OpinAIS: An Artificial Immune System-based Framework for Opinion Mining

Alejandro Baldominos Gómez, Nerea Luis Mingueza, Mª Cristina García del Pozo

Universidad Carlos III de Madrid, Leganés, Spain

***Abstract* — This paper proposes the design of an evolutionary algorithm for building classifiers specifically aimed towards performing classification and sentiment analysis over texts. Moreover, it has properties taken from Artificial Immune Systems, as it tries to resemble biological systems since they are able to discriminate harmful from innocuous bodies (in this case, the analogy could be established with negative and positive texts respectively). A framework, namely OpinAIS, is developed around the evolutionary algorithm, which makes it possible to distribute it as an open-source tool, which enables the scientific community both to extend it and improve it. The framework is evaluated with two different public datasets, the first involving voting records for the US Congress and the second consisting in a Twitter corpus with *tweets* about different technology brands, which can be polarized either towards positive or negative feelings; comparing the results with alternative machine learning techniques and concluding with encouraging results. Additionally, as the framework is publicly available for download, researchers can replicate the experiments from this paper or propose new ones.**

***Keywords* — Artificial immune system, evolutionary computation, sentiment analysis, machine learning, classification.**

I. Introduction

SENTIMENT ANALYSIS (also referred as *opinion mining*) [27] is a field of Natural Language Processing (NLP) which aims at extracting emotional or subjective information from a source, which may be a document, a website, a publication in a social network, etc.

A specific task within sentiment analysis is retrieving the polarity of the document, i.e., whether it expresses a positive or negative feeling (sometimes, the case when the document does not express any feeling at all is also observed). This is definitely not a simple task, as natural language semantics are very complex, and there are many ways, sometimes too rhetorical, to express a positive or negative feeling. In fact, sentiment analysis involves so many challenges that many works over the last decade have discussed them [28, 22, 24, 38] and most if not all of those difficulties remain invariant and are widely discussed today [3], as social networks start to set up enormous corpus which are increasingly interesting for this task [26, 19].

From the computational side, a Machine Learning (ML) approach perfectly fits this task. The problem of guessing the polarity of a document is analogous to a binary classification problem. Yet some decisions, such as how the features for classification are retrieved from the document, or which particular ML algorithm will be used must be taken before some results could be obtained.

This work aims at applying an Artificial Immune System (AIS) approach, which is a biologically-inspired ML technique based on the immune system of vertebrates, to solve this problem. Actually, the algorithm can be easily extended to support multiclass classification and prediction problems. To provide additional value, this work also have the purpose of developing a framework which can be extended to new algorithms and applications, so that it can be reused by the scientific community.

A brief introduction of AIS, as well as some related work is provided in section 2. Sections 3 and 4 discuss how features can be extracted from text, and how the immune-based algorithm is applied for the sentiment analysis task. Meanwhile, section 5 focuses on the design and the development of OpinAIS, the AIS-based framework for solving sentiment analysis problems.

Finally, section 6 shows some results obtained from using OpinAIS with two public datasets involving voting records for the US Congress and a Twitter corpus. Secion 7 provides some conclusive remarks on this work, and appendices are included which detail how to run and extend the framework.

II. Related Work

The purpose of this section is to briefly discuss the problem of sentiment analysis, analysing previous works where ML techniques were used to face this problem, and finally describing how AIS work and some state of the art applications where AIS are applied for the task of sentiment analysis.

As it was stated in the previous section, sentiment analysis (or opinion mining) is a problem that involves many of the challenges brought by NLP. Techniques located within the field of Artificial Intelligence (AI) are well suited for facing this problem [5]. In particular, detecting the polarity of a text (whether it contains positive or negative feelings) can be in most cases reduced to a problem of binary classification by using bag of words (where binary attributes indicate whether a particular word appear or not in the document). By doing so, many classical ML techniques can be applied [18], including

DOI: 10.9781/ijimai.2015.333

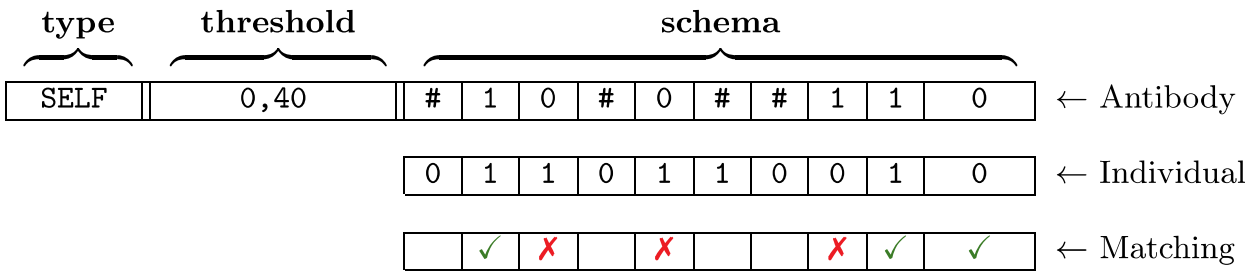

Fig. 1. Example of an antibody detecting an individual.

variations of the Naïve Bayes classifier [40], Support Vector Machines [37], Kernel Trees [1] or semi-supervised approaches [7, 21]. Some works compare several of these techniques for detecting emotions and personality in

platforms such as Whatsapp or SMS [33]. Other works do not use bag of words but rather different approaches such as graph-based techniques for *tweet* classification [6], while in this case the work do not focus on opinion mining but rather topic detection. In the recent years, several surveys summarizing the most relevant techniques and contributions have been published [23, 25, 42, 16].

Biologically inspired AI techniques have also proved to be relevant for solving this problem, as many works in the last couple of years use these approaches for opinion mining, such as it is the case of genetic algorithms [14], particle swarm and ant colony optimization [39], neural networks [2] or a combination of several of them [17].

Regarding biologically-inspired artificial intelligence, relevant techniques include AIS which appeared in the mid-90s, when efforts for understanding the immune system [12] significantly increased. The idea beyond these systems is to imitate the biological adaptive immune system and its ability to recognize external harmful individuals, which can be generalized to approach and solve a variety of problems.

