*Learning and Nonlinear Models* - Journal of the Brazilian Society on Computational Intelligence (SBIC), Vol. 17, Iss. 2, pp. 27–41, 2019


# GA-TCTN: A FRAMEWORK FOR HYPER-PARAMETER OPTIMIZATION AND TEXT CLASSIFICATION USING TRANSDUCTIVE SEMI-SUPERVISED LEARNING THROUGH TERM NETWORKS

**Felipe Provezano Coutinho, Solange Oliveira Rezende**
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Universidade de São Paulo
fpcoutinho@usp.br, solange@icmc.usp.br

**Rafael Geraldeli Rossi**
Departamento de Ciências de Computação
Universidade Federal de Mato Grosso do Sul
rafael.g.rossi@ufms.br

**Abstract –** Transductive Classification through Term Network (TCTN) is an interesting and accurate approach to perform semi-supervised learning based on term networks for text classification. TCTN can surpass the accuracies obtained by transductive classification approach considering texts represented in other types of networks or vector space model. Also, TCTN can surpass the accuracies obtained by inductive supervised learning algorithms. Besides, the term networks in TCTN can have their size decreased while still keeps its classification performance. This implies a less computational cost than other semi-supervised learning approaches based on networks. Originally, TCTN considered just manually defined hyper-parameters. However, even better results can be achieved with a more carefully chosen hyper-parameters values. Thus, in this article, we present a genetic algorithm that (GA) can be used for finding better hyper-parameter values for TCTN. The proposed approach is called GA-TCTN. Our approach is applied in 25 text collections, and results demonstrate that a GA can be useful together with TCTN for semi-supervised text classification. Besides this contribution, comparisons among hyper-parameters distributions are performed to identify some pattern in its structure. The results indicate that TCTN and GA-TCTN tend to generate a similar set of hyper-parameters. However, GA-TCTN still allows the use of more specific hyper-parameters values being more flexible and practical than TCTN with manually defined parameters. Besides, GA-TCTN obtained better results than TCTN with statistically significant differences.

**Keywords –** Hyper-parameter Optimization, Genetic Algorithm, Transductive Classification, Semi-supervised Learning, Term Network.

## 1. INTRODUCTION

Text mining algorithms aim to extract useful patterns from unstructured text data [1]. These patterns can be used, for example, in e-mail filtering, document organization and retrieval, metadata generation, news categorization, and opinion mining, to cite few [1–3].

A common text mining task to perform the applications mentioned above is text classification [1]. Generally, text classification is carried out through the use of inductive supervised learning algorithms [2,4]. Supervised learning usually requires a significative number of labeled examples for each class to perform an accurate classification. Labeling examples is time-consuming and might impair the application of text classification in practical situations.

Transductive semi-supervised learning can perform text classification using some labeled examples and also unlabeled examples [5, 6]. Transductive learning classifies unlabeled examples directly without the necessity of inducing a classification model [7]. Even with few labeled examples, transductive learning achieves classification performances comparable to conventional supervised approaches with far fewer labeled examples [5, 8, 9]. Also, transductive learning is an essential component in inductive semi-supervised learning [10].

Usually, network-based approaches have been provided with better classification performances in transductive learning. In case of text classification, text can be represented by different networks such as document network [11], bipartite networks [9], or by networks composed by more types of objects and links [12].

Although most network representation for transductive learning relies on term frequency to build relations between documents and terms in bipartite or more complex networks, the term network approach proposed in [13] has the ability to represent useful relations among terms in a way to improve relevance scores of the terms and consequently improve the classification performance. This approach, named *Transductive Classification through Term Networks* (TCTN) provided results that surpass the classification

performance of other transductive learning algorithms based on networks (document and bipartite) or based on vector space model. Besides, the term networks in TCTN can have their size decreased while still keeps its classification performance.

On the other hand, TCTN has some hyper-parameters that needs to be tuned to produce better classification results. In [13], TCTN was carried out with manually defined parameters, which limits possibilities of maximizing TCTN's classification performance. Besides, this type of tuning requires a manually defined set of parameters, which can also impair obtaining better classification performances.

It is known that methods such as Grid-Search and Random-Search can be used for hyper-parameter optimization [14]. However, genetic algorithms keep some characteristics of Grid-Search as simplicity in implementation and parallelization, and introduces some level of randomness to try to obtain better solutions in the search space (Random-Search characteristics). Therefore, to improve the benefits of TCTN and also obtain better classification performances, in this article, a method based on Genetic Algorithm (GA) to tune TCTN parameters (GA-TCTN) is proposed. Also, our method allows searching in a larger search space, i.e., while in [13] one of hyper-parameters is $\alpha \in [0.1, 0.3, 0.5, 0.7, 0.9]$, in our method $\alpha \in [0, 1]$. Thus, GA-TCTN allows the use of more specific and non-trivial hyper-parameters values, being more flexible and practical than TCTN with manually defined parameters.

We applied GA-TCTN in 25 text collections. We also compared our proposal with other widely used methods for hyper-parameter optimization: Grid-Search and Random-Search. The results demonstrate that GA can be useful together with TCTN for transductive text classification, and surpass the results obtained by Grid-Search and Random-Search with statistically significant difference.

This remainder of this article is organized as follows. In Section 2 we presented related work and the main concepts related to this article: hyper-parameter optimization and the TCTN approach. In Section 3 our approach for hyper-parameter optimization of TCTN is presented. The results of our approach compared to Grid-Search and Random-Search is presented in Section 3.4 and Section 5 contains our conclusions and future work ideas.

## 2. Background & Related Work

In this section, we present the concepts and related work about hyper-parameter optimization. We also present the details of TCTN to better understand the functioning of the proposed approach.

### 2.1 Hyper-parameter Tuning

Hyper-parameter tuning is used when there are parameters values for learning algorithms, and the values of these parameters can affect the classification performance. Hyper-parameter tuning is usually performed manually or by automatically testing combinations of predefined values for each hyper-parameter. The last one is called Grid-Search [15]. These ideas have some advantages such as simplicity and reproducibility, but they flaw when the number of possible values for hyper-parameters is large or when dealing with real-valued parameters.

More sophisticated approaches treat hyper-parameter tuning as an optimization problem and use some optimization algorithm for solving it. Hyper-parameter optimization is useful since the optimal hyper-parameter values of the learning algorithms can vary for different datasets, and also when the parameter values and their combination are hard to define.

One of the most common hyper-parameter optimization technique for general learning algorithms is Random-Search [14]. Random-Search (RS) selects parameter value combinations to test randomly. Despite the randomness, Random-Search can find hyper-parameter values close to the best possible ones in far fewer iterations than Grid-Search, since the last one may spend too much time evaluating unpromising regions of the hyper-parameter search space. Also, Random-Search produces better results than Grid-Search [14, 16]. Thus, a combination of randomness with some criteria to guide the searching process seems to be a good idea for hyper-parameter tuning. Algorithms with these characteristics as Genetic Algorithms (GA) has been used for finding good hyper-parameters for machine learning algorithms.

GAs are commonly used as optimization algorithm and search problems since it can find solutions in promising search space regions. Also, it introduces randomness, as Random-Search, to expand the search to find better solutions. Genetic algorithms are used for different algorithms and areas in machine learning. For instance, in [17] and [18], genetic algorithms are proposed for feature selection and hyper-parameter optimization for Support Vector Machines and Support Tucker Machine, respectively. Likewise, a genetic algorithm is proposed by [19] to optimize real-valued parameters for Support Vector Machines and to demonstrate its efficiency on a particular dataset related to bankruptcy. Last but not least, a genetic algorithm can be applied for hyper-parameter optimization in Deep Learning [20], since manual model selection remains a tedious and highly intuition driven task.

### 2.2 Network Definition, Notations and Computational Structures for Transductive Semi-supervised Classification

A network can be defined by $N = \langle \mathcal{O}, \mathcal{R}, \mathcal{W} \rangle$, in which $\mathcal{O}$ is a set of objects of a network, $\mathcal{R}$ is the set of relations between objects, and $\mathcal{W}$ is the set of weights of the relations between objects. Network representations are a suitable choice to perform transductive semi-supervised learning, mainly through label propagation strategies [5, 6, 9, 21].

Regardless of the network representation, i.e., the type of the objects and relations, the computational structures to perform transductive classification are the same. Let $\mathcal{C} = \{c_1, c_2, \ldots, c_l\}$ be the set of label classes, and let $\mathbf{f}_{o_i} = \{f_{c_1}, f_{c_2}, \ldots, f_{c_{|\mathcal{C}|}}\}$ be

the relevance score (or class information) vector of an object $o_i$, which determines its weight or relevance score for each class. The $\mathbf{f}$ are usually analyzed to determine which class a network object belongs [5,22]. All class relevance score vectors are stored in a matrix $\mathbf{F}$.

