Mining M-Grams by a Granular Computing Approach for Text Classification

Antonino Capillo[a], Enrico de Santis[b], Fabio Massimo Frattale Mascioli[c] and Antonello Rizzi[d]
Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza", Via Eudossiana 18, 00184 Rome, Italy

Keywords: Text Mining, Text Categorization, Granular Computing, Knowledge Discovery, Explainable AI.

Abstract: Text mining and text classification are gaining more and more importance in AI related research fields. Researchers are particularly focused on classification systems, based on structured data (such as sequences or graphs), facing the challenge of synthesizing interpretable models, exploiting gray-box approaches. In this paper, a novel gray-box text classifier is presented. Documents to be classified are split into their constituent words, or tokens. Groups of frequent m tokens (or m-grams) are suitably mined adopting the Granular Computing framework. By *fastText* algorithm, each token is encoded in a real-valued vector and a custom-based dissimilarity measure, grounded on the Edit family, is designed specifically to deal with m-grams. Through a clustering procedure the most representative m-grams, pertaining the corpus of documents, are extrapolated and arranged into a Symbolic Histogram representation. The latter allows embedding documents in a well-suited real-valued space in which a standard classifier, such as SVM, can safety operate. Along with the classification procedure, an Evolutionary Algorithm is in charge of performing features selection, which is able to select most relevant symbols – m-grams – for each class. This study shows how symbols can be fruitfully interpreted, allowing an interesting knowledge discovery procedure, in lights with the new requirements of modern explainable AI systems. The effectiveness of the proposed algorithm has been proved through a set of experiments on paper abstracts classification and SMS spam detection.

1 INTRODUCTION

With the rapid evolution of Web-based Internet and mobile applications, an exponential growth of *unstructured* data in the form of web pages occurs. Thus, social network sites, microblog textual sources, IM sources, electronic papers and books have become very challenging problems. Among these, text mining - which means obtaining high-level information from natural language text excerpts - and text categorization (TC) - meaning assignment of a given text document to a class - can be considered. Text mining is by now a really wide research area at the intersection with Data Mining, Artificial Intelligence, Natural Language Processing (NLP), Information Retrieval (IR), Knowledge Management and Discovery, corpus-based Computational Linguistics and other Computer Science related disciplines (Feldman et al., 2007). Text categorization or text classification (TC) is a sub-discipline that exploits Machine Learning for solving a supervised learning problem, where documents in a corpus are provided together with well-suited labels, reflecting the meaning of the category or topic in which the document falls.

In the context of *information overload* (too much information, beyond the most relevant data) (Buckland, 2017), driven by Big Data applications, it is extremely useful having automatic procedures able to deal with a huge amount of unstructured text data, for summarization, knowledge discovery or classification purposes.

On the other hand, natural language can be considered as a communication system, defined and working at the interface between biology and social interactions (Kwapień and Drożdż, 2012). Natural language arises from information processing and integration performed by different brain areas, hence produced by a real-world complex system. Moreover, language has a hierarchical structure. A higher level of language organization is formed by clauses and

[a] https://orcid.org/0000-0002-6360-7737
[b] https://orcid.org/0000-0003-4915-0723
[c] https://orcid.org/0000-0002-3748-5019
[d] https://orcid.org/0000-0001-8244-0015

Capillo, A., de Santis, E., Mascioli, F. and Rizzi, A.
Mining M-Grams by a Granular Computing Approach for Text Classification.
DOI: 10.5220/0010109803500360
In *Proceedings of the 12th International Joint Conference on Computational Intelligence (IJCCI 2020)*, pages 350-360
ISBN: 978-989-758-475-6
Copyright © 2020 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

sentences which are the most important units of information transfer. In the case of written language, information levels are organized in a hierarchy of entities (characters, words, sentences, paragraphs, chapters, documents, concepts and so on) (Kwapień and Drożdż, 2012). This makes the problem of modeling and mining text data more challenging, even from the scientific point of view.

Heeman, from its own hand, some years ago pointed out how text documents possess a well-defined granular structure that can be divided in a high-level structure and a low-level one (Heeman, 1992). While the high-level structure relies on the organization of a given document, the low-level structure reflects the hierarchical structure of the language including characters, words, phrases, concepts, clauses, sentences, paragraphs (Jing and Lau, 2009). While words are the ground of semantic richness, groups of words or sentences can be considered, at a more advanced level of the hierarchy, as a basic unit dealing with concepts, that are notoriously entities richer than words.

The natural granular structure of text leads to find better representation and processing paradigms solving related downstream language modeling problems. Moreover, most of the techniques developed in Computational Linguistics try to grasp the meaning of text pieces with statistical measures on suitable text representations, hence finding regularities and recurrences within text data (Mikolov et al., 2013b). Granular Computing (GRC) dates back to 2008 (Apolloni et al., 2008) and provides a well-suited approach to represent text data through the notion of "information granules", that are complex information entities which arise in the process of data abstraction and knowledge formation. Information granules are collections of entities that originate at the numeric level and are arranged together due to their similarity, functional or physical adjacency, indistinguishability, coherency, etc. (Yao, 2006)(Martino et al., 2018). The granular processing paradigm allows to exploit the structure of a data source, providing a multilevel hierarchical view. Hence, this methodology leads to work to the right hierarchical level represented by the structure of the information granule (Martino et al., 2017). For example, in text data the *m*-gram decomposition can be seen as a text granulation procedure whose output are small pieces of text or phrases. In TC problems the GrC approach can help in finding regularities in text data, building a performing classification system able also to accomplish knowledge discovery tasks.