The present work is based on a previous research on applying AIS to document classification [41], which was already based on an earlier work that applied AIS-based techniques for concept learning [30]. In these approaches, which will be described in further detail in section 4, a population of antibodies is evolved with a co-evolutionary technique. Eventually, a set of antibodies conform a classifier, which can be used to infer the class a certain item. As long as a document classification problem can be represented as a binary string, the system can learn a classifier from a set of training instances. Additionally, there are more recent works which study the convenience of using AIS for opinion mining [34], and use this kind of techniques for selecting features for opinion mining [35] or analysing sentiments in newspapers [31].

The use of an evolutionary algorithm somehow recalls from other AIS techniques such as clonal selection, as the evolutionary operators resemble the operators in algorithms such as *CLONALG* [8]. Additionally, the process of affinity maturation is achieved by the evolutionary algorithm, which tries to increase the fitness of the antibodies, i.e., their ability to correctly detect antigens.

Finally, prior work proposed a theoretical framework for AIS [8], and other ML frameworks such as Weka [15] also incorporates AIS-based techniques for general-purpose classification as well as specific text mining algorithms which would enable performing document classification or opinion mining [32]. However, the framework proposed in this work is more specifically aimed towards document classification (and opinion mining in particular) and therefore is simpler to be used and to be extended, whereas others are more complete and supports other problems beyond classification itself but fail to provide such specific parameterization for opinion mining.

III. Data Wrangling

For applying ML to documents expressed in natural language, a preliminary phase of data wrangling is often required so that these can be converted to a format accepted by the algorithm. For this work, the input is converted into a binary string (a list of boolean features).

The current section details the process followed to obtain a set of binary individuals from a set of documents expressed in natural language.

A. Preprocessing

When dealing with natural language, some processing of the input may lead to better results, as raw data is typically too noisy. An approach to this processing involves implementing a series of filters running in a pipeline [11], each of those performing some processing over data, which is then inputted to the next filter. The ultimate goal of these phases is to increase the ratio of meaningful words by reducing the total number of different words, while trying to keep semantics. This section describes the preprocessing phase applied in this work, and how it could help to improve the results.

1) **Removing Non-Alphanumerical Symbols**: usually, non-alphanumerical symbols in a text lack from any semantic meaning thus can be ignored. However, other words formed only by symbols (e.g. emoticons such as :-) or xD) not only do have semantic meaning, but also store a strong emotional load [20].
2) **Converting to Lower Case**: in many cases, words keep semantics regardless whether they are written uppercase or lowercase. For this reason, it is useful to turn all symbols to the same case, to represent the same word always with the same characters.

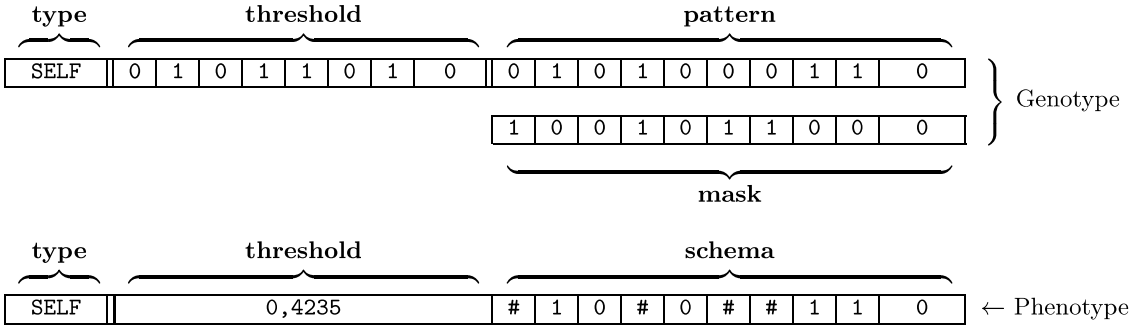

Fig. 2. Example of a translation between an antibody's genotype and phenotype

3) **Removing Stop Words**: in natural languages, there are many words that are completely meaningless, and only have syntactical value, such as determiners, prepositions, etc. These words usually appear with very high frequencies, and so may lead the algorithm to think that they are relevant. While there are works involving the automatic identification of stop words [44], the approach followed in this work uses a stop words dictionary for the English language.
4) **Stemming Words**: stemming is the process to reduce a word to its stem (e.g. "work", "working" and "worked" are all reduced to "work"). This way, the space of words is considerably reduced, while the original meanings persist, increasing the ratio of semantics versus the number of different tokens. This work uses the Porter Stemming Algorithm [29], as it is widely used and considered the *de facto* standard for stemming English words.

B. Extracting the Features

After the preprocessing phase, the input is still a set of documents, each of these reduced to a set of tokens (stems) resulting from applying the filters to the original words. The purpose of the second step is to decide which of the tokens are more relevant when deciding the class of each document. To do so, a metric known as *expected information gain* may be used, which estimates the information provided by a token based on the entropy of the set of documents containing and not containing that token.

In particular, the expected information gain for a word w and a set of documents S is calculated as follows:

$$E(w,S) = I(S) - \left(P(w)I(D_w)\right) + \left(P(\neg w)I(D_{\neg w})\right)$$

where:

- $P(w)$ is the probability that w appears in a document, i.e., the percentage of documents containing w.
- $P(\neg w)$ is the probability that w does not appear in a document, i.e., the percentage of documents not containing w.
- $D(w)$ is the subset of documents containing w.
- $D(\neg w)$is the subset of documents not containing w.
- $I(S), S = \{D, D_w, D_{\neg w}\}$ is the entropy of the set S for each of the classes, which is defined as follows:

$$\mathrm{I(S)} = \sum_{\mathrm{c}\in\{+,-\}} -\mathrm{P(S_c)}\log_2(\mathrm{P(S_c)})$$

The computation of the entropy can only be performed if a training set exists where the class is known in advance for each document in the set, i.e., under a supervised learning scheme. While the previous equation refers to the class as either positive or negative, it could be generalized to any arbitrary number of different classes.

Finally, when the expected information gain is computed for all words, the n words with the highest value of E are chosen, which can be expressed as $F = \{w_1\ w_2\ w_3\ ...\ w_n\}$.