Another important information is to have some labeled objects to propagate their labels. The predefined label of an object $o_i$, is stored in a vector $\mathbf{y}_{o_i} = \{y_{c_1}, y_{c_2}, \ldots, y_{c_{|\mathcal{C}|}}\}$. Usually. this vector contains the value 1 in the the corresponding class position and 0 to the others. Another important structure for label propagation is the weights of connections, which are stored in a matrix $\mathbf{W}$.

### 2.3 Transductive Classification through Term Networks

Transductive Classification through Term Networks (TCTN) is an interesting approach for text classification using terms (words) of those texts. According to [13], TCTN can be divided into four main steps: (1) term network generation, (2) initial relevance score setting, (3) relevance score propagation, and (4) text classification. In the next subsections, we present details of these steps.

#### 2.3.1 Term Network Generation

A viable way to perform transductive learning in networks is representing a text collection in a single network or few sub-networks that can model most of the existing relations in a dataset. Networks with these characteristics allow to perform label propagation, which is a linear and effective algorithm for semi-supervised learning based on networks [5, 6, 21].

To attempt those criteria, TCTN extract relations between terms considering the entire text collection. In this case, the network objects correspond to terms of a text collection, i.e., $\mathcal{O} = \mathcal{T}$, in which $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ for $m$ terms extracted after a preprocessing step. Relations ($\mathcal{R}$) are based on the similarity of terms, and the relation weights ($\mathcal{W}$) correspond to the similarity values. The goal is to use similarity function ($SF$) that will assign high weights to relations among specific terms and low weights to relations among generic terms. Similarity functions with those characteristics are derived from the contingency matrix presented in Table 1, in which:

- $p(t_i, t_j)$: probability of co-occurrence of two terms $t_i$ and $t_j$, i.e., the number of times $t_i$ and $t_j$ occurs together in the same documents, divided by the number of documents;
- $p(t_i, \neg t_j)$: probability of a term $t_i$ not co-occur with a term $t_j$;
- $p(\neg t_i, t_j)$: probability of a term $t_j$ not co-occur with a term $t_i$;
- $p(\neg t_i, \neg t_j)$: probability of terms $t_i$ and $t_j$ not co-occur together.

Table 1: Contingency Matrix

| | $t_j$ | $\neg t_j$ | Total |
|---|---|---|---|
| $t_i$ | $p(t_i, t_j)$ | $p(t_i, \neg t_j)$ | $p(t_i)$ |
| $\neg t_i$ | $p(\neg t_i, t_j)$ | $p(\neg t_i, \neg t_j)$ | $p(\neg t_i)$ |
| Total | $p(t_j)$ | $p(\neg t_j)$ | 1 |

The similarity functions considered in TCTN and based on the contingency matrix from Table 1 are presented in Table 2. Once computed the similarity between terms, two criteria are considered for deciding which relations will be made: (1) Threshold and (2) Top-$k$ approach. In the first, terms with similarity above a specific threshold are connected. In the second, terms are wired to its $k$ most similar terms.

Table 2: Similarity Functions considered in TCTN [23, 24]

| ID | Similarity Function | Formula | Range |
|---|---|---|---|
| 1 | *Piatetsky-Shapiro* | $p(t_i, t_j) - p(t_i)p(t_j)$ | [-0.25, 0.25] |
| 2 | *Mutual Information* | $p(t_i, t_j)\log_2(\frac{p(t_i,t_j)}{p(t_i)p(t_j)}) + p(t_i, \neg t_j)\log_2(\frac{p(t_i,\neg t_j)}{p(t_i)p(\neg t_j)}) + p(\neg t_i, t_j)\log_2(\frac{p(\neg t_i,t_j)}{p(\neg t_i)p(t_j)}) + p(\neg t_i, \neg t_j)\log_2(\frac{p(\neg t_i,\neg t_j)}{p(\neg t_i)p(\neg t_j)})$ | [-1, 1] |
| 3 | *Kappa* | $\frac{p(t_i,t_j)+p(\neg t_i,\neg t_j)-p(t_i)p(t_j)-p(\neg t_i)p(\neg t_j)}{1-p(t_i)p(t_j)-p(\neg t_i)p(\neg t_j)}$ | [-1, 1] |
| 4 | *Yule's Q* | $\frac{p(t_i,t_j)p(\neg t_i,\neg t_j)-p(\neg t_i,t_j)p(t_i,\neg t_j)}{p(t_i,t_j)p(\neg t_i,\neg t_j)+p(\neg t_i,t_j)p(t_i,\neg t_j)}$ | [-1, 1] |
| 5 | *Support* | $p(t_i, t_j)$ | [0, 1] |

In this article, we consider the Top-$k$ approach since networks with these characteristics have been obtained the best results in semi-supervised learning [21]. However, $k$ is an important hyper-parameter that has impact in the classification performance [5, 13].

Although five similarity functions are considered for term relation weight, only a subset of them produced good results for a specific text collection [13]. Therefore, similarity function is an important hyper-parameter that must be chosen carefully since it can impact the classification performance.

### 2.3.2 Initial Relevance Score Setting

Label propagation techniques require some labeled objects in the network, i.e., some objects have their initial relevance score ($\mathbf{y}$) defined. TCTN considers an approach to infer the initial relevance score based on labeled documents. Given a collection of documents $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, in which a usually small subset of $D$, denoted $D^L$ are composed of labeled documents, the initial relevance score of terms $t_i$ for a class $c_j$ is given by:

$$y_{t_i,c_j} = \sum_{d_k \in D^l} freq(d_k, t_i) y_{d_k,c_j} / \sum_{d_k \in D^l} freq(d_k, t_i), \tag{1}$$

in which $freq(d_k, t_i)$ is the frequency of term $t_i$ in document $d_k$, and $y_{d_k,c_j}$ is equal 1 if document $d_k$ belongs to class $c_j$ and 0 otherwise.

### 2.3.3 Relevance Score Propagation

Relevance score propagation in TCTN is performed by a mechanism similar to Learning with Local and Global Consistency (LLGC) [22]. Algorithm 1 presents the label propagation algorithm used to set the relevance score of all terms in a term network [13]. Line 1 computes the degree (sum of relation weights) of each term. The degree is used in Line 2 to calculate a normalized symmetric matrix, which is necessary for the convergence of relevance score propagation. Lines 3-5 performs relevance score propagation, in which the relevance score of each term is set by the relevance score of related terms (first term of Line 4) and their initial relevance score (second term of Line 4). The relevance score of related terms and their initial relevance score are weighted by $\alpha$ and $(1 - \alpha)$ respectively.

**Algorithm 1:** Relevance Score Propagation [13]

**Input :** $\mathcal{T}$,$\mathbf{W}$,$\mathbf{Y}$,
$\alpha$ - parameter to attenuate differences of the predefined relevance scores of network objects in consecutive iterations ($0 < \alpha < 1$)
**Output:** $\mathbf{F}$

1 $\mathbf{D} = diag(\mathbf{W} \cdot \mathbf{I}_{|\mathcal{T}|})$ `/* diag(...) is the diagonal matrix operator */`
2 $\mathbf{S} = \mathbf{D}^{-1/2} \cdot \mathbf{W} \cdot \mathbf{D}^{-1/2}$
3 **repeat**
4 $\quad \mathbf{F} \leftarrow \alpha \cdot \mathbf{S} \cdot \mathbf{F} - (1 - \alpha) \cdot \mathbf{Y}$
5 **until** *relevance scores of terms remains the same or fixed number of iterations*

At this point, a new hyper-parameter $\alpha$ is introduced. It can be any real value in [0, 1], and can also affect the classification performance [5]. Therefore, the decision in considering to optimize this hyper-parameter impose the optimization technique work with two types of values: integer and real. We highlight that the proposed approach correctly treats this imposition.

### 2.3.4 Text Classification

In this last step, information of relevance score of each term is used for document classification. The relevance score of an unlabeled document $d_i$ for a specific class $c_j$ is given by a weighted linear function taking into account the relation weights and the relevance scores of the terms, i.e.:

$$f_{d_i,c_j} = \sum_{t_k \in T} \text{freq}(d_i, t_k) \cdot f_{t_k,c_j}. \tag{2}$$

Then, the class or label of document $d_i$ is given by the $\arg\max$ value of $\mathbf{f}_{d_i}$, i.e., $class(d_i) = \arg\max_{c_j \in \mathcal{C}} f_{d_i,c_j}$.