This paper deals with a TC system grounded on the GrC approach that provides an interesting and novel text embedding technique. The first step concerns the text corpus granulation, randomly extracting a suitable number of *m*-grams, i.e. information granules, of a given size. A per-class clustering procedure is then performed providing a proper hard partition of the space generated over a suitable numerical representation of the *m*-grams, obtained, in turn, trough a neural word embedding technique, performed on each token within the *m*-gram. Hence, the information granule is represented as a variable-length ordered word-vector, lying in a rich semantic space. In this work word-vectors are obtained through *fastText* (Bojanowski et al., 2016). The relatively new *fastText* model among the variety of word embedding techniques not based on the transformer architecture (Vaswani et al., 2017) is found to perform better on several language modeling tasks (Ritu et al., 2018).

As a further novelty, the adopted dissimilarity measure within the clustering procedure is a custom-based Edit distance, whose substitution cost is the Euclidean distance between word-vectors. The last dissimilarity metric allows measuring the dissimilarity even between *m*-grams – as a set of word-vectors – of different length. Once obtained the cluster model, the *m*-gram representatives are interpreted as symbols of a proper alphabet. The alphabet is used to estimate the statistical distribution of symbols in a given test document, through a proper membership rule. In other words, this approach leads to represent each document in a corpus as a Symbolic Histogram (SH) (Rizzi et al., 2012), a real-valued vector of fixed length useful for feeding standard Machine Learning algorithms. The text mining GrC system consists, finally, in an evolutionary optimization procedure in charge of both finding the (sub-)optimal hyper-parameters of the system and performing a wrapper-like features selection over symbols of the SH representation. The last optimization allows obtaining a simpler document representation that is computationally efficient and more interpretable. Specifically, a comparison between two features selection approaches is made by deciding at which step of the work-flow to binarize the features selections weights, since they are encoded in the GA as real-valued genes. Moreover, the obtained symbols, that are small phrases, can be used for interpreting the decision rule generated by the learning system in assigning a class label, opening to the knowledge discovery paradigm. Experiments are conducted on different datasets. The first dataset is a corpus of scientific paper abstracts for various topics pertaining some published famous journals in English language. The second one is a corpus of SMS adopted for spam detection. Classification performances together with the extracted concepts underlying the *m*-grams demonstrate the effectiveness of the GrC approach in text mining and classification.

This paper is organized as follows.

In section 2 is carried out a deeper explanation about the GrC TC system along with each work-flow step. The experimental setup, results and performance evaluations are reported and discussed in section 3. Finally, conclusions are drawn in section 4.

2 THE GRANULAR COMPUTING TEXT CATEGORIZATION SYSTEM

2.1 A Quick Overview

This work aims at synthesizing a learning algorithm for TC relying on NLP techniques, grounded on the GrC approach, within the context of Text Mining. A textual dataset $\mathcal{D} = \{t_i\}_{i=1}^n$ composed by n documents t_i is preprocessed, achieving a granular tokenization based on the GrC paradigm. The GrC model is generated by a per-class clustering process allowing to define a suitable robust *embedding* procedure representing documents in $\mathbb{R}^{\tilde{k}}$ through the SH approach. In other words, we seek a suitable mapping $\mathcal{M}$ starting from the document (unstructured) space $\mathcal{D}$ to real-valued $\tilde{k}$-tuples, that is $\mathcal{M} : \mathcal{D} \to \mathbb{R}^{n\times\tilde{k}}$.

The overall documents corpus $\mathcal{D}$ is partitioned in three disjoint sets, namely the training set $\mathcal{S}_{TR}$, the validation set $\mathcal{S}_{VAL}$ and the test set $\mathcal{S}_{TS}$.

Hence, during the learning phase document embedding vectors in $\mathcal{S}_{TR}$ can be fed to a standard classification algorithm, such as νSVM (Schölkopf et al., 2000), where both the selection of hyper-parameters and a wrapper feature selection procedure can be optimized by evolutionary meta-heuristics, driven by the classification performances obtained on the validation set $\mathcal{S}_{VAL}$ as fitness function. Feature selection has been chosen as a profitable procedure for the most relevant m-grams selection, both for improving classification performances and the knowledge discovery outcome. The global optimization capabilities of Evolutionary Algorithms are exploited through a wrapping approach, at a computational cost that is worth the final result. In particular, it is used a GA that encodes the cluster membership threshold adopted to build the embedding (of which we will provide further details), the SVM classifier hyper-parameters and the features selection weights as its genes. The work-flow is briefly synthesized as follows.

In line with the GrC paradigm, a set of information units called m-grams[1] is extracted from the dataset. Each m-gram η is an ordered m-tuple of words (short phrases), that is $\eta = [w_1, w_2, ..., w_m]$, whose maximum and minimum size is a meta-parameter set in advance. Let the complete per-class set of m-grams be χ_ω, where ω is the generic class label. It is worth to note that for a large document corpus the cardinality $|\chi_\omega|$ is combinatorial, thus a sub-sampling is performed through a uniformly random extraction procedure, where only a sub-set of cardinality $\vartheta \cdot |\chi_\omega|, \vartheta \in [0,1]$ is extracted and successively used.

These information granules create the knowledge base for the construction of the embedding that is the ground for the downstream classification procedure. Furthermore, the adoption of m-grams, as a set of words, together with the granular approach, allows building an interpretable model that behaves as a *gray-box* model.

The m-gram set is the information basin for the per-class k-medoids-based clustering process, which in general needs a suitable dissimilarity measure to group objects. To this aim, the Edit Distance (Navarro, 2001) is defined between two given sequences of words, not necessary of the same lenght. In this work, words are appropriately converted into real-valued vectors through a word embedding process, based on pre-trained Artificial Neural Networks (ANNs), eliciting a semantic relation between underlying word representations. It is worth to note that since the k-medoids clustering algorithm selects the most representative m-grams – medoids interpreted as symbols of a suitable alphabet Λ – pertaining each cluster among the m-grams set, the set of word-based medoids for the generic cluster is useful for performing knowledge discovery tasks beside the overall classification process.