During the last phase, known as vectorization, the objective is to convert documents to individuals represented by a binary string: $d = \{b_1\ b_2\ b_3\ ...\ b_n\}$. To do so, for each bit b_i in the individual representing the document, $b_i = 1$ if the document contains the word w_i, or $b_i = 0$ if it does not. After the vectorization process takes place, the original set of documents is converted into a set of binary individuals.

IV. The Artificial Immune System

For the development of AIS, the approach provided in [41] is followed. This section provides first an intuition of how antibodies are represented in the AIS and how they can be used to detect individuals of a certain type. Later, it describes the process to obtain a classifier, composed of a set of antibodies, given a set of training examples.

A. Design of Antibodies

The design of antibodies is a key task in the development of an AIS, as they are the entities responsible for detecting the type of the individuals and, in the end, of the classification task. In the AIS developed for this work, antibodies (also called *detectors*), integrate the next elements:

- A **type** indicating the polarity of the individuals this detector should recognize, which usually are *self* (i.e., part of the body) or *non-self* (i.e., foreign to the body and thus potentially harmful), which in the task of sentiment analysis are identified to positive and negative items respectively. Nevertheless, this definition can be extended to support a set of k different classes.

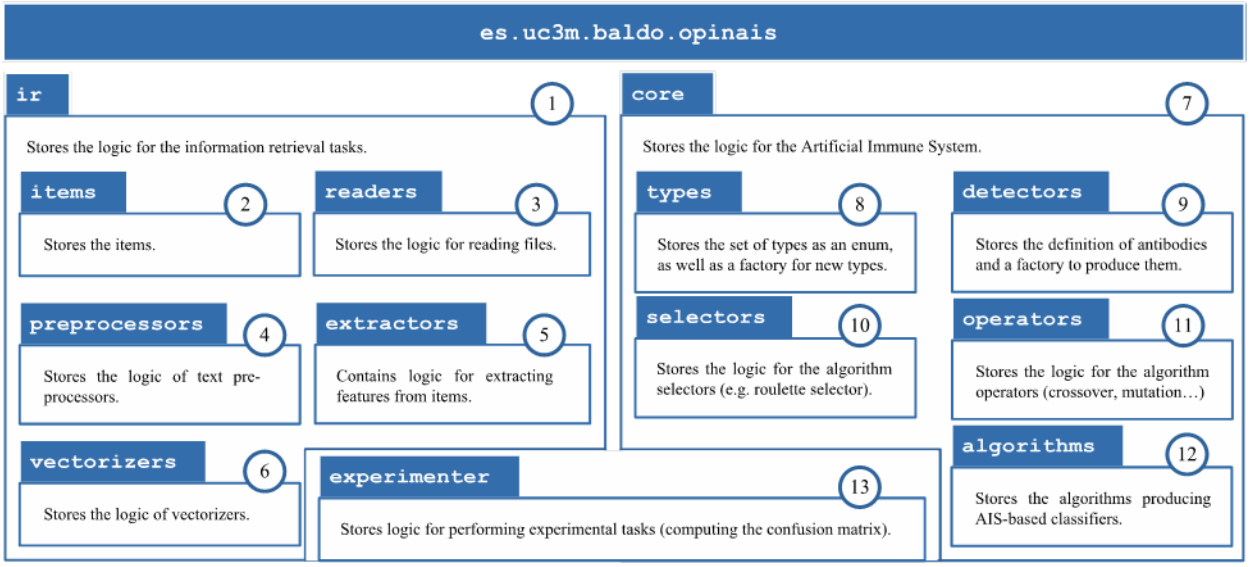

Fig. 3. Package structure for the OpinAIS framework

- A binary **schema** of the same length that the number of features of an individual. While this schema is binary, besides *0*s and *1*s it can also contain wildcard (*#*) positions.
- A real threshold in the interval [0,1].

After a representation for the antibodies is chosen, it is important to decide the process by which an antibody detects an individual as being from its type. The steps for this process are detailed next:

1) Each bit in the schema is compared to each bit of the individual in the same position. Bits in wildcards positions, marked as *#*, are ignored.
2) The matching ratio is computed as the number of bits matching in the same position divided by the total number of comparisons performed (i.e., the number of non-wildcard positions in the antibody's schema).
3) If the matching ratio exceeds the threshold, then the antibody detects the individual as being of its same type. Otherwise, it does not detect it.

An example of individuals matching is provided in Figure 1. In this particular example, there are 3 matches from a total of 6 comparisons, so the matching ratio is $3/6 = 0.5$, which is greater than the threshold ($0.5 > 0.4$), and thus the individual is recognized as *self*.

B. Evolving the Classifier

A classifier is a set of antibodies, one for each possible type. When an individual is inputted to the classifier, each antibody tries to detect it. The type of the individual is obtained as the type of the antibody who detects the individual and maximizes the matching ratio. It is important to notice that, in the case that no antibody detects the individual, then it remains unclassified.

An evolutionary approach is chosen to obtain the classifier. Before the details of the algorithm are described, it is important to decide the way the antibody is represented in order to be treated by the evolutionary algorithm, i.e., its genotype. Antibodies are encoded as follows:

- The **type** does not need to be encoded, as the evolutionary operators do not affect it.
- The **threshold** is encoded as an 8-bit number in Gray code, as small changes in this binary representation lead to small changes in the number it represents. Because an 8-bit string represents an unsigned integer n the interval [0, 255], the resulting value is normalized in the range [0, 1], thus dividing it by 255.
- The **schema** is represented by two different binary strings, named **pattern** and **mask**, both with the same length that the schema. Given a pattern and a mask, the schema can be determined as follows:
  1. If the *i*-bit in the mask is *1*, then the corresponding bit in the schema will be a wildcard (*#*).
  2. If the *i*-bit in the mask is *0*, then the corresponding bit in the schema will correspond to the *i*-bit from the pattern.

  It must be noticed that with this encoding, many different genotypes may translate into the same phenotype. Actually, this is common in natural immune systems, as different chains of amino acids may fold into antibodies recognizing the same pattern [30].