Along with TCTN's steps, we cited some of its hyper-parameters as $k$, $\alpha$ and $SF$. As previously discussed, TCTN's hyper-parameters must be carefully chosen. Thus, the main idea of our approach is to use a Genetic Algorithm for hyper-parameter tuning. In the next section, our proposed approach is presented in detail.

## 3. PROPOSED APPROACH

The hyper-parameters we handle in this article are similarity function ($SF$), the number of nearest neighbors ($k$), and the trade-off between initial/given relevance scores and the relevance scores of related terms ($\alpha$). While $SF$ and $k$ can be threat as discrete values, $\alpha$ is continuous in $[0, 1]$. A summary of TCTN hyper-parameters and its characteristics are presented in Table 3. It is important to point out that all five similarity functions presented in Table 2 are considered. That is why the values of this hyper-parameter can range between 1 and 5.

According to [25], in order to define a particular GA, the following main components must be defined: (1) representation; (2) fitness function; (3) population and initialization procedure; (4) parent selection mechanism; (5) crossover and mutation; (6) survivor selection mechanism; and (7) termination condition. Therefore, in the next subsections, we present a diagram, details of each step of the proposed approach, and how the GA components are instantiated.

Table 3: TCTN hyper-parameters

| Hyper-parameter | Range Values | Type |
|---|---|---|
| $SF$ | $[1..5]$ | Discrete |
| $k$ | $[1..60]$ | Discrete |
| $\alpha$ | $[0, 1.0]$ | Continuous |

### 3.1 Diagram of GA-TCTN Process

Firstly, each individual or genotype is initialized with random gene values. The set of these individuals make up the initial population. Each individual is a set of hyper-parameters values that will serve as input of TCTN together with a specific text collection with labeled and unlabeled documents. The results of each TCTN execution are evaluated regarding some validation criteria for text classification. If the termination condition is satisfied, the best individual (set of hyper-parameters) is returned. Otherwise, individuals are selected, mutate and recombined and then inserted in a new population that will restart this cycle. Figure 1 presents a diagram of the use of GA to tune the parameters of TCTN.

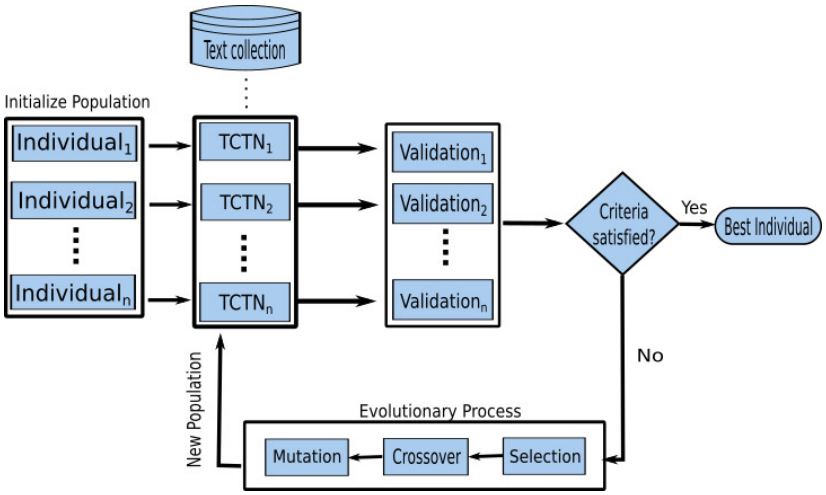



Figure 1: Diagram of the use of GA to tune TCTN's parameters

The execution of TCTN and validation steps are the bottleneck of this method. However, they can be performed in parallel once they are independent of each individual.

### 3.2 Representation, Population and Initialization Procedure

In GA-TCTN, individuals are vectors composed of three components (genes). The Figure 2 presents an illustration of an individual, in which the first component represents the similarity function ($SF$), the second component represents the number of nearest neighbors in Top-$k$ and lastly, $\alpha$ is a hyper-parameter for label propagation, and Figure 3 presents an example of an individual. It can be noted that $SF$ and $k$ are Natural values, while $\alpha$ is a Real value. Therefore, individuals comprise two types of values so that mutation operators must be different in order to deal with different types of values. Another important point is that $\alpha$ is not a value chosen from a finite set of values as in [13]. It can be any value in $[0, 1.0]$.

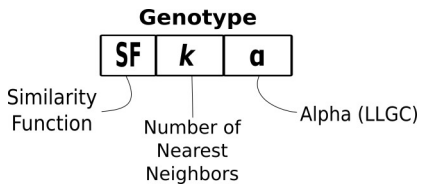



Figure 2: Definition of individuals

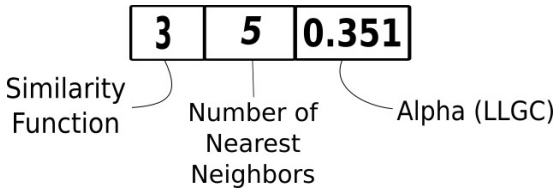



Figure 3: Individual example

### 3.3 Fitness Function

Fitness function is useful for evaluating each individual and provide a value for each one to make them comparable. In order to evaluate an individual, a separate set of examples, called validation set, is used to compute the weighted $F_1$ score[1]. From here

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

to the end, we will refer to weighted $F-1$ score just as $F_1$.

The weighted $F1$ score corresponds to a weighted average of the precision and recall, as presented in Equation 3. The values of precision and recall correspond to an average of precision and recall for each class. However, in weighted $F_1$ score, the precision and recall are weighted by the proportion of examples for each class. Equations 5, and 4 presented the weighted precision and recall computation for a class $c_i$ respectively. In both equation, $num(c_i)$ returns the number of examples of class $c_i$, TP, FN and FP stands for true positive, false negative, false positive, respectively [4].

$$\textit{Weighted } F_1 \textit{ score} = 2 \times \frac{\textit{Precision} \times \text{Recall}}{\textit{Precision} + \text{Recall}} \tag{3}$$

$$Recall_{c_i} = \frac{num(c_i)}{\sum_{c_j \in \mathcal{C}} num(c_j)} \times \frac{TP_{c_i}}{TP_{c_i} + FN_{c_i}} \tag{4}$$

$$Precision_{c_i} = \frac{num(c_i)}{\sum_{c_j \in \mathcal{C}} num(c_j)} \times \frac{TP_{c_i}}{TP_{c_i} + FP_{c_i}} \tag{5}$$

GA-TCTN aims to maximize the weighted $F_1$ score over generations. From here to the end, we will refer to the weighted $F_1$ score just as $F_1$.

### 3.4 Crossover, Mutation, Selection Mechanism, and Termination Condition

In GAs, a population of candidate solutions (individuals) is optimized over generations (iterations of GA) towards better solutions [25]. This can be achieved through the application of individual variation operators, such as mutation and crossover. In crossover, two or more individuals are selected from the population according to some strategy, to produce two or more individuals (children). A parameter known as crossover probability defines how often this operator will be applied among individuals. The mutation operator is applied to one individual with a certain probability (mutation probability). This operator is important since it induces the exploration of new search space locations. At each generation, a new population substitutes old population completely (Generational Algorithm), except when using strategies such as Elitism, in which $k$ best individuals of current population is kept in next generation. Finally, a termination condition must be defined. Several strategies as a predefined number of generations and a predefined number of generations without improvements in the best individual are found in literature [25].

As aforementioned, individuals are selected according to some selection strategy. We use a fast, easy, and most widely used method called Tournament Selection [25]. In this approach, $t$ (tournament size) individuals are selected randomly from the current population, and the best of them is returned as the winner. Therefore, we need to execute Tournament Selection several times until the number of selected individuals reaches our predefined population size. From this set of selected individuals, a recombination strategy called Uniform Crossover is applied between $individual_{i-1}$ and $individual_i$ with a probability called crossover probability ($cx_p$). In Uniform Crossover, genes of two parents are swapped with a predefined uniform crossover probability ($ucx_p$). For example, in Figure 4, two parents were selected ($Parent_1$ and $Parent_2$) and they are recombined to generate two children ($Child_1$ and $Child_2$). Each gene of $Parent_1$ and $Parent_2$ are swapped with probability $ucx_p$.