Hence, the clustering procedure allows partitioning the m-gram representation space in k clusters $\mathcal{C}_s, s = 1, 2, ..., k$ (for each class $\omega \in \Omega$), so that each document $t \in \mathcal{D}$ can be represented by the SH, counting the occurrences in t of each symbol in the alphabet. Specifically, this procedure constitutes the embedding of an unstructured object – through the mapping $\mathcal{M}$. Therefore, a text document is transformed in a $\tilde{k}$-tuple of real numbers ready to be fed to the classification algorithm, in charge of solving a supervised learning problem.

A wrapper-like features selection procedure is then performed by a GA, where some genes represent feature selection weights and, according to their values, m-grams related to the corresponding symbols

[1]In the following we use m to denote the m-grams (instead of n) emphasizing the specific dimension herein adopted, specified in the experimental section.

in Λ are deleted following a suitable rule. Thence, the new (reduced) SH is passed to the νSVM classifier. The resulting classification performances on a validation set is considered as the GA fitness function, driving the evolution of GA population until a suitable stopping condition is reached. As a result, the TC procedure is optimized and, consequently, a suitable model of the text corpus is learned. In the following we will provide deeper details on the various building-blocks of the proposed system.

2.2 The Symbolic Histogram Construction

The SH encloses the information needed by the νSVM classifier as it represents the *m*-grams distribution for a given pattern (document). In other words, for a generic document, pertaining to a class $\omega \in \Omega$, the SH is an array of (normalized) counters $\mathbf{c} = [c_1, c_2, ..., c_k] \in \mathbb{R}^k$, each one related to an alphabet symbol $\lambda_s \in \Lambda^{\omega}, s = 1,2,...,k$, that measure the occurrence frequency of the *m*-grams pertaining a given generic document t. In this work we build an alphabet for each class, therefore the total alphabet $\bigcup_{\omega \in \Omega} \Lambda^{\omega}$ is of dimension $\tilde{k} = |\Lambda^{\omega}| \cdot |\Omega|$, where the first term in the second member is the cardinality of the alphabet for each class ω, while the second term indicates the number of available classes. In the following we omit the subscript ω if not strictly necessary.

The assignment is performed by a suitable method that relies on selecting the nearest over k representatives and checking if the *m*-gram vector representation falls within a threshold τ. The procedure is called "*minDist*" and it is specified in Algorithm 1 pseudocode.

We indicate a generic *m*-gram (numerical) representation for the i-th *m*-gram η_i as a suitable mapping Φ, whose nature will be specified further, such as $\Phi(\eta_i)$.

Firstly, the Edit Distance, – indicated by $edit(\cdot)$ in the following – between each *m*-gram representation $\Phi(\eta)_{t,i}$ from the generic document t and each symbol λ_s of the alphabet Λ is measured and arranged into the distance vector $\mathbf{d}_{\Phi(\eta)_{t,i}}$. Once the minimum value $d_{j,min} = \min(\mathbf{d}_{\phi(\eta)_{t,i}})$ is estimated, it is compared to the cluster threshold τ (the same for all clusters). If and only if d_j^{min} is lower than τ the value in the counter vector $\mathbf{c}_t$ of size k is incremented, at the index $j = \arg\min_j d_j^{min}$. In detail:

$$\forall \lambda_s \in \Lambda,\ s = 1,..,k \qquad \mathbf{d}_{\phi(\eta)_{t,i}} = edit(\lambda_s, \phi(\eta)_{t,i}),$$

$$\mathbf{c}_t(j) = \begin{cases} \mathbf{c}_t(j) + 1 & d_{j,min} < \tau \\ \mathbf{c}_t(j) & otherwise, \end{cases} \qquad (1)$$

where $\mathbf{c}_t(j)$ is the counter variable for the document t at position (index) j.

Thus, the *m*-grams pertaining to the document t that belong to each cluster are tracked by calculating their occurrence number, following the procedure reported in the Algorithm 1.

The procedure is performed for each document t of the dataset $\mathcal{D}$ so that a counter matrix $\mathbf{C}$ can be defined, where documents t are arranged as rows while alphabet symbols λ as columns, where each entry $c_{i,s}, i = 1,2,..,n,\ s = 1,2,...,k$ is the number of occurrences of symbol λ_s in a document t_i. Finally, the $\mathbf{C}$ matrix collects the SH $\mathbf{c}_{t_i}$ for each document providing an algebraic representation of the underlying unstructured text object. As concerns the normalization procedure, each column of the counter matrix $\mathbf{C}$ is divided by the maximum value reached on each column, that is: $\mathbf{c}_{i,s}^{norm} = \frac{\mathbf{c}_{i,s}}{\max(\mathbf{c}_{:,s})}, i = 1,2,..,n,\ s = 1,2,...,k$, where $\mathbf{c}_{:,s}$ is the s-th column vector of matrix $\mathbf{C}$.