An example of a translation between the genotype and the phenotype is shown in Figure 2. Once the binary representation for the antibodies is depicted, the details of the evolutionary algorithm can be discussed. This algorithm follows the next steps:

1) **Initialization**: to begin with, the algorithm generates an initial population of antibodies, of a fixed (yet configurable) size. While this initialization is performed randomly, it attends to some parameters:
   - The **type bias** represents the probability that the new antibody detects *self* individuals. For instance, if the type bias has a value of 0.6, then it means that in average, 60% of the antibodies in the population will detect *self* individuals.

- The **generality bias** represents the probability that a bit in the schema is a wildcard (#). For instance, if the generality bias has a value of 0.3, then it means that in average, 30% of the bits in an antibody schema will be wildcards.

2) **Fitness calculation**: once the initial population is generated, the algorithm calculates the fitness of each antibody. This fitness is calculated as the number of correctly classified individual minus the number of false positives. Unclassified individuals are considered as correctly classified if they are not of the same type that the antibody. To prevent negative values for the fitness, it is normalized in the range [0, 1].
3) **Selection**: two antibodies from the same type are chosen, in a random yet fitness-proportional manner. To do so, a technique known as *roulette selection* is performed, by which antibodies with higher fitness have more chances to be selected.
4) **Reproduction**: the two selected antibodies serve as parents for a new one. This reproduction is performed using *standard crossover*, by which the genome of the child antibody is filled by selecting, for each bit, one random bit in the same position from either of their parents. A parameter, known as the **crossover rate**, establishes the probability that crossover takes place. In the cases where crossover is not performed, the child results as an exact copy of one of their parents.
5) **Mutation**: the child is mutated, by performing bit flipping for each individual bit. In this case, a parameter known as **mutation rate** controls the probability that a single bit is flipped.
6) **Generational replacement**: steps 3-5 are performed until the new population has as many antibodies of the same type as the previous one. When such a thing occurs, the original population is replaced with the new one. During this phase, *elitism* can be introduced through a parameter, known as **elitism rate**, which controls the percentage of best antibodies that are kept between generations. By introducing elitism, the best detectors are maintained.
7) **Stop condition**: if the maximum number of generations is not achieved, the algorithm restarts from step 2. Otherwise, the algorithm stops and a classifier is built by choosing the best antibody from each type. Each bit in the schema is compared to each bit of the individual in the same position. Bits in wildcards positions, marked as #, are ignored.

An improved version of this algorithm, which includes a cooperative approach, has been also developed. This algorithm, which is known as *co-evolutionary algorithm*, inserts a new phase after the fitness calculation. Indeed, it performs a second fitness calculation, which computes a *cooperative fitness* rather than an individual one.

Particularly, the cooperative fitness for a certain antibody is the result of classifying all the individuals with that antibody combined with the best antibodies of the remaining types. The fitness sums up all the hits and subtracts the misses (i.e., individuals wrongly classified). It remains as a user-configurable parameter to decide whether unclassified individuals are considered as wrongly classified or are ignored (some applications may benefit from ignoring unclassified instances, such as those where unclassified instances are preferred over misclassified ones). Finally, the fitness is normalized in the range [0, 1].

The cooperative fitness evaluates a potential classifier rather than each antibody itself. For this reason, results are usually better, but computing time can also be significantly higher.

V. The OpinAIS Framework

OpinAIS is an extensible framework that enables the application of AIS to a variety of classification problems, as long as instances can be represented as binary strings.

The power of OpinAIS remains in its extensibility. While the algorithms described in the prior section are already implemented, it is relatively simple to develop new ones. This principle not only applies to algorithms, but also to input readers, information retrieval processors, etc.

Figure 3 shows the package structure of the framework. The heading of the figure refers to the path where the OpinAIS framework is placed in the package. The purpose of this section is to describe the responsibility of each one, so that it can serve as a quick developer guide. The *ir* package (1) stores the logic required to retrieve a set of individuals (which are computable by the algorithm) from an input source, such as a text file containing one individual in each line. This package is divided in several subpackages, with clearly defined responsibilities:

- Package *items* (2) contains items, which a generic type of individuals, i.e., *something* which can be potentially converted to an individual, but so far is not, such as a *tweet* or an HTML document are some kinds of items.
- Package *readers* (3) will store readers, whose responsibility is to generate a set of items from an input source. For instance, there may be a folder containing HTML documents, and a reader that returns a set of objects representing those.
- Package *preprocessors* (4) stores classes responsible for performing some preprocessing tasks over text items which may significantly increase the performance of the classifier, as it was shown in section 3.
- Package *extractors* (5) contains classes whose purpose is to extract features from a set of items in those cases when specific logic for this task is required (e.g. implementation details on the extractor for text items was provided in section 3).
- Package *vectorizers* (6) contains the logic for converting items into individuals encoded as a set of bits. The behaviour of vectorizers for text items was explained in section 3.

On the other hand, the *core* package (7) stores the logic required for obtaining a classifier from a set of input individuals. Most details on this process were already given in section 4, so this section will limit to explain how this

functionality is broken into different packages:

- Package *types* (8) contains an enumerated type, *Type* with the set of all possible classes for individuals. For the sake of flexibility, this enumerated type is empty and it is filled dynamically when classes are known.
- Package *detectors* (9) contains the *Detector* class, representing the definition of an antibody, as it was described in section 4. Moreover, the *DetectorFactory* class implements some logic for initializing the first population of antibodies.
- Package *selectors* (10) contains classes which implement some logic for choosing an antibody from a population. So far, the evolutionary algorithm developed uses a roulette selector, yet many others could be implemented by the user (e.g. a tournament selection).
- Package *operators* (11) stores auxiliar logic containing operators used by the algorithms. An example of such operators is the standard crossover and mutation, which were already described in section 4, and which are implemented in the classes *CrossoverOperator* and *MutationOperator* respectively.
- Package *algorithms* (12) stores the algorithms, whose responsibility is to receive a set of training individuals and build a classifier from them.