After application of Uniform Crossover, each resulting individual can suffer mutation with a mutation probability $m_p$. Specifically, the first two genes of selected individuals are mutated according to Uniform Mutation, while the last is mutated according to Gaussian Mutation. This is necessary because they differ in their type of value. The Uniform Crossover changes gene values of an individual with independent probability $indm_p$ to an integer uniformly drawn between a lower and upper bound value, inclusively. Similarly, Gaussian Mutation changes gene values of an individual with independent probability $indm_p$ to a real value drawn from a Gaussian distribution with parameters $\mu$ and $\sigma$. The mutation scheme is depicted in Figure 5. In Uniform Mutation, the value of a gene is randomly changed to a value in a specific range ($[1..5]$ for $SF$ and $[1..60]$ for $k$). Oh the other hand, in Gaussian Mutation, a new value is selected randomly according to a normal probability distribution. These three operators are applied until a predefined number of generations or a predefined number of generations without improvements in the best individual is achieved.

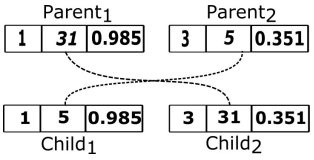



Figure 4: Crossover example in which $\text{Parent}_1$ and $\text{Parent}_2$ exchanges information to generate new individuals ($Child_1$, $Child_2$)

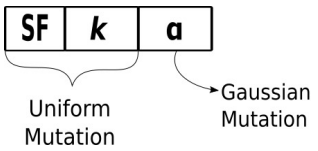



Figure 5: Uniform Mutation is applied in the first two genes ($SF$ and $k$), while Gaussian Mutation is applied in the last one (alpha)

## 4. Experimental Evaluation

GA-TCTN, Grid-Search, and Random-Search applied to TCTN were implemented in Python 3 with the aid of [26] framework for Evolutionary and Parallel Computing. All experiments were performed on a computer with GNU/Linux, 62GB of Ram

Memory and 12 processors. In next sections, we present the text collections used in the experimental evaluation, experiment configuration, evaluation criteria, results, and discussion.

### 4.1 Text Collections

Twenty-five real text collections were considered for performance evaluation of proposed method [27]. The collections have different characteristics and belong to different domains: medical documents (MD), news articles (NA), scientific documents (SD), sentiment analysis (SA), TREC (Text Retrieval Conference) documents (TD), and web pages (WP). Table 4 presents a summary of characteristics of each text collection. Columns are the name of text collection, number of documents ($|\mathcal{D}|$), number of terms ($|\mathcal{T}|$), average number of terms per document ($|\overline{\mathcal{T}}|$), number of classes ($|\mathcal{C}|$), standard deviation considering the class percentages in each collection ($\sigma(\mathcal{C})$), and the percentage of the majority class ($max(\mathcal{C})$). Also, we select 10% of top-ranked terms according to the sum of TF-IDF [28]. This speeds up classification task and, in this case, keeps quality of classification results [13].

Table 4: Summary of text collections characteristics used in experimental evaluation ($|\mathcal{D}|$ - number of documents; $|\mathcal{T}|$ - number of terms; $|\mathcal{C}|$ - number or classes; $\sigma(\mathcal{C})$ - standard deviation considering the class percentages; $max(\mathcal{C})$ - percentage of the majority class)

| **Collection** | **Domain** | $\lvert\mathcal{D}\rvert$ | $\lvert\mathcal{T}\rvert$ | $\lvert\overline{\mathcal{T}}\rvert$ | $\lvert\mathcal{C}\rvert$ | $\sigma(\mathcal{C})$ | $\max(\mathcal{C})$ |
|---|---|---|---|---|---|---|---|
| CSTR | SD | 299 | 1726 | 54.27 | 4 | 18.89 | 42.81 |
| Dmoz-Health-500 | WP | 6500 | 4217 | 12.40 | 13 | 0.00 | 7.69 |
| Dmoz-Science-500 | WP | 6000 | 4821 | 11.52 | 12 | 0.00 | 9.63 |
| Fbis | NA | 2463 | 2001 | 159.24 | 17 | 5.66 | 26.54 |
| Hitech | NA | 2301 | 12942 | 141.93 | 6 | 8.25 | 26.21 |
| IrishEconomicSentiment | SA | 1660 | 8659 | 112.65 | 3 | 6.83 | 39.46 |
| Oh0 | MD | 1003 | 3183 | 52.50 | 10 | 5.33 | 19.34 |
| Oh5 | MD | 918 | 3013 | 54.43 | 10 | 3.72 | 16.23 |
| Oh10 | MD | 1050 | 3239 | 55.64 | 10 | 4.25 | 15.71 |
| Oh15 | MD | 913 | 3101 | 59.30 | 10 | 4.27 | 17.20 |
| Opinosis | SA | 6457 | 2693 | 7.56 | 51 | 1.42 | 8.18 |
| Re0 | NA | 1504 | 2887 | 51.73 | 13 | 11.56 | 40.43 |
| Re1 | NA | 1657 | 3759 | 52.70 | 25 | 5.54 | 22.39 |
| Reuters-21578 | NA | 8723 | 14035 | 81.72 | 120 | 5.75 | 45.29 |
| Review Polarity | SA | 2000 | 15698 | 205.06 | 2 | 0.00 | 50.00 |
| SyskillWebert | WP | 334 | 4340 | 93.16 | 4 | 10.75 | 41.02 |
| Tr11 | TD | 414 | 6430 | 281.66 | 9 | 9.80 | 31.88 |
| Tr12 | TD | 313 | 5805 | 273.60 | 8 | 7.98 | 29.71 |
| Tr21 | TD | 336 | 7903 | 469.86 | 6 | 25.88 | 68.75 |
| Tr23 | TD | 204 | 5833 | 385.29 | 6 | 15.58 | 44.61 |
| Tr31 | TD | 927 | 10129 | 268.50 | 7 | 13.37 | 37.97 |
| Tr41 | TD | 878 | 7455 | 195.33 | 10 | 9.13 | 27.68 |
| Tr45 | TD | 690 | 8262 | 280.58 | 10 | 6.69 | 23.19 |
| WAP | WP | 1560 | 8461 | 141.33 | 20 | 5.20 | 21.86 |
| WebACE | WP | 3900 | 8881 | 43.15 | 21 | 8.44 | 35.74 |

### 4.2 Experimental Configuration and Evaluation Criteria

We compared GA-TCTN with traditional and state-of-the-art hyper-parameter tuning approaches: Grid-Search and Random-Search. The configurations of the approaches used in the experimental evaluation are:

- **Grid-Search**: the sets of hyper-parameters are $k = \{1, 7, 17, 37, 57\}$, $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $SF = \{$(1) Piatetsky Shapiro, (2) Mutual Information, (3) Kappa, (4) Yule's Q and (5) Support$\}$. Therefore, Grid-Search will test 125 combinations of hyper-parameters.
- **Random-Search**: we run Random-Search with a maximum number of 100 iterations. We considered as stopping criteria 5 iterations with no improved in the obtained solution.
- **GA-TCTN**: the configurations of the GA comprised population size = 50, maximum number of generations = 100, maximum number of generations without improvement = 5, crossover probability ($cx_p$) = 80%, uniform crossover probability

($ucx_p$) = 50%, mutation probability ($m_p$) = 10%, tournament size ($t$) = 3, elitism size = 1, $indm_p$ = 10, $\mu = 0$ and $\sigma = 0.2$. We highlight that other values for the parameters of GA could be considered but we chose those values in order to obtain faster solutions.

We generate three sets of documents to tune the hyper-parameters and evaluate the algorithms: labeled set, validation set, and unlabeled/test set. For the labeled set, we follow the procedure adopted in the original proposal of TCTN and carried out experiments considering 1, 10, 20, 30, 40, and 50 labeled documents per class. As validation set, 10% of documents were selected. We highlight the documents in validation set are considered as unlabeled during transductive learning. They are used after learning to measure the classification performance to tune the parameters. The remaining documents are used as unlabeled documents and also to evaluate the classification performance after obtained the best hyper-parameters.

The sets of documents presented above are generated 10 times randomly, and the classification performance corresponds to an average of the 10 executions. We used the weighted $F_1$ score (presented in Section 3.3) as classification performance measure. The $F1$ values of the algorithms were submitted to Friedman test and Nemenyi's post hoc test with 95% of confidence level to assess statistically significant differences [29].

We highlight that for both GA-TCTN and Random-TCTN, a more flexible set of values is allowed for $k$ and $\alpha$. In this case, we considered $k = [1..60]$, and $\alpha = [0, 1.0]$. Also, it is important to mention that 10 runs were carried out for GA-TCTN and Random-Search for each experimental configuration presented above. The results presented in the next section consists of the best solution obtained for both algorithms in those runs. We also will present the average result of GA-TCTN in the 10 runs.