Algorithm 1: The minDist algorithm.

```
Require: the dataset $\mathcal{D}$, the $m$-grams in each document $t$, the threshold $\tau$,
    the alphabet $\Lambda$
Ensure: the SH matrix $\mathbf{C}$
 1: procedure MINDIST(DATASET,$\Lambda$,$\tau$)
 2:     for $t$-document in dataset do
 3:         for $m$-gram $i$ in $t$ do
 4:             for $\lambda_s$ in $\Lambda$ do
 5:                 $\mathbf{d}_{\phi(\eta)_{t,i}} \leftarrow edit(\lambda_s, \phi(\eta)_{t,i})$
 6:             end for
 7:             $d_{j,min} \leftarrow \min(\mathbf{d}_{\phi(\eta)_{t,i}})$
 8:             if $d_{j,min} < \tau$ then
 9:                 $\mathbf{c}_t(j) \leftarrow \mathbf{c}_t(j) + 1$
10:             else
11:                 $\mathbf{c}_t(j) \leftarrow \mathbf{c}_t(j)$
12:             end if
13:         end for
14:     end for
15: end procedure
```

2.3 The M-Gram Representation

In the last decade, a series of new techniques grounded on shallow and deep ANN models have been developed, such as skip-gram with negative sampling (SGNS)(Goldberg and Levy, 2014; Mikolov et al., 2013a), known generally as the *word2vec* algorithm. These techniques rely on a suitable windowing procedure and the ANN is in charge of providing a vector-based representation of words while solving a suitable language task (i.e. guessing words around a target word in the window) in an unsupervised fashion. Some recent findings claim that the *word2vec* words representation derives from an implicit matrix factorization related to the HAL family representation

(Levy and Goldberg, 2014).

These by now popular word representation models ignore words morphology by associating a vector per-word. Therefore, some relevant semantic information could be ignored by learning algorithms, with a consequent lack of performance on downstream language modeling or classification tasks.

An alternative words representation model – adopted here – is *fastText* (Bojanowski et al., 2016). It is based on information units named *n*-grams[2] that are a set of sub-words. The procedure increases the effectiveness by incrementing data granularity. Thence, a single word becomes a set of *n*-tuples of characters. In the example below, the word "`where`" is represented by its *n*-grams of size 3, with two suffix and prefix extreme: `<wh, whe, her, ere, re>`.

Therefore, it is possible to define the dictionary $\mathcal{G}_w \in \{sub\_w_1, sub\_w_2, ..., sub\_w_g\}$ as the set of *n*-grams for the generic word *w*.

In this way, morphological information about the words are taken into account. With reference to what mentioned above, a learning algorithm could be more effective in grouping apparently very different words with the same root by computing the semantic information lead by the root *n*-gram. Moreover, a rare word could be linked to others on the base of similar *n*-grams. Data granularity, ruled by the size of the *n*-grams, is a crucial parameter to set, since the learning algorithm performance is strictly related to it. Thereby, the word representation consists in the numeric encoding of words assigning similar vectors to words with similar (distributional) meaning. As we will see, it can be adopted a pre-trained word embedding leading to transfer learning tasks or the word-vector representation can be learned on a given training set.

Hence, in this paper, a given *m*-gram $\eta^{(m)} = [w_1, w_2, ..., w_m]$ is extracted from $t \in \mathcal{D}$. Then, a mapping for each word w_i is obtained trough *fastText*, such as $\mathcal{F} : w_i \rightarrow \mathbb{R}^u$, where *u* is the size of the real-valued vector which encodes the single word (this is a parameter fixed at training time). Thus, the entire *m*-gram $\eta^{(m)}$ is a matrix of encoded words. In particular, it is mapped with an ordered set of apposed column vectors obtained by the mapping $\mathcal{F}$ acting on each constituent words, generating a matrix $\phi_\eta \in \mathbb{R}^{u \times m}$. The mapping is represented as $\Phi : [w_1, w_2, ..., w_m] \rightarrow \mathbb{R}^{u \times m}$, that is $\Phi(\eta^{(m)}) = \phi \in \mathbb{R}^{u \times m}$.

In this way, unlike the original approach, each *m*-gram is a sequence of words instead of some kind of averaged vector that can lead to a information loss.

[2]Here for *n*-grams we intend pieces of sub-words and not a group of contiguous words, such as for the approach adopted in the current work.

2.4 The Custom-based Edit Dissimilarity Measure

The clustering procedure adopted for the SHs construction needs a suitable dissimilarity measure as a kernel for partitioning an input dataset. In this work we adopt a suitable modified version of the well-known Edit distance.

According to the classic Edit dissimilarity measure, the minimum cost between two words is computed as a sequence of Edit operations, i.e. *substitutions*, *insertions* and *deletions*, needed to transform one word into the other. In this simple case, a word is a string and the object of manipulation are its characters. In this work, the information unit is not a single word (as a sequence of characters) but the *m*-gram as a sequence of vectors (each representing a single word) and different *m*-gram sizes are generally set to achieve a good granularity exploration.

The custom approach presented here is designed to achieve this goal: to compare *m*-grams of different sizes providing a dissimilarity measure for the clustering process. Through the *fastText* model each token in a *m*-gram is encoded in a real-valued vector whose dimension is *u*. Hence, it is possible to extend the Edit distance to the *m*-gram numerical representation given by $\phi_\eta \in \mathbb{R}^{u \times m}$ – see Sec. 2.3.

Given two *m*-gram representations ϕ_{η_1} and ϕ_{η_2} of length $m_1 = |\phi_{\eta_1}|_{col}$ and $m_2 = |\phi_{\eta_2}|_{col}$, the generic entry $\phi_\eta[i]$ is the *i*-th column vector of ϕ_η corresponding to a suitable word *w*.

Here, the custom substitution cost between the two given word-vectors within the *m*-grams representation, $\mathbf{x} = \phi_{\eta_1}[i_1]$ and $\mathbf{y} = \phi_{\eta_2}[i_2]$, belonging to different sequences, is the Euclidean distance $d^{cost}(\mathbf{x}, \mathbf{y})$, computed as follows:

$$d^{cost}(\mathbf{x}, \mathbf{y}) = \frac{\|\mathbf{x} - \mathbf{y}\|_2}{\sqrt{|u|}} = \frac{\sqrt{\sum_{i=1}^{u} (x_i - y_i)^2}}{\sqrt{|u|}}, \qquad (2)$$

where $\sqrt{|u|}$ is a normalization factor.

