classified by the information needed to compute them. We used 28 attributes to characterize each instance. Table 2 collects the ranking of the most relevant attributes obtained from the application of information gain.

Table 1. Selected attributes and their classification

Attribute type	Attribute Name	Description
Intrinsic	hyphen punctuation initCaps lettersAndDigits number	Set of lexical and morphological features depend only on the current word itself: begins with a capital letter, contains letters and digits or only letters or numbers, is a punctuation mark, or includes a hyphen.
Global extrinsic	frequentWords frequentWordsInEntity prev1Unigrams prev2Unigrams startingWords endingWords	These attributes receive a value indicating whether the word can be usually found as a starting or ending word in a named entity, if it is a frequent word inside or outside an entity, or if it can be found just before the beginning of an entity.
Contextual extrinsic	endOfSentence punctuation $\pm n$ initCaps $\pm n$ frequentWords $\pm n$ frequentWordsInEntity $\pm n$	EndOfSentence indicates if the current word is placed at the end of a sentence, so its value is given by the context. The others are values obtained from a window of $\pm 2$ words.

Table 2. Ranking of attributes

Ranking	Attribute
1	frequentWords
2	frequentWordsInEntity
3	frequentWords - 1
4	frequentWords + 1
5	endingWords
6	frequentWordsInEntity - 1
7	prev1Unigrams
8	frequentWordsInEntity + 1
9	lettersAndDigits
10	startingWords
11	frequentWords - 2
12	frequentWords + 2
13	frequentWordsInEntity - 2
14	prev2Unigrams
15	hyphen

Table 3. Effectiveness on the test data

	C4.5 unpruned	C4.5 pruned
<b>Precision</b>	50.09	53.37
<b>Recall</b>	46.12	42.46
<b>F-measure</b>	48.02	47.29

3. Classifier learning. Using WEKA, we generated a set of models with different Machine Learning algorithms. From these classifiers, C4.5 decision tree achieved the best results. The C4.5 algorithm allows to done a pruned tree in a reduced time but increasing the error rate. We built two classifiers, both pruned and unpruned.
4. Evaluation of text classifiers. Table 3 shows the effectiveness of both classifiers evaluate using the test data set. The C4.5 unpruned achieves a scarcely improvement of the F-measure respect to C4.5 pruned. However, the time needed to build the model of the pruned version is a 22% of the time required by the unpruned version. The classification time of the pruned algorithm is also very lower, being the 6% of the time employed by the C4.5 unpruned.

## 6 Approach and performance on the PPI-IAS Task

For this task we applied the same process as in Gene Mention task, also using JTLlib and WEKA:

1. Document indexing. After processing the 5,500 abstracts that make up the training data, a representation based on the selected attributes is obtained and configured into the ARFF format. In this case, the input data is composed of all texts pre-classified with either `curation_relevance = 1` or `curation_relevance = 0`.

In order to build a model, we have defined a set of attributes that consists of the most relevant words (unigrams) for classification, as well as the most relevant pairs (bigrams) and trios (trigrams) of words. The lists of n-grams are determined using a correlation coefficient, and each n-tuple of words becomes an attribute, with a value of 1, if it is contained in the text, or 0 otherwise.

2. Dimensionality reduction. During the iterative process, we searched for the n-tuples with higher and lower correlation coefficient to build the attribute vector, and tried with several combinations of amounts of unigrams, bigrams and trigrams. The n-tuples with higher coefficient are likely to present higher information gain for documents classified as PPI articles, whereas those tuples with lower coefficient are representative of documents not classified as PPI articles. Table 1 shows some of the n-tuples with higher correlation coefficient.

As a consequence of the ranking of tuples, in our process to obtain the best set of attributes we made some experiments using a stemmer, but the results were unsatisfactory. Moreover, we tried to group some biomedical terms using character n-grams, but we did not get a good outcome either.