### 4.3 Results

Tables 5, 6, 7, 8, 9, and 10 present the experiments results for text collections with 1, 10, 20, 30, 40 and 50 labeled documents, respectively, and each table is organized as follows: the first column is the text collection name, the second column is the best TCTN's $F_1$ score ($F_1$ score of best parameter set found via Grid-Search), the third column is the best GA-TCTN's $F_1$ score among 10 runs of GA-TCTN, the fourth is average of $F_1$ score over 10 runs of GA-TCTN followed by its standard deviation, and the last column is the margin of error of confidence interval of GA-TCTN average (fourth column) with 95% of confidence level. The best results for each text collection is highlighted in bold. It is important to note that not every text collection has the number of labeled documents per class specified in the experimental evaluation. In this case, we omitted the results of those text collections in the following tables.

Table 5: $F_1$ values obtained with 1 labeled document for each class

| Text collection | Grid-TCTN | Random-TCTN | GA-TCTN | GA-TCTN avg. | Margin of Error |
|---|---|---|---|---|---|
| CSTR | 0.4089 | 0.4072 | **0.4120** | 0.4089 (0.0011) | 0.0008 |
| Dmoz-Health | 0.2593 | 0.2603 | **0.2609** | 0.2604 (0.0004) | 0.0003 |
| Dmoz-Science | 0.1926 | 0.1927 | **0.1930** | 0.1930 (0.0000) | 0.0000 |
| Fbis | 0.2297 | 0.2297 | **0.2297** | 0.2297 (0.0000) | 0.0000 |
| Hitech | **0.2354** | **0.2354** | 0.2353 | 0.2352 (0.0000) | 0.0000 |
| IrishEconomicSentiment | 0.3210 | 0.3208 | **0.3210** | 0.3210 (0.0000) | 0.0000 |
| Oh0 | 0.4253 | 0.4263 | **0.4272** | 0.4270 (0.0003) | 0.0002 |
| Oh10 | 0.3760 | 0.3762 | **0.3772** | 0.3772 (0.0001) | 0.0001 |
| Oh15 | 0.3641 | 0.3621 | **0.3648** | 0.3645 (0.0004) | 0.0003 |
| Oh5 | 0.4208 | 0.4211 | **0.4228** | 0.4224 (0.0004) | 0.0003 |
| Opinosis | 0.2704 | 0.2716 | **0.2741** | 0.2740 (0.0001) | 0.0001 |
| Re0 | 0.2900 | 0.2915 | **0.2916** | 0.2916 (0.0000) | 0.0000 |
| Re1 | 0.3177 | 0.3183 | **0.3206** | 0.3203 (0.0003) | 0.0002 |
| Reuters-21578 | **0.1933** | 0.1917 | 0.1922 | 0.1921 (0.0001) | 0.0001 |
| Review Polarity | **0.4547** | 0.4501 | 0.4496 | 0.4496 (0.0000) | 0.0000 |
| SyskillWebert | 0.7056 | 0.7054 | **0.7068** | 0.7067 (0.0002) | 0.0002 |
| Tr11 | 0.3669 | 0.3668 | **0.3674** | 0.3673 (0.0001) | 0.0001 |
| Tr12 | 0.4262 | 0.4222 | **0.4288** | 0.4275 (0.0006) | 0.0005 |
| Tr21 | 0.4331 | 0.4329 | **0.4362** | 0.4362 (0.0000) | 0.0000 |
| Tr23 | 0.3824 | 0.3812 | **0.3827** | 0.3827 (0.0000) | 0.0000 |
| Tr31 | 0.5197 | 0.5155 | **0.5225** | 0.5224 (0.0003) | 0.0002 |
| Tr41 | 0.3810 | 0.3802 | **0.3864** | 0.3841 (0.0024) | 0.0017 |
| Tr45 | 0.4312 | 0.4355 | **0.4392** | 0.4392 (0.0001) | 0.0001 |
| WAP | 0.3327 | 0.3329 | **0.3342** | 0.3341 (0.0002) | 0.0002 |
| WebACE | 0.3402 | 0.3413 | **0.3511** | 0.3503 (0.0016) | 0.0008 |

Considering 1 labeled document per class (Table 5), GA-TCTN produces results that overpass Grid-TCTN and Random-TCTN for most of the collections. Even the average result of GA-TCTN obtained better results for most of the text collection. It is important to point out that GA-TCTN is consistent. This can be observed in standard deviation and confidence interval of $F_1$ average where usually presents values in the third or fourth decimal case. This is a desired feature, once it indicates GA-TCTN will produce similar results for new runs of GA.

Table 6: $F_1$ values obtained with 10 labeled documents for each class

| Text collection | Grid-TCTN | Random-TCTN | GA-TCTN | GA-TCTN avg. | Margin of Error |
|---|---|---|---|---|---|
| CSTR | 0.7445 | 0.7434 | **0.7461** | 0.7457 (0.0008) | 0.0005 |
| Dmoz-Health | 0.5633 | 0.5626 | **0.5652** | 0.5652 (0.0001) | 0.0000 |
| Dmoz-Science | 0.3951 | 0.3949 | **0.3961** | 0.3955 (0.0011) | 0.0008 |
| Fbis | 0.4402 | 0.4392 | **0.4403** | 0.4403 (0.0000) | 0.0000 |
| Hitech | 0.4763 | 0.4774 | **0.4787** | 0.4786 (0.0002) | 0.0001 |
| IrishEconomicSentiment | 0.4355 | 0.4343 | **0.4357** | 0.4356 (0.0001) | 0.0001 |
| Oh0 | 0.7489 | 0.7476 | **0.7496** | 0.7495 (0.0003) | 0.0002 |
| Oh10 | 0.6924 | 0.6923 | **0.6929** | 0.6929 (0.0000) | 0.0000 |
| Oh15 | 0.7013 | 0.7025 | **0.7026** | 0.7026 (0.0000) | 0.0000 |
| Oh5 | **0.7592** | 0.7546 | 0.7556 | 0.7556 (0.0000) | 0.0000 |
| Opinosis | 0.4905 | 0.4908 | **0.4910** | 0.4910 (0.0000) | 0.0000 |
| Re0 | 0.6125 | 0.6120 | **0.6151** | 0.6151 (0.0001) | 0.0001 |
| Re1 | 0.6256 | 0.6265 | **0.6321** | 0.6319 (0.0002) | 0.0002 |
| Review Polarity | **0.5647** | 0.5644 | 0.5622 | 0.5622 (0.0000) | 0.0000 |
| SyskillWebert | 0.7848 | 0.7866 | **0.7850** | 0.7850 (0.0001) | 0.0001 |
| Tr45 | 0.4110 | 0.4109 | **0.4112** | 0.4112 (0.0000) | 0.0000 |
| WebACE | 0.6718 | 0.6726 | **0.6741** | 0.6733 (0.0005) | 0.0002 |

The same pattern of the results obtained with 1 labeled document per class occurs for other amounts of labeled documents, i.e., GA-TCTN or even the average results of GA-TCTN got the best classification performances for most of the text collections. The exception occurs only when using 40 labeled examples per class, in which Grid-TCTN obtained the best results for 7 text collections while GA-TCTC obtained the best results for 5 text collections.

Table 7: $F_1$ values obtained with 20 labeled documents for each class

| Text collection | Grid-TCTN | Random-TCTN | GA-TCTN | GA-TCTN avg. | Margin of Error |
|---|---|---|---|---|---|
| CSTR | 0.7851 | 0.7860 | **0.7859** | 0.7859 (0.0000) | 0.0000 |
| Dmoz-Health | **0.6162** | 0.6153 | 0.6158 | 0.6156 (0.0001) | 0.0001 |
| Dmoz-Science | 0.4792 | 0.4767 | **0.4799** | 0.4797 (0.0002) | 0.0001 |
| Fbis | 0.4470 | 0.4470 | **0.4471** | 0.4471 (0.0000) | 0.0000 |
| Hitech | 0.5620 | 0.5626 | **0.5631** | 0.5631 (0.0000) | 0.0000 |
| IrishEconomicSentiment | **0.4712** | 0.4699 | **0.4712** | 0.4711 (0.0002) | 0.0001 |
| Oh0 | 0.7944 | 0.7953 | **0.7970** | 0.7970 (0.0000) | 0.0000 |
| Oh10 | 0.7571 | 0.7574 | **0.7578** | 0.7577 (0.0002) | 0.0001 |
| Oh15 | 0.7682 | 0.7696 | **0.7711** | 0.7704 (0.0007) | 0.0005 |
| Oh5 | **0.8103** | 0.8090 | 0.8093 | 0.8093 (0.0000) | 0.0000 |
| Opinosis | 0.5390 | 0.5389 | **0.5395** | 0.5395 (0.0000) | 0.0000 |
| Review Polarity | **0.5107** | 0.5107 | **0.5107** | 0.5107 (0.0000) | 0.0000 |
| SyskillWebert | **0.8314** | 0.8248 | 0.8265 | 0.8263 (0.0005) | 0.0004 |