Finally, the *experimenter* package (13) contains additional logic for assisting the experimental tasks, such as:

- dividing a set of individuals into training and test sets, where the size of these tests can be set by the user.
- given a classifier and a set of individuals, computing the confusion matrix, i.e., a table showing up the number of correctly classified instances, as well as false positives, true negatives and unclassified individuals.
- computing the performance of a classifier measured as its accuracy, given the corresponding confusion matrix.

The OpinAIS framework is publicly available for download from a GitHub repository[1], and more information and developers documentation can be found in the project website[2], including instructions on how to run and extend it.

VI. Experimental Results

Once the OpinAIS framework is developed, experiments over two different datasets are performed using the implemented AIS-based algorithm in order to validate the system, and a comparative evaluation with classic ML techniques is also carried out. Additionally, further evaluation for the algorithms underlying this proposal has already been published in previous works [30, 41].

A. US Congressional Voting Records Dataset

The first experiment in this section will execute over the *US Congressional Voting Records Data Set* from the *UCI Machine Learning Repository* [36], which is composed of 117 (34.21%) instances of republican votes and 225 (65.79%) instances of democrat votes. As instances are fairly unbalanced, two experiments are executed: the first one will deal with all input instances, while the second will balance them, thus taking 117 instances of republican congressmen and the same number of democrats.

TABLE I
Confusion matrix for the US Voting Records dataset with unbalanced data

	Republican	Democrat	N/C
Republican	86/22	5/3	1/0
Democrat	2/2	179/41	1/0

TABLE II
Confusion matrix for the US Voting Records dataset with balanced data

	Republican	Democrat	N/C
Republican	85/26	3/0	2/1
Democrat	2/0	95/19	1/0

The results for an average execution are shown in Table I and Table II, which contains the confusion matrix for the experiment with unbalanced and with balanced data respectively. Cells in the confusion matrix contain two values, the first one referring to the training set and the second to the test set. The last column refers to non-classified instances.

As it can be seen, the results are pretty good. For the original dataset, the accuracy is 96.72% (92.65%), while for the balanced subset the value is 95.74% (97.83% for the test set). From these numbers, two conclusions can be drawn: in the first place, the AIS-based evolutionary algorithm provides good classification accuracy, which validates that the system is working properly. Secondly, the algorithm shows a good generalization ability, as long as it does not fall into overfitting the individuals from the training set, achieving very similar results for both the training and the test sets.

B. Twitter Sentimental Corpus

The second battery of experiments is performed over a set of actual *tweets*, which are short publications in Twitter. This dataset is provided by Sanders Analytics[3], consists in 5,513 tweets about technological companies and it is especially interesting as it provides two different classifications. The first one has to do with the polarity of the *tweet*, which can either be *positive*, *negative*, *neutral* or *irrelevant*. In most cases, *tweets* are classified as *irrelevant* if they are written in languages other than English or have nothing to do with the topic (i.e., they are spam). Also, *tweets* are classified as *neutral* when they are neither positive nor negative in a clear way, they are simple factual statements or they express questions with no strong emotions. The second classification criterion has to do with the technological enterprise related to the content of the *tweet*.

An example of a *tweet* for each polarity and enterprise is

[1] http://github.com/alexbaldo/opinais
[2] http://baldo.uc3m.es/opinais/
[3] http://www.sananalytics.com/lab/twitter-sentiment

shown in Table III. As it can be seen, the classification task may find some difficulties. In the first place, a *tweet* with negative polarity contains a high load of positive words such as "greatly impressed". Secondly, as irrelevant *tweets* can be in any language, they add a huge number of possible words that may harden the features extraction task.

TABLE III
TWEETS FOR EACH POLARITY AND ENTERPRISE

Brand	Polarity	*Tweet*
Apple	Positive	@apple @siri is efffing amazing!
Microsoft	Neutral	Creating #Pareto charts using #Microsoft #Excel
Google	Negative	Not greatly impressed with #Google and #Samsung presentation skills.
Twitter	Irrelevant	#twitter sos un vicioooooo

TABLE IV
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* ENTERPRISE USING OPINAIS

	A	G	M	T	NC
A	699/75	42/2	27/10	44/3	165/29
G	27/4	742/68	59/5	96/12	196/25
M	50/6	59/6	791/93	24/1	208/19
T	20/4	18/2	82/9	825/93	128/11

TABLE V
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING OPINAIS

	Positive	Negative	NC
Positive	296/30	107/20	42/1
Negative	72/9	385/39	32/4

TABLE VI
RESULTS FOR THE EVALUATION OF THE TWEET'S POLARITY PREDICTION USING OPINAIS

	Train	Test		Train	Test
ACC	72.9%	67.0%	TPR	73.5%	60.0%
PPV	80.4%	76.9%	TNR	84.3%	81.3%
NPV	78.3%	66.1%	MCC	58.2%	48.9%

TABLE VII
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING NAIVE BAYES

	Positive	Negative
Positive	196/80	144/76
Negative	47/16	339/139

For both experiments, a population of 200 individuals each one recognizing 250 features is evolved during 1000 generations. The dataset is divided into a training set containing 70% of the original instances and a test set with the remaining 30%.

First, a classifier is trained to infer the enterprise related with the *tweet*, either Apple (A), Google (G), Microsoft (M) or Twitter (T). The number of instances for each enterprise is approximately the same, i.e., the dataset is balanced. The confusion matrix for this problem is shown in Table IV. As it can be seen the results are good, providing a great improvement over random guess with an accuracy of 92.41% (90.41% over the test set, which also shows a good generalization).

TABLE VIII
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING C4.5

	Positive	Negative
Positive	233/90	107/66
Negative	50/37	336/18

TABLE IX
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING RANDOM FOREST

	Positive	Negative
Positive	317/126	23/30
Negative	29/34	357/121

TABLE X
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING SVM

	Positive	Negative
Positive	143/61	197/5
Negative	40/12	346/143

TABLE XI
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING MLP

	Positive	Negative
Positive	198/79	142/77
Negative	26/13	360/142

TABLE XII
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING CLONALG

	Positive	Negative
Positive	181/81	159/75
Negative	78/37	308/118

TABLE XIII
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING AIRS

	Positive	Negative
Positive	145/65	195/91
Negative	44/15	342/140

TABLE XIV
CONFUSION MATRIX FOR THE TWITTER SENTIMENT CORPUS WHEN CLASSIFYING THE *TWEET'S* POLARITY USING IMMUNOS-81

	Positive	Negative
Positive	113/55	227/101
Negative	36/11	350/144

Finally, the algorithm is executed with the objective of inferring the polarity of a tweet. For this experiment, only positive and negative *tweets* are considered. This is due to a couple of reasons, the first one being that neutral and irrelevant *tweets* take the most part of the corpus (almost 80%).