Table 1. n-tuples with higher information gain

unigrams	bigrams	trigrams
interaction	two hybrid	yeast two hybrid
domain	domain of	two hybrid system
binding	yeast two	we show that
hybrid	with the	the yeast two
yeast	interacts with	a yeast two
with	in vitro	the interaction of
interacts	the interaction	to interact with
terminal	interact with	in vitro and
complex	interaction between	is required for
interact	interaction with	two hybrid screen

3. Classifier learning. After several experiments with different Machine Learning algorithms, such as Naïve Bayes, C4.5 decision tree and Adaboost, Adaboost with Naïve Bayes showed to be the most effective. Then, we continued our experiments only with the latter, considering different attributes vectors.
4. Evaluation of text classifiers. Table 2 shows the evaluation results obtained using the test data set. The linear increment on the amount of n-tuples used increases precision and F-Measure, but results in a poorer recall. The increment in the amount of bigrams and trigrams produces a higher recall again, but with lower precision and F-measure. The experiments realized with the training set proved that increasing the number of attributes would produce similar results, but not necessarily better.

Table 2. Effectiveness on the test data

	unigrams/bigrams/trigrams		
	3628 / 14240 / 6817	5140 / 18510 / 10224	5140 / 24206 / 18174
<b>Precision</b>	0.555	0.583	0.539
<b>Recall</b>	0.981	0.952	0.984
<b>F-measure</b>	0.709	0.723	0.696

## 7 Conclusions and Future Work

The participation of our team in the Biocreative competition has primarily served us as a proof-of-concept for our systematic approach to feature engineering in text classification tasks. We believe we have obtained reasonable results with respect to the effort we have invested in the competitions. We intend to join other competitions in order to refine our approach, and to improve the feature modeling library. In particular, we address the next two competitions:

- The International Challenge on Classifying Clinical Free Text Using Natural Language Processing<sup>1</sup>, organized by the Computational Medicine Center (Cincinnati, Ohio, US). The goal of the challenge is to create and train computational intelligence algorithms that automate the assignment of ICD-9-CM codes to clinical free text. This is a kind of Text Categorization task, very related to the goals of our research projects. The challenge is currently running (as by mid February).
- The second edition of the i2b2 Natural Language Processing Challenge<sup>2</sup>. This challenge is promoted by the Informatics for Integrating Biology and the Bedside Center, an NIH-funded National Center for Biomedical Computing (based at Partners HealthCare System). The first edition has proposed two tasks: the medical record anonymization (a form of Named Entity Recognition, in which entities are after changed to preserve patient privacy), and the Smoking Status detection (a form of Text Categorization, in which a label stating the smoking status of the patient has to be assigned to medical records). We have taken part in this latter task, with average results.

Once we have further tested our library, and we can fully trust it and provide some teaching material, we will release it as opensource software, intended to complement WEKA for text analysis.

In the framework of the project Sinamed, we also plan to build several Information Access systems in biomedicine, specifically targeting the medical doctors when preparing their patient cases and when researching, and for students preparing and documenting assignments. We intend to connect the clinical record information with related scientific information, by using Text Categorization of documents according to the SNOMED ontology concepts. We also have to meet the privacy requirements of patients' information by using the anonymization techniques proposed by others in the i2b2 Challenge.

## References

- [1] Avancini, H.; Rauber, A. and Sebastiani, F. *Organizing Digital Libraries by Automated Text Categorization*. Proceedings of ICDL 2004, TERI, 2004, 919 - 931.
- [2] Gowda, K. C. and Diday E. *Symbolic clustering using a new similarity measure*. IEEE Tr. SMC, Vol 22, No 2, 1991.
- [3] Lewis, D.D.: *Representation and learning in information retrieval*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, US, 1992.
- [4] Sebastiani, F.: *Machine learning in automated text categorization*. ACM Computing Surveys 34 (2002) 1-47
- [5] Buenaga, M.; Maña, M.; Gachet, D. and Mata, J. *The SINAMED and ISIS Projects: Applying Text Mining Techniques to Improve Access to a Medical Digital Library*. 10th European Conference, ECDL 2006, Alicante, Spain, September 17-22, 2006. 548-551.
- [6] Witten, I. H. and Frank, E. *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufman, San Francisco, CA, USA, 2000.

<sup>1</sup> <http://www.computationalmedicine.org/challenge/index.php>.

<sup>2</sup> <http://i2b2.org/NLP/>.