Table 8: $F_1$ values obtained with 30 labeled documents for each class

| Text collection | Grid-TCTN | Random-TCTN | GA-TCTN | GA-TCTN avg. | Margin of Error |
|---|---|---|---|---|---|
| Dmoz-Health | 0.6611 | 0.6605 | **0.6614** | 0.6614 (0.0000) | 0.0000 |
| Dmoz-Science | 0.5305 | 0.5274 | **0.5307** | 0.5296 (0.0004) | 0.0003 |
| Fbis | 0.4908 | 0.4908 | **0.4909** | 0.4909 (0.0000) | 0.0000 |
| Hitech | 0.6189 | 0.6188 | **0.6195** | 0.6194 (0.0001) | 0.0001 |
| IrishEconomicSentiment | 0.5150 | 0.5162 | **0.5168** | 0.5166 (0.0003) | 0.0002 |
| Oh0 | 0.8193 | 0.8184 | **0.8198** | 0.8198 (0.0001) | 0.0001 |
| Oh10 | 0.7731 | 0.7730 | **0.7738** | 0.7736 (0.0002) | 0.0002 |
| Oh15 | 0.7927 | 0.7930 | **0.7936** | 0.7931 (0.0007) | 0.0005 |
| Oh5 | **0.8251** | 0.8229 | 0.8232 | 0.8232 (0.0000) | 0.0000 |
| Opinosis | 0.5677 | 0.5672 | **0.5682** | 0.5682 (0.0000) | 0.0000 |
| Review Polarity | 0.6137 | 0.6131 | **0.6144** | 0.6144 (0.0000) | 0.0000 |
| SyskillWebert | **0.8302** | 0.8220 | 0.8231 | 0.8231 (0.0000) | 0.0000 |

Table 9: $F_1$ values obtained with 40 labeled documents for each class

| Text collection | Grid-TCTN | Random-TCTN | GA-TCTN | GA-TCTN avg. | Margin of Error |
|---|---|---|---|---|---|
| Dmoz-Health | **0.6808** | 0.6806 | 0.6807 | 0.6807 (0.0000) | 0.0000 |
| Dmoz-Science | 0.5524 | 0.5500 | **0.5531** | 0.5530 (0.0002) | 0.0001 |
| Hitech | 0.6476 | 0.6469 | **0.6478** | 0.6478 (0.0001) | 0.0000 |
| IrishEconomicSentiment | **0.5066** | 0.5058 | 0.5058 | 0.5057 (0.0002) | 0.0001 |
| Oh0 | **0.8219** | 0.8235 | 0.8218 | 0.8218 (0.0000) | 0.0000 |
| Oh10 | **0.7912** | 0.7911 | 0.7908 | 0.7908 (0.0000) | 0.0000 |
| Oh15 | 0.8071 | 0.8070 | **0.8074** | 0.8074 (0.0000) | 0.0000 |
| Oh5 | **0.8367** | 0.8356 | **0.8367** | 0.8367 (0.0000) | 0.0000 |
| Opinosis | 0.5768 | 0.5762 | **0.5769** | 0.5769 (0.0000) | 0.0000 |
| Review Polarity | **0.5933** | 0.5784 | 0.5790 | 0.5788 (0.0002) | 0.0002 |
| SyskillWebert | **0.8647** | 0.8523 | 0.8629 | 0.8629 (0.0001) | 0.0001 |

The results of the different tuning approaches were also compared through a statistical significance test, as presented in Section 3.4. In Figure 6 we present the critical difference diagrams[2] (CDDs) to show the results of the statistical analysis.

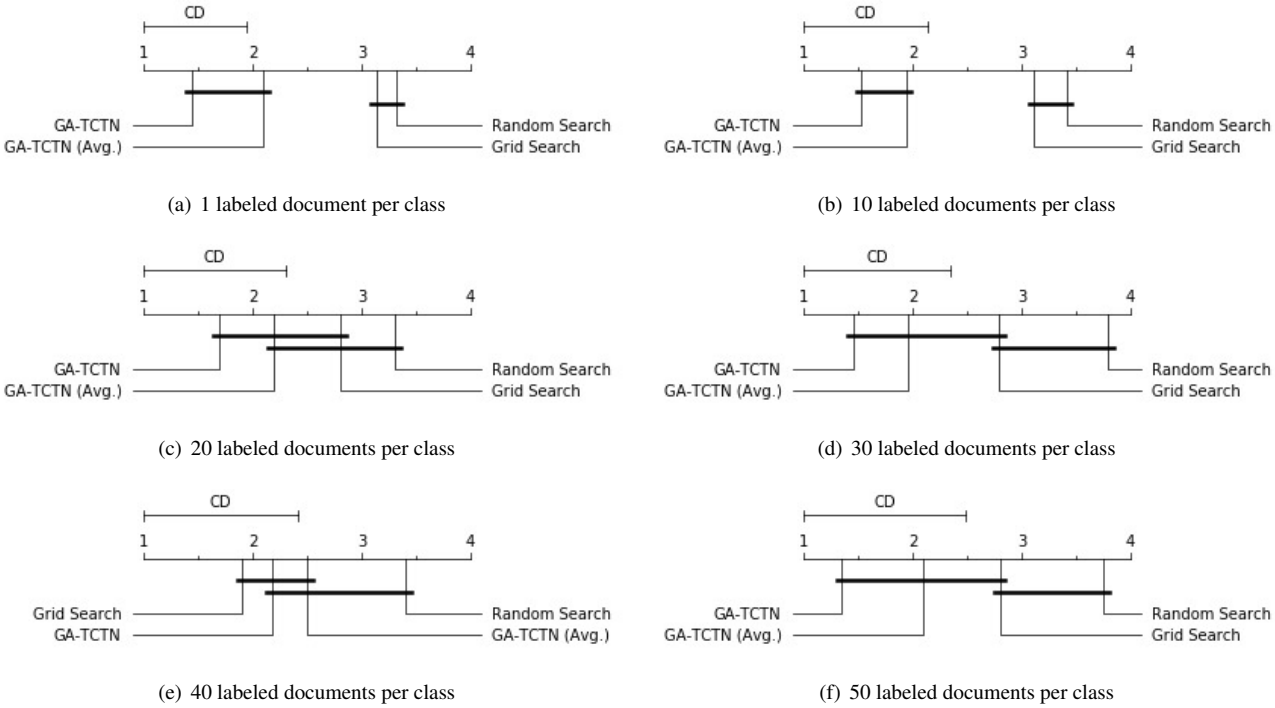
According to the CDDs, the GA-TCTN obtained the first position in the average ranking, except when used 40 labeled documents per class. Also, when considered 1 and 10 labeled documents per class, GA-TCTN obtained better results with statistically significant differences in comparison with Grid-Search and Random-Search. For 20 or more labeled documents per class, there were also statistically significant differences in comparison with Random-Search.

Other interesting results are the distributions of best results' hyper-parameters values according to the number of labeled documents. The distributions of hyper-parameters values ($alpha$, $k$ and $SF$) that provided the highest $F_1$ scores considering 1, 10, 20, 30, 40, and 50 labeled documents per class are presented, respectively, in Figures 7, 8,9,10, 11 and 12. We can notice some minor differences in the values of $\alpha$ and $k$ for the different search strategies, and for the different number of labeled documents. However, those minor differences represented a gain in $F_1$ score for GA-TCTN as previously presented. The major differences are related to the similarity functions. We can notice that GA-TCTN did not select the $SF = 1$ when using 10 or more labeled documents per class, and also $SF = 2$ when using 20 or more labeled documents per class while other search strategies did.

[2]In the critical difference diagrams, each method is sorted according to the average ranking, and the methods connected by a line do not present statistically significant differences among them.