TABLE XV
CLASSIFICATION METRICS COMPARED FOR DIFFERENT MACHINE LEARNING TECHNIQUES. BOLD CELLS INDICATE THE BEST VALUE FOR EACH METRIC

	ACC	PPV	NPV	TPR	TNR	MCC
OpinAIS	73.31%	69.52%	79.03%	**83.33%**	63.23%	47.54%
Naive Bayes	70.42%	83.33%	64.65%	51.28%	89.68%	44.33%
C4.5	66.88%	70.87%	64.13%	57.69%	76.13%	34.40%
Random Forest	**79.42%**	78.75%	**80.13%**	80.77%	78.08%	**58.86%**
SVM	65.59%	83.56%	60.08%	39.10%	92.26%	36.99%
MLP	71.06%	**85.87%**	64.84%	50.64%	91.61%	46.29%
CLONALG	63.99%	68.64%	61.14%	51.92%	76.13%	28.91%
AIRS	65.92%	81.25%	60.61%	41.66%	90.32%	36.59%
Immunos-81	63.99%	83.33%	58.78%	35.26%	**92.90%**	34.44%

While this handicap could be solved by balancing the classes, a second problem appears: words from the irrelevant *tweets* will take a considerable part of the features array, if not the whole. This happens given that irrelevant *tweets* contains many words in different languages, which only appear in *tweets* from that class and which turn out to be very good discriminators. For this reason, the classifier would specialize in neutral and irrelevant *tweets* rather than on positive or negative ones, which was the original problem.

The number of positive and negative *tweets* is balanced (47.83% vs. 52.17% respectively). The confusion matrix for this problem is shown in Table V, and Table VI shows different metrics for evaluating the prediction quality [13] for both the train as test sets, including the accuracy (ACC), the positive predictive value (PPV, also known as precision), the negative predictive value (NPV), the true positive rate (TPR, also known as recall), the true negative rate (TNR, also known as specify) and the Matthews correlation coefficient (MCC). Unclassified instances are treated as if they were misclassified. It can be seen that the results are fairly good and the generalization power is acceptable as well, as the performance over the test set is pretty similar to that over the training set.

Finally, a comparative evaluation is performed to check how the results obtained using OpinAIS for the task of sentiment analysis compare to those achievable using classic ML techniques. For this task, the Weka framework is used [15]. An ARFF file is generated directly from the OpinAIS framework, after the preprocessing and vectorization phases. Thus, the resulting ARFF is a file where instances have 200 binary attributes (one per each selected word) and a class, either *positive* or *negative*.

For the comparative evaluation, classic ML techniques have been used (Naive Bayes, C4.5 decision trees and random forest), as well as Kernel-based techniques (support vector machines) and biologically-inspired algorithms (multilayer perceptron). Also, alternative immune classifiers[4] have been compared, namely CLONALG [9, 10], AIRS [43] and Immunos-81 [4]. In all cases, default parameters are used.

Confusion matrices displaying the classification results for each of this techniques are shown as follows: Table VII shows the confusion matrix using Naive Bayes, Table VIII for C4.5 decision trees, Table IX for random forest, Table X for support vector machines (SVM), Table XI for multilayer perceptron (MLP), Table XII for CLONALG, Table XIII for AIRS and Table XIV for Immunos-81.

All these results are synthesized in Table XV where the metrics for evaluating the prediction quality described above are computed for the results obtained in the test set for each technique. Results show that for class-independent metrics (accuracy and Matthew's correlation coefficient), OpinAIS performs better than most of the other classifiers, with the only exception of random forests. In a per-class basis, OpinAIS provides the best results for the true positive rate (TPR), i.e., is able to classify most positive *tweets* correctly, outperforming the other techniques. On the other hand, the results are worse for the true negative rate (TNR), meaning that many negative *tweets* are either misclassified or not classified at all. Also, the negative predictive value (NPV) compares quite well to the alternative techniques, only surpassed by random forest, meaning that most of the *tweets* predicted as negative are really negative. From all the techniques compared, OpinAIS is the second with highest generalization power (measured as the absolute difference between the accuracy of the training and the test set), outperforming all its competitors expect for Immunos-81.

VII. CONCLUSIONS AND FUTURE WORK

As a result of the present work, a framework for applying IS to a variety of classification problems, including those involving sentiment analysis or some natural language processing, has been developed. While this framework is initially built as an implementation of an evolutionary algorithm, it has been refactored to keep extensibility as the main priority. This way, scientists can easily adapt the framework to their needs, either adding new algorithms or information retrieval processes or supporting new input data.

The first framework prototype has been evaluated by using two different public datasets. Results are encouraging, as binary classification metrics for the evaluated datasets are always greater than 50% and in some cases close to 100% and MCC is significantly higher than zero, and the built classifiers proved to generalize fairly well the concepts they learnt. When these results are compared to other machine learning techniques, OpinAIS outperforms them in terms of accuracy, with the only exception of the random forest classifier, and in any case behaves significantly better in a class-independent basis than its immune-based competitors.

[4] Added to WEKA as a plugin available at: http://wekaclassalgos.sourceforge.net

As the framework remains in a phase of active development, many improvements can be proposed as future work. For instance, many new algorithms, parameters and information retrieval features can still be added, and n-grams rather than words could be used for vectorization.

APPENDIX

A. Running the Framework

The OpinAIS class, which is placed in the root package (es.uc3m.baldo.opinais), contains the entry point for the application. This class only requires an argument, which is the path for a *properties* file (a special type of file in Java, very similar to *.ini* files), containing a bunch of parameters.