2.5 The Classification Algorithm

The SH representation of a given set of documents, both for $\mathcal{S}_{TR}$ and $\mathcal{S}_{VAL}$, consists of counter matrices $\mathbf{C} \in \mathbb{R}^{n \times \tilde{k}}$. The $\mathbf{C}$ matrices – conceived as *data matrices* in the classification contest – have documents as rows and (normalized) counters for symbols as columns – see Sec. 2.2. Once obtained the above described SH representation, the vector-based documents mapping can be classified through any standard learning algorithm able to process real-valued tuples.

Algorithm 2: The custom-based Edit dissimilarity measure algorithm.

Require: The two m-grams representations to compare ϕ_{η_1} and ϕ_{η_2}, the m-grams legths m_1 and m_2

Ensure: The Edit dissimilarity measure $\mathbf{d}$

```
1:  procedure CUSTEDITDIST($\phi_{\eta_1}$, $\phi_{\eta_2}$, $m_1$, $m_2$)
2:      for $i_1$ from 0 to $m_1$ do
3:          $\mathbf{d}[i_1, 0] := i_1$
4:      end for
5:      for $i_2$ from 0 to $m_2$ do
6:          $\mathbf{d}[0, i_2] := i_2$
7:      end for
8:      for $i_1$ from 1 to $m_1$ do
9:          for $i_2$ from 0 to $m_2$ do
10:             if $\phi_{\eta_1}[i_1 - 1] = \phi_{\eta_2}[i_2 - 1]$ then
11:                 $d \leftarrow 0$
12:                 $\mathbf{d}[i_1, i_2] \leftarrow \mathbf{d}[i_1 - 1, i_2 - 1]$
13:             else
14:                 $d \leftarrow d^{cost}$ (Euclidean distance)
15:                 $a \leftarrow \mathbf{d}[i_1 - 1, i_2]$
16:                 $b \leftarrow \mathbf{d}[i_1, i_2 - 1]$
17:                 $c \leftarrow \mathbf{d}[i_1 - 1, i_2 - 1]$
18:                 $\mathbf{d}[0, i_2] \leftarrow min(a, b, c)$
19:             end if
20:         end for
21:     end for
22:     return $\mathbf{d}[m_1, m_2]$
23: end procedure
```

In this work, we use a νSVM with RBF kernel adopting the LibSVM library (Chang and Lin, 2011).

The νSVM hyper-parameters $\nu \in [0,1]$ and $\gamma \in \mathbb{R}^+$ rule the amount of outlier patterns and the size of the RBF kernel, respectively. The former is related to the degree of classifier generalization (lower values could cause over-fitting) and the latter is defined as $K(\mathbf{x}, \mathbf{x}') = exp(-\gamma \cdot ||\mathbf{x} - \mathbf{x}'||^2)$ where $\mathbf{x}$ and $\mathbf{x}'$ are real-valued data patterns that belong to the input space, that in turn, foresees a different dimension compared to the arrival (kernel) space (that is infinite). The hyper-parameters ν and γ are encoded as genes of the wrapper GA as detailed in the next section.

2.6 Wrapper-based Features Selection and Hyper-parameters Optimization

The GA-Wrapper is in charge of achieving the optimal classification performances thanks to an evolutionary procedure, which catches the best values of the νSVM hyper-parameters (ν and γ), the threshold τ for the SH construction and the set of real-valued figures in the weight vector $\mathbf{v}$, which is used in the features selection procedure. The dimension of the weights vector is $dim(\mathbf{v}) = \tilde{k}$.

With the purpose of achieving an appropriate generalization level, the model is trained using the $\bar{k}$-fold cross-validation. The dataset is partitioned in $\bar{k}$ disjoint subsets, which, in turn, are equally divided in a training set $\mathcal{S}_{TR}$ and a validation set $\mathcal{S}_{VAL}$. The classifier hyper-parameters are separately optimized for each of the $\bar{k}$ subsets.

In other words, the clustering procedure and the consequent alphabet Λ definition are completed once for all, before the evolutionary optimization procedure. In line with the $\bar{k}$-fold cross-validation technique, the same generic GA individual is provided to all the $\bar{k}$-folds. For each $\bar{k}$-fold, after the features selection (described at the end of this subsection), a new classification model is defined and the related error rate $\varepsilon_f^{(\bar{k})}$ is evaluated on the $\mathcal{S}_{VAL}^{\bar{k}}$, that is $\varepsilon_f^{(\bar{k})} = 1 - \alpha_f^{(\bar{k})}$, where $\alpha_f^{(\bar{k})}$ is the accuracy obtained as:

$$\alpha = \frac{\text{number of correct predicted patterns}}{\text{total number of predicted patterns}}. \tag{3}$$

The subscript f stands for "*fold*".

The overall error rate ε for the classification task is the average of the $\varepsilon_f^{(\bar{k})}$ values, that is:

$$\varepsilon = \frac{1}{\bar{k}} \sum_{i=1}^{\bar{k}} \varepsilon_f^{(i)}. \tag{4}$$

The final objective function – to be minimized – for the overall optimization scheme is given by:

$$J = \beta \cdot \varepsilon + (1 - \beta) \cdot \frac{\sum v_r}{\tilde{k}}. \tag{5}$$

The last expression (5) is a convex linear combination of the total (average) error rate and a structural complexity measure, which accounts for the number of selected features (i.e., the number of alphabet symbols).

It is worth to note that, despite the increasing complexity of the learning procedure, the $\bar{k}$-fold cross-validation scheme leads to better generalization capabilities. The GA performs the evolutionary optimization until the stopping condition is reached, returning the best individual.