Table 10: $F_1$ values obtained with 50 labeled documents for each class

| Text collection | Grid-TCTN | Random-TCTN | GA-TCTN | GA-TCTN avg. | Margin of Error |
|---|---|---|---|---|---|
| Dmoz-Health | 0.6900 | 0.6900 | **0.6902** | 0.6901 (0.0001) | 0.0001 |
| Dmoz-Science | **0.5785** | 0.5758 | 0.5782 | 0.5774 (0.0005) | 0.0003 |
| Hitech | 0.6759 | 0.6760 | **0.6768** | 0.6767 (0.0001) | 0.0000 |
| IrishEconomicSentiment | **0.5591** | 0.5575 | 0.5576 | 0.5575 (0.0002) | 0.0001 |
| Oh0 | 0.8242 | 0.8240 | **0.8256** | 0.8256 (0.0000) | 0.0000 |
| Oh10 | 0.8009 | 0.8009 | **0.8011** | 0.8011 (0.0000) | 0.0000 |
| Oh15 | 0.8098 | 0.8097 | **0.8106** | 0.8101 (0.0004) | 0.0003 |
| Oh5 | 0.8440 | 0.8413 | **0.8441** | 0.8441 (0.0000) | 0.0000 |
| Review Polarity | 0.6117 | 0.5939 | **0.6127** | 0.6123 (0.0010) | 0.0007 |
| SyskillWebert | 0.8762 | 0.8689 | **0.8772** | 0.8770 (0.0001) | 0.0001 |




(a) 1 labeled document per class

(b) 10 labeled documents per class

(c) 20 labeled documents per class

(d) 30 labeled documents per class

(e) 40 labeled documents per class

(f) 50 labeled documents per class

Figure 6: Critical difference diagrams

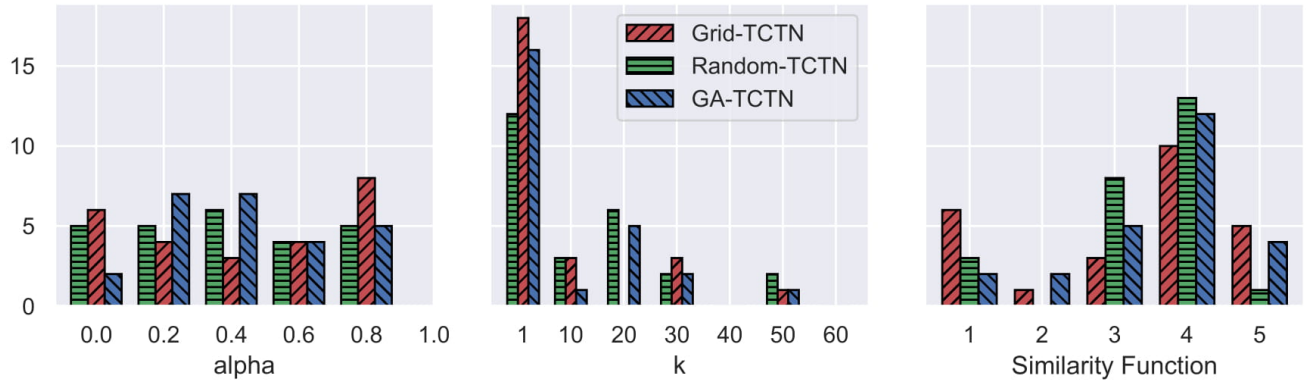



Figure 7: Distribution of hyper-parameters values for the different tuning approaches with 1 labeled document per class

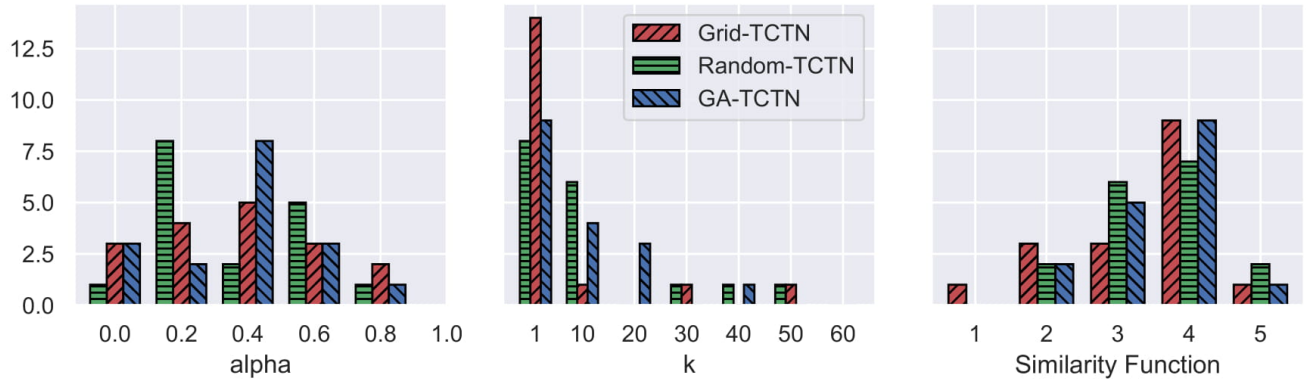



Figure 8: Distribution of hyper-parameters values for the different tuning approaches with 10 labeled documents per class

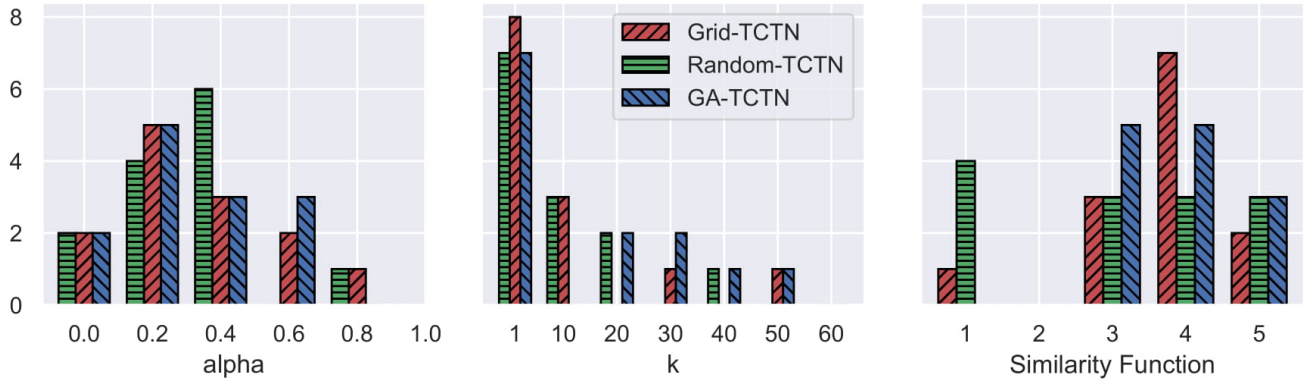



Figure 9: Distribution of hyper-parameters values for the different tuning approaches with 20 labeled documents per class

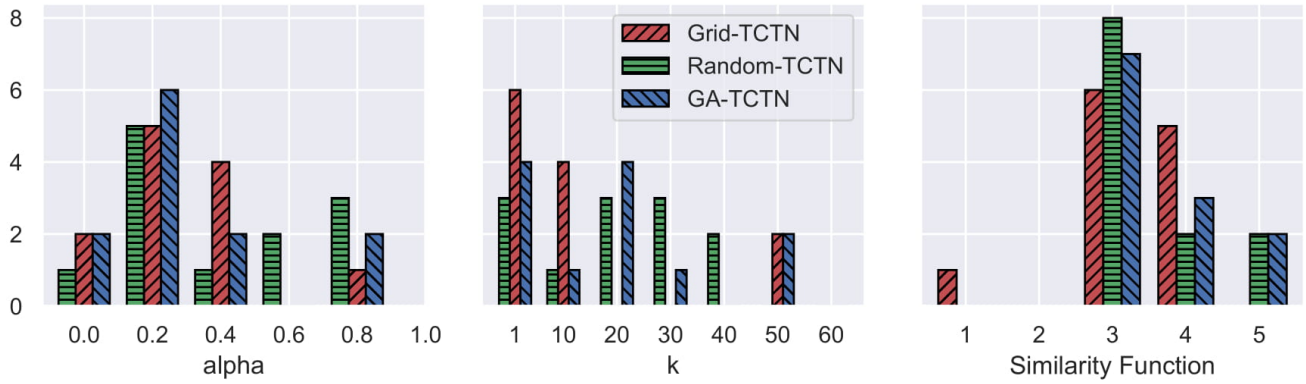



Figure 10: Distribution of hyper-parameters values for the different tuning approaches with 30 labeled documents per class