CODE I
A SAMPLE PROPERTIES FILE

```
# Set of possible types (classes).
types=apple,google,microsoft,twitter

# Reader which will retrieve the items from the input source.
reader=TweetReader

# Factory which will process and convert the items to individuals.
factory=TextIndividualsFactory

# Source file with input.
inputFile=data/SandersAnalytics/tweets_brand.txt

# Maximum number of individuals. 0 means all.
individualsSize=0

# Must the number of individuals for each type be balanced?
isBalanced=false

# Size of the population of detectors.
speciesSize=200

# Length of the features vector.
featuresLength=1000

# Preprocessors to be applied, in order.
preprocessors=LowerCaser,StopWordsRemover,Stemmer

# Percentage of the individuals to be used in the test set.
testPct=0.1

# Name of the algorithm to be used.
algorithm=EvolutionaryAlgorithm

# Types of the arguments required by the constructor
algorithmTypes=Integer,Double,Double,Double,Double,Double
```

A fragment of a sample properties file is shown in Code I. Besides, the distributed source code contains also properties files for some applications, which the user may want to take a look at to get a better understanding of all the parameters that can be customized.

B. Extending the Framework

The purpose of this section is to provide a brief overview to developers on how they can extend the framework to support new inputs.

For this example, the *US Congressional Voting Records Data Set* [36] described before has been chosen. This dataset contains a set of instances representing a certain congressman, which can be either republican or democrat. Each of these stores the particular vote of the congressman for 16 different votations, where this vote can be a *yes*, a *no*, or an abstention.

The steps for supporting classification over this dataset are the next ones:

1) In the first place, the developer must create a class in the *ir.items* package, which represents a vote record and may be called *VotingRecord*. This class must extend from *Item* and will store the vote for each votation.
2) Secondly, a class converting input lines into instances of the *VotingRecords* class will be implemented, and stored in the *ir.readers.factories* package, while implementing the *Factory* interface. This class may be called *VotingRecordFactory*.
3) Later, a reader in the *ir.readers* class will be developed, which must implement the *Reader* interface. This reader will eventually return a set of voting records given an input file.
4) A vectorizer must be developed to encode voting records as a binary string. To do so, the approach of [41] can be observed, where yes is represented as *01*, no is represented as *10* and abstention is represented as *00* (notice that this encoding is not arbitrary, and it has been chosen so that opposite values differs in its genomic representation as much as possible). This class can be called *VotingRecordVectorizer* and must be placed in the *ir.vectorizers* package.
5) Finally, the individuals factory implementing the interface *IndividualsFactory* must be developed, which essentially coordinates the flow between the classes above to generate a set of individuals.

ACKNOWLEDGMENT

This work was partially funded by the Spanish Ministry of Science and Innovation under MOVES project (TIN2011-28336) and European Union's CIP Programme (ICT-PSP-2012) under grant agreement no. 325146 (SEACW project).

REFERENCES

[1] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment Analysis of Twitter Data. In Proc. LSM, pages 30 - 38, 2011.
[2] E. Cambria, T. Mazzocco, and A. Hussain. Application of Multi-Dimensional Scaling and Artificial Neural Networks for Biologically Inspired Opinion Mining. BICA, 4:41-53, 2013.
[3] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New Avenues in Opinion Mining and Sentiment Analysis. IEEE Intelligent Systems, 28(2):15-21, 2013.
[4] J. H. Carter. The Immune System as a Model for Classification and Pattern Recognition. Genetic Programming and Evolvable Machines, 7:28-41, 2000.
[5] H. Chen and D. Zimbra. AI and Opinion Mining. IEEE Intelligent Systems, 25(3):74-76, 2010.
[6] H. Cordobés, A. Fernández-Anta, L. F. Chiroque, F. Pérez, T. Redondo, and A. Santos. Graph-based Techniques for Topic Classification of Tweets in Spanish. IJIMAI, 2:31-37, 2014.