As concerns the feature selection, it involves the counter matrices $\mathbf{C}$ of both $\mathcal{S}_{TR}$ and $\mathcal{S}_{VAL}$. We remark that the $\mathbf{C}$ matrices have documents as rows and symbols as columns. According to the weights encoded in the GA as genes, the columns $\mathbf{c}_{:,j}$ of both the counter matrices are deleted if their indices corresponds to the indices of null weights in the weight vector $\mathbf{v} = [v_1, v_2, ..., v_{\tilde{k}}]$:

$$\mathbf{c}_{:,j} = \begin{cases} \mathbf{c}_{:,j} & v_j = 1 \\ 0 & v_j = 0. \end{cases} \tag{6}$$

It is worth to note that the weights in $\mathbf{v}$ are real-valued when they are processed by the GA and then they are binarized with the following operation:

$$v_j = \begin{cases} v_j & v_j \geq \theta \\ 0 & v_j < \theta, \end{cases} \tag{7}$$

where θ is a threshold computed as $\theta = \frac{1}{k}$, that is the reciprocal of the number of weights.

Such a choice about the genes encoding leads to better GA performances , thanks to an higher granularity of the genes values.

3 SIMULATION SETTINGS AND RESULTS

3.1 The Dataset

The first experimented dataset, namely the "Abstract" dataset, consists of a collection of scientific paper abstracts, published on famous journals in English. More precisely, it counts 460 abstracts distributed over the following four balanced classes (115 documents per-class): *Anatomy*, *Information Theory*, *String Theory* and *Semiconductors*. A further experiment is conducted adding documents for another class, namely *Smart Grids*. Tab. 1 reports the main statistics related to the corpus adopted for the experiments.

The second dataset consists of a corpus of 5574 SMS labeled as *spam* and *ham* with 81175 tokens. It is used for spam detection tasks. Further details can be found in (Almeida et al., 2011; Asuncion and Newman, 2007).

3.2 Experimental Setup

Before performing experiments, the text data has been pre-processed. As concerns the "Abstract" dataset, in this work, uppercase characters have been transformed in lowercase ones (*capitalization*) and stop words stored in a pre-defined list have been eliminated. Stop words are words without a semantic consistency, like the definite article `the` in the English language. The remaining words are then normalized, that is transformed in their simplest form (*normalization*). In the experiments with the "Abstract" dataset lemmatization is adopted. The last pre-processing phase is the *tokenization* step, where each document is mapped in a set of tokens (the *m*-grams) representing the document itself.

Instead, for the "SMS" dataset only the lowercasing pre-processing is performed, with no stop word elimination and retaining the punctuation, which is included in the tokens just like letters. The motivation resides in the fact that part of the discriminative information is contained in the punctuation and special characters.

The experiments have been conducted with some pre-defined conditions. Since clustering is performed once per-experiment, the granularity level (i.e. the number of clusters) k is fixed and set to 10. As concerns the m-grams extractor, the size $m = 1$ (unigrams), $m = 2$ (bi-grams) and $m = 3$ (tri-grams) are considered. The undersampling rate ϑ is fixed and set to 0.1. We remark that the number of the complete list of m-grams can be huge, thus the *undersampling rate* ϑ is used to reduce the computational burden, by considering only a relative percentage of the m-grams for each class. The numbers of m-grams per-class after the uniform random undersampling for the "Abstract" dataset are: 1284 for "Anatomy", 1775 for "Information Theory", 995 for "String Theory"and 1563 for "Semiconductor".

For the "Abstract" dataset, the *fastText* word embedding model is pre-trained on 1 million words, with an embedding dimension of $u = 300$. We introduced even a special parameter that allows considering only the dictionary words which have an occurrence frequency of at least equal to ψ, that, in the current experiments, has been set to 5. Instead, for the "SMS dataset", the word embedding is computed directly on the SMS corpus, specifically on the training set. This leads to better classification performances.

The parameter β in the objective function (5) is set to 0.8 giving more importance to the classification error than to the sparsity parameter. For the evolutionary optimization, it is used a kind of elitism that brings ahead the two fittest individuals. The dimension of population is set to 50 individuals and the fitness evaluation can be carried out in parallel fashion, on a many-core workstation.

The current investigation grounds firstly on the comparison between two evolutionary feature selection strategies, specifically on the "Abstract" dataset. In the first one, the weights related to the features selection are encoded as real-valued GA genes, which are binarized *before* each time the fitness function is evaluated. This online binarization (namely the Approach *A*) is performed for every single individual and for each generation. In the second one, the weights for features selection are yet encoded as real-valued GA genes, but they are binarized at the end of the optimization process, that is after the last generation. We call this procedure "post-processing binarization" (Approach *B*).

A second investigation is conducted with the aim

Table 1: "Abstract" dataset statistics: total number of words (*# words)*, minimum and maximum number of words per document (#minWords and #maxWords), average number of words per document μ_{words} and the variance of the number of words per document σ^2_{words}. The statistics concern the input dataset raw (before pre-processing), the dataset after pre-processing and after the word embedding procedure with *fastText*.

	# words	# minWords	# maxWords	μ_{words}	σ^2_{words}
Dataset Raw	52201	14	342	113.48	3229
Preprocessed Dataset	28151	10	200	61.19	965
fastText	26825	9	193	58.31	876

of measuring the generalization capability on the SMS spam detection task adopting what we found as the best feature selection strategy.

For robustness purposes, experimental results related to the current multi-class classification problem are averaged on 5 run of the optimization procedure with different seeds for the pseudo-random number generator.