[7] S. Dasgupta and V. Ng. Mine the Easy, Classify the Hard: a Semi-Supervised Approach to Automatic Sentiment Classification. In Proc. ACL-IJCNLP, pages 701-709, 2009.
[8] L. N. de Castro and J. Timmis. Artificial Immune Systems: A New Computational Intelligence Approach. Springer, 2002.
[9] L. N. de Castro and F. J. Von Zuben. The Clonal Selection Algorithm with Engineering Applications. In Proc. GECCO, pages 36-37, 2000.
[10] L. N. de Castro and F. J. Von Zuben. Learning and Optimization Using the Clonal Selection Principle. IEEE Transactions on Evolutionary Computation, 6:239-251, 2002.
[11] L. Dey and S. M. Haque. Opinion Mining from Noisy Text Data. IJDAR, 12:205-226, 2009.
[12] K. D. Elgert. Immunology - Understanding the Immune System. John Wiley & Sons, Inc., 1996.
[13] T. Fawcett. An Introduction to ROC Analysis. Pattern Recognition Letters, 27:861-874, 2006.
[14] M. Govindarajan. Sentiment Analysis of Movie Reviews using Hybrid Method of Naive Bayes and Genetic Algorithm. IJACR, 3(4):139-145, 2013.
[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. SIGKDD Explorations, 11(1):10-18, 2009.
[16] R. R. S. Jandail, P. Sharma, and C. Agrawal. A Survey on Sentiment Analysis and Opinion Mining: A Need for an Organization and Requirement of a Customer. IJETAE, 4(3):17-24, 2014.
[17] A. N. Jebaseeli and E. Kirubakaran. Genetic Optimized Neural Network Algorithm to Improve Classification Accuracy for Opinion Mining of M-Learning Reviews. IJETTCS, 2(3):345-349, 2013.
[18] J. Khairnar and M. Kinikar. Machine Learning Algorithms for Opinion Mining and Sentiment Classification. IJSRP, 3(6), 2013.
[19] E. Kouloumpis, T. Wilson, and J. Moore. Twitter sentiment analysis: The good the bad and the omg! In Proc. ICWSM, 2011.
[20] S. Leon-Rojas, U. Kirschenmann, and M. Wolpers. We Have No Feelings, We Have Emoticons ;-). In Proc. ICALT, pages 642-646, 2012.
[21] S. Li, Z. Wang, G. Zhou, and S. Y. M. Lee. Semi-Supervised Learning for Imbalanced Sentiment Classification. In Proc. IJCAI, pages 1826-1831, 2011.
[22] B. Liu. Sentiment analysis: A multifaceted problem. IEEE Intelligent Systems, 25(3):76-80, 2010.
[23] B. Liu and L. Zhang. A Survey of Opinion Mining and Sentiment Analysis. In Mining Text Data, pages 415-463. Springer, 2012.
[24] D. Maynard, K. Bontcheva, and D. Rout. Challenges in developing opinion mining tools for social media. In Proc. LREC Workshop, pages 15-22, 2012.
[25] N. Mishra and C. K. Jha. Classification of Opinion Mining Techniques. IJCA, 56(13):1-6, 2012.
[26] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In Proc. LREC, pages 1320-1326, 2010.
[27] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. FTIR, 2(1-2):1-135, 2008.
[28] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs Up? Sentiment Classification using Machine Learning Techniques. In Proc. EMNLP, pages 79-86, 2002.
[29] M. F. Porter. An Algorithm for Suffix Stripping. Program, 14(3):130-137, 1980.
[30] M. A. Potter and K. A. De Jong. The Coevolution of Antibodies for Concept Learning. In Proc. PPSN, pages 530-539, 1998.
[31] M. Puteh, N. Isa, S. Puteh, and N. A. Redzuan. Sentiment Mining of Malay Newspaper (SAMNews) Using Artificial Immune System. In Proc. WCE, 2013.
[32] A. Puurula and S.-H. Myaeng. Integrated Instanceand Class-based Generative Modeling for Text Classification. In Proc. ADCS, pages 66-73, 2013.
[33] Y. Saez, C. Navarro, A. Mochón, and P. Isasi. A System for Personality and Happiness Detection. IJIMAI, 2:7-15, 2014.
[34] N. Samsudin, M. Puteh, A. R. Hamdan, and M. Z. A. Nazri. Is Artificial Immune System Suitable for Opinion Mining. In Proc. DMO, pages 131-136, 2012.
[35] N. Samsudin, M. Puteh, A. R. Hamdan, and M. Z. A. Nazri. Immune Based Feature Selection for Opinion Mining. In Proc. WCE, 2013.
[36] J. C. Schlimmer. Concept Acquisition through Representational Adjustment. PhD thesis, University of California, Irvine, 1987.
[37] B. Schuller and T. Knaup. Learning and Knowledge-Based Sentiment Analysis in Movie Review Key Excerpts. In Toward Autonomous, Adaptive, and Context-Aware Multimodal Interfaces. Theoretical and Practical Issues, volume 6456 of LNCS, pages 448-472. Springer, 2011.
[38] N. R. Sharma and V. D. Chitre. Opinion Mining, Analysis and its Challenges. IJIACS, 3(1):59-65, 2014.
[39] G. Stylios, C. D. Katsis, and D. Christodoulakis. Using Bio-inspired intelligence for Web opinion Mining. IJCA, 87(5):36-43, 2014.
[40] S. Tan, X. Cheng, Y. Wang, and H. Xu. Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis. In Advances in Information Retrieval, volume 5478 of LNCS, pages 337-349. Springer, 2009.
[41] J. Twycross and S. Cayzer. An Immune-based Approach to Document Classification. Technical report, HP Laboratories, Filton Road, Stoke Gifford, Bristol U.K., 2002.
[42] G. Vinodhini and R. M. Chandrasekaran. Sentiment Analysis and Opinion Mining: A Survey. IJARCSSE, 2(6), 2012.
[43] A. Watkins, J. Timmis, and L. Boggess. Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm. Genetic Programming and Evolvable Machines, 5:291-317, 2004.
[44] J. Wilbur and K. Sirotkin. The Automatic Identification of Stop Words. JIS, 18:45-55, 1992.

Alejandro Baldominos Gómez (M'14) was born in Madrid (Spain) in 1990. He obtained his bachelor in computer science and engineering from Universidad Carlos III de Madrid (Spain) in 2012 and his master in computer science and technology in 2013 with a specialization in artificial intelligence from the same university. He has been working as a Research Assistant at Universidad Carlos III de Madrid in the Advanced Databases Group since 2010, and in the Evolutionary Computation and Artificial Intelligence Group since 2013. He is now a Ph.D. student with a studentship granted by the Spanish Ministry of Education, Culture and Sport. He also works as Professor of the master in visual analytics and big data at Universidad Internacional de la Rioja. He has published several conference papers in the fields of context-aware systems, artificial intelligence and big data and has been involved in several national and European research projects, such as Semants, Cadooh, Memento and SEACW. Prof. Baldominos is AAAI Student Member and IEEE Student Member.

Nerea Luis Mingueza was born in Madrid (Spain) in 1991. She obtained her bachelor in computer science and engineering from Universidad Carlos III de Madrid (Spain) in 2013 and her master in computer science and technology in 2014 with a specialization in artificial intelligence from the same university. She has held a research fellowship since 2012 in the Planning and Learning Group of Universidad Carlos III de Madrid, where she is now a Ph.D. student. She is also Professor of the bachelor in computer science and engineering, where she teaches the programming course. Her research fields are humanoid robotics and multi-agent planning, having several papers of these topics published in conference proceedings.

Mª Cristina García del Pozo was born in Madrid (Spain) in 1983. She obtained her computer management engineering degree in 2006, her master in computer security in 2007 and her industrial organization degree in 2009, all of them from Universidad Pontificia de Salamanca. She is currently studying the master programme in technological projects design and management from Universidad Internacional de la Rioja. She has participated in several national and European projects, including Yuste Digital and GBIC at Universidad Pontificia de Salamanca, and SEACW and E-Space at Universidad Carlos III de Madrid, where she currently works as Research Assistant in the Evolutionary Computation and Artificial Intelligence Group. She had previously worked as JAE-Tec in CSIC (Consejo Superior de Investigaciones Científicas) for two years developing projects, including the platform used for managing researchers' normalized curriculum vitae (CVN).