Classification performances are generated through the evaluation of the confusion matrix obtained on the test set $\mathcal{S}_{TS}$. Specifically, the proposed figure of merits are the *Accuracy*, the *Precision*, the *Specificity*, the *Sensitivity*, the $F1$ *Score* and the *Informedness* computed as: $Informedness = Specificity + Sensitivity - 1$.

3.3 Experimental Results

Main results are reported in Tab 2 for the "Abstract" dataset without the *Smart Grids* class. The *Accuracy* average values for Approach A are greater than the ones for Approach B, even if it is not true for the variance values. Moreover, the same holds for all performance indicators. According to that, it can be stated that Approach A leads to a more effective solution than Approach B, although less robust. Nevertheless, even if figures in variance for Approach A are greater of one order of magnitude than the ones for Approach B (except of *Specificity*), they are reasonably and enough exiguous to assert their scarce relevance in the overall classification performance. Thus, Approach A performs better than Approach B.

A more accurate observation of the results can reveal some interesting considerations about the difference between the two approaches.

In Tab. 2 it can be seen that the difference between the *Precision* average value on one hand and the same figure for *Specificity* and *Sensitivity* on the other hand is greater in Approach B. Specifically, a larger gap between these two values is possible when the model classifies more True Negatives than True Positives. In addition, with reference to *Precision* and *Sensitivity*, a larger gap between these two values is possible when the model classifies more False Positives than False Negatives. What just said is true for both the approaches but figures hint that in following Approach B there is a greater gap between True Negatives and True Positives and a greater gap between False Positives and False Negatives, than for Approach A.

Similar consideration can be done for the experi-

Table 2: Measures of performances for the first "Abstract" dataset (without the *Smart Grids* class) obtained with the two genes binarization approaches. Results are averaged on 5 runs of the optimization procedure with different seeds for the pseudo-random number generator. Variance is reported in brackets.

Perf. Indicator	**Approach A**	**Approach B**
Accuracy	**0.9293** (0.0010)	0.9066 (0.0002)
Precision	0.9668 (0.0022)	0.9259 (0.0006)
Specificity	0.9884 (0.0003)	0.9739 (0.0001)
Sensitivity	0.9696 (0.0012)	0.9578 (0.0007)
$F1$ *Score*	0.9678 (0.0013)	0.9356 (0.0000)
Informedness	0.9580 (0.0021)	0.9217 (0.0003)

Table 3: Measures of performances for the second "Abstract" dataset (with the *Smart Grids* class) obtained with the two genes binarization approaches. Results are averaged on 5 runs of the optimization procedure with different seeds for the pseudo-random number generator. Variance is reported in brackets.

Perf. Indicator	**Approach A**	**Approach B**
Accuracy	**0.9130** (0.0009)	0.9061 (0.0003)
Precision	0.9660 (0.0011)	0.9750 (0.0013)
Specificity	0.9913 (0.0000)	0.9935 (0.0000)
Sensitivity	0.9569 (0.0028)	0.9478 (0.0022)
$F1$ *Score*	0.9621 (0.0005)	0.9661 (0.0006)
Informedness	0.9479 (0.0024)	0.9413 (0.0019)

Table 4: Alphabet symbols (a.k.a. representative *m*-grams) extracted from the best performing experiment (the first “Abstract” dataset). In blue the selected symbols by the feature selection procedure.

Anatomy	*Information Theory*	*String Theory*	*Semiconductor*
”first” ”case”	”first” ”type”	”first” ”time”	”effect” ”present” ”work”
”another”	”present” ”first” ”control”	”latter”	”well” ”current”
”first” ”case” ”many”	”another”	”theory” ”area” ”application”	”one”
”potential” ”treatment” ”patient”	”information” ”process” ”method”	”open” ”string” ”theory”	”various” ”type” ”semiconductor”
”point” ”patient”	”result” ”show” ”propose”	”field” ”theory” ”string”	semiconductor” ”effect”
”patient” ”type” ”2”	”management” ”system”	”string” ”theory”	”photoexcited” ”semiconductor” ”analysis”
”multivessel”	”method” ”information”	”show” ”string”	”semiconductor” ”device” ”aim”
”just” ”3” ”day”	”one” ”develop” ”model”	”string” ”field” ”theory”	”concept” ”present”
”disease” ”heart”	”analysis” ”method” ”information”	”perturbatively” ”up” ”second”	”metal” ”semiconductor”
”myocardial” ”infarction” ”revascularization”	”system” ”information” ”entropy”	”present” ”paper” ”field”	”semiconductor”

ments performed on the “Abstract” dataset increased with the *Smart Grids* class, whose results are reported in Tab. 3.

As concerns the knowledge discovery task, in Tab. 4 the selected symbols for each class for the best performing experiment (*Accuracy* 0.9293) with Approach A (“Abstract” dataset without the *Smart Grids* class) are reported . *m*-grams highlighted in blue are the ones selected during the feature selection phase. As concerns the classes *Semiconductor* and *Anatomy* all symbols are retained. Documents which belongs to *Anatomy* do not contain the word “anatomy” or its derived words. Maybe, as a consequence, the algorithm needs as much information as possible to classify documents in this class. On the other side, *Semiconductor* has many contextual words, roughly equally relevant for classification.

For the *Information Theory* class several symbols are discarded, such as `first type`, `result show purpose`, `management system`, while it is retained `system information entropy`, that is meaningful for its membership class. For the class *String Theory* the discarded symbols are generic *m*-grams, such as `present paper field`, `theory area application`, while they are retained, for example, `string field theory`. At least in this example, the system seems to prefer, among *m*-grams containing the tokens `theory` and `field`, the ones where these tokens are closed to the word `string`, that it is found meaningful for the underlying class.

In Tab. 5 are reported the performances obtained with the Approach A for the “SMS” dataset. The system reach good results in terms of *Accuracy* (0.9505) with an high *Precision* (0.9592) and $Informedness$ (0.7138). This demonstrates that the proposed approach is in charge of solving an instance of spam detection problem. But what is really interesting consists of the selected symbols, especially for the spam class – see Tab. 6. In fact, the *m*-gram selected by the evolutionary feature selection wrapper are words remarkably related to spam or adv., such as the unigram `subsciber`, or the bi-grams `customer care` or `only 10p`. This allow to conduct an in depth analysis of what is discriminative for each class even varying the granulation level, for example, setting more alphabet symbols in the SH synthesis.

Table 5: Performance measures for the “SMS” dataset obtained with the approach A. Results are averaged on 5 runs of the optimization procedure with different seeds for the pseudo-random number generator. Variance is reported in brackets.

$Acc.$	$Prec.$	$Spec.$	$Sens.$	$F1Score$	$Informed.$
0.9502	0.9592	0.7294	0.9843	0.9716	0.7138
(0.0000)	(0.0000)	(0.0014)	(0.0000)	(0.0000)	(0.0013)

Table 6: Alphabet symbols (a.k.a. representative *m*-grams) extracted from the best performing experiment (“SMS” dataset). In blue the selected symbols by the feature selection procedure.

ham	*spam*
”&” ”gt”	”subscriber”
”without”	”customer” ”care”
”city”	”nokia” ”plus”
”party” ”&”	”Ã¢” ”Â£”
”probably” ”take”	”the”
”great” ”personal”	”future” ”reply”
”made” ”fun”	”9.05E+09”
”.” ”makes”	”great”
”chennai”	”only” ”10p
”you” ”actually”	”spanish”

4 CONCLUSIONS

The current work represents the endeavor of building a TC system able to reach good recognition performances and at the same time able to provide an interpretable model, in which classification rules are even comprehensible for the final user with no experience in the Machine Learning field. The paradigm adopted is the one provided by GrC, where an appropriate granulation of the text in *m*-grams, along with a neural word embedding representation of the single tokens, allows to build a real-valued embedding of documents (the SH approach). This procedure yields a supervised learning problem to solve through a standard Machine Learning algorithm, likewise νSVM. The appealing of the current work is twofold. On one hand, there is the possibility of measuring the dissimilarity between a word-vector representation of *m*-grams of different lengths by means of a custom-based dissimilarity pertaining to the family of Edit distances. From the other, the entire processing pipeline allows building a *gray-box* model enabling the users to understand how the core classifier takes decisions, outputting a series of meaningful symbols, such as small sequences of words related to the class label. An evolutionary strategy along with the tuning of the classifier hyper-parameters is used for a wrapper-like feature selection, where feature weights (genes pertaining to the overall chromosome), originally casted as real-valued vectors, are binarized in two different ways, that is in an online and an off-line fashion. The first approach outperform clearly the second on the conducted experiments. The satisfying recognition performances together with the remarkable possibility to have additional information for knowledge discovery tasks, let us be confident in further developments of the described system. As concerns the GrC model, it is possible to adopt even an external text corpus (e.g. Wikipedia) eliciting a kind of focused transfer-learning procedure. The decision rule "minDist" experimented here can be substituted with other proper rules, making more robust the process of construction of the SH, hence improving the alphabet symbols synthesis. Finally, as concerns the dissimilarity measure between information granules (i.e. *m*-grams), other dissimilarity measures can be experimented, in order to provide a good semantic background to the system, such as the plain Euclidean distance between equal-sized *m*-grams or more general Edit distances such as the multidimensional Dynamic Time Warping. In the last case, longer *m*-grams can be used pushing the boundary towards more explainable AI systems.

REFERENCES

Almeida, T. A., Hidalgo, J. M. G., and Yamakami, A. (2011). Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262.

Apolloni, B., Bassis, S., Malchiodi, D., and Pedrycz, W. (2008). *The puzzle of granular computing*, volume 138. Springer.

Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Buckland, M. (2017). *Information and society*. MIT Press.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Feldman, R., Sanger, J., et al. (2007). *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press.

Goldberg, Y. and Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Heeman, F. C. (1992). Granularity in structured documents. *Electronic publishing*, 5(3):143–155.

Jing, L. and Lau, R. Y. (2009). Granular computing for text mining: New research challenges and opportunities. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 478–485. Springer.

Kwapień, J. and Drożdż, S. (2012). Physical approach to complex systems. *Physics Reports*, 515(3-4):115–226.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Martino, A., Giuliani, A., and Rizzi, A. (2018). Granular computing techniques for bioinformatics pattern recognition problems in non-metric spaces. In Pedrycz, W. and Chen, S.-M., editors, *Computational Intelligence for Pattern Recognition*, pages 53–81. Springer International Publishing, Cham.

Martino, A., Rizzi, A., and Frattale Mascioli, F. M. (2017). Efficient approaches for solving the large-scale k-medoids problem. In *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI,*, pages 338–347. INSTICC, SciTePress.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Ritu, Z. S., Nowshin, N., Nahid, M. M. H., and Ismail, S. (2018). Performance analysis of different word embedding models on bangla language. In *2018 Inter-*

national Conference on Bangla Speech and Language Processing (ICBSLP), pages 1–5. IEEE.

Rizzi, A., Del Vescovo, G., Livi, L., and Mascioli, F. (2012). A new granular computing approach for sequences representation and classification. In *Proceedings of the International Joint Conference on Neural Networks*. cited By 16.

Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. (2000). New support vector algorithms. *Neural computation*, 12(5):1207–1245.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yao, Y. (2006). Granular computing for data mining. In *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2006*, volume 6241, page 624105. International Society for Optics and Photonics.