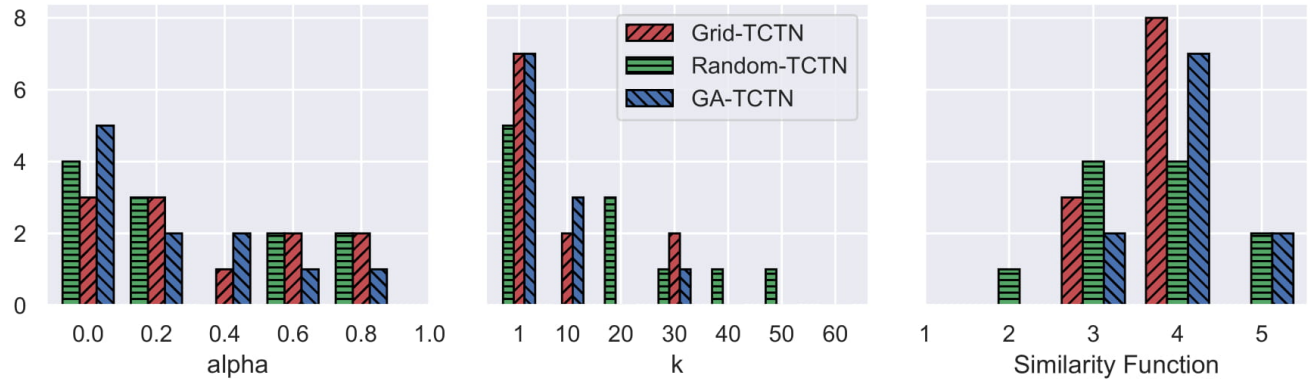



Figure 11: Distribution of hyper-parameters values for the different tuning approaches with 40 labeled documents per class

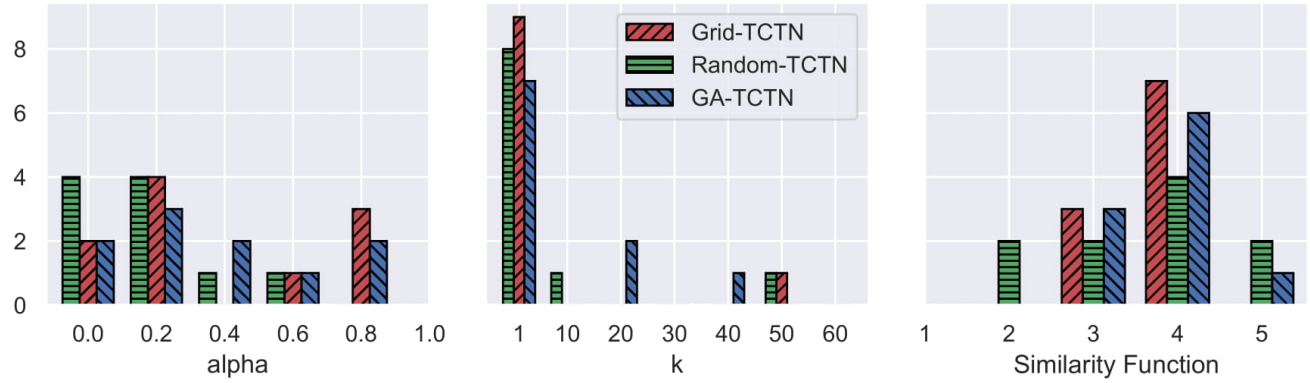



Figure 12: Distribution of hyper-parameters values for the different tuning approaches with 50 labeled documents per class

Lastly, in Table 11, we present examples of the best results of GA-TCTN to show its capability to obtain non-trivial sets of hyper-parameters values. We can note that the same parameters would hardly be chosen manually or in a Grid-Search, such as $k = 44$ or $\alpha = 0.0805$. Besides, through Table 11 is clear the variability of the different parameter values that generated the best results.

Table 11: Table 11. Examples of non-trivial hyper-parameters obtained by GA-TCTN

| **Parameters** | **1 Lab. Doc.** | **10 Lab. Docs.** | **20 Lab. Docs.** | **30 Lab. Docs.** | **40 Lab. Docs.** | **50 Lab. Docs.** |
|---|---|---|---|---|---|---|
| Collection | WAP | Review Polarity | Oh5 | IrishSent. | Dmoz-Health-500 | Oh10 |
| $SF$ | Mut. Inf. | Support | Yule's Q | Yule's Q | Support | Yule's Q |
| $k$ | 2 | 1 | 44 | 26 | 8 | 21 |
| $\alpha$ | 0.5955 | 0.2693 | 0.0805 | 0.8545 | 0.0073 | 0.0103 |

## 5. CONCLUSIONS AND FUTURE WORK

A big issue in most text mining algorithms is hyper-parameters setting. They must be carefully chosen since they can influence classification results. Especially when using an algorithm based on networks, the parameters to build a network have also impact on the classification performance.

Usually, parameters are tuned manually or by Grid-Search technique. However, these approaches have a limited search space. In this article, we demonstrate through experimental evaluation that TCTN's hyper-parameters can be tuned using a Genetic Algorithm. Also, we presented, implemented, and evaluated a framework for TCTN's hyper-parameters tuning that overpass Grid-Search and Random-Search in most text collections with different amounts of labeled documents. The results

indicate that the use of a GA for hyper-parameter tuning can be useful in finding better or even non-trivial hyper-parameters for TCTN, since the proposed approach (GA-TCTN) allows for a more expansive exploration of the search space.

GA-TCTN is simple and can be parallelized to speed up the hyper-parameter tuning process. Besides, GA-TCTN mixes randomness and guided search, which is a good characteristic in Random-Search, and at the same time disregard unpromising parameters explored by Grid-Search [14]. Also, the results obtained by the proposed approach are consistent since standard deviations and interval confidence of average regarding 10 runs of GA-TCTN always presented very small values.

As future work, more options of networks and relevance score propagation parameter will be considered. Other parameters of GA also will be tested to provide a trade-off between computational cost and classification performance.

## Acknowledgement

This work was supported by Coordination of Superior Level Staff Improvement (CAPES), finance code 001, São Paulo Research Foundation (FAPESP), grants #2016/17078-0 and #433082/2018-6, and Brazilian National Research Council (CNPq), process #433082/2018-6.

## BIBLIOGRAPHY

[1] C. Aggarwal. *Machine Learning for Text*. Springer Int. Publishing, 2018.

[2] S. Weiss, N. Indurkhya and T. Zhang. *Fundamentals of Predictive Text Mining*. Texts in Computer Science. Springer London, 2015.

[3] C. Manning, P. Raghavan and H. Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008.

[4] F. Sebastiani. "Machine Learning in Automated Text Categorization". *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[5] R. G. Rossi, A. A. Lopes and S. O. Rezende. "Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts". *Information Processing & Management*, vol. 52, no. 2, pp. 217–257, 2016.

[6] X. Zhu and A. B. Goldberg. *Introduction to Semi-Supervised Learning*. Morgan and Claypool Publishers, 2009.

[7] A. Gammerman, V. Vovk and V. Vapnik. "Learning by transduction". In *Proc. the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 148–155. Morgan Kaufmann Publishers Inc., 1998.

[8] F. A. Breve, L. Zhao, M. G. Quiles, W. Pedrycz and J. Liu. "Particle Competition and Cooperation in Networks for Semi-Supervised Learning". *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 9, pp. 1686–1698, 2012.

[9] R. G. Rossi, A. A. Lopes and S. O. Rezende. "A Parameter-Free Label Propagation Algorithm Using Bipartite Heterogeneous Networks for Text Classification". In *Proc. Symposium on Applied Computing*, pp. 79–84. ACM, 2014.

[10] R. G. Rossi, A. de Andrade Lopes and S. O. Rezende. "Using bipartite heterogeneous networks to speed up inductive semi-supervised learning and improve automatic text categorization". *Knowledge-Based Systems*, vol. 132, pp. 94–118, 2017.

[11] R. Angelova and G. Weikum. "Graph-based text classification: learn from your neighbors". In *Proc. the Special Interest Group on Information Retrieval Conference*, pp. 485–492. ACM, 2006.

[12] M. Ji, Y. Sun, M. Danilevsky, J. Han and J. Gao. "Graph regularized transductive classification on heterogeneous information networks". In *Proc. European Conf. Machine Learning and Knowledge Discovery in Databases*, pp. 570–586. Springer-Verlag, 2010.

[13] R. G. Rossi, S. O. Rezende and A. de Andrade Lopes. "Term network approach for transductive classification". In *Int. Conference on Intelligent Text Processing and Computational Linguistics*, pp. 497–515. Springer, 2015.

[14] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization". *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[15] R. Liu, E. Liu, J. Yang, M. Li and F. Wang. "Optimizing the hyper-parameters for SVM by combining evolution strategies with a grid search". In *Intelligent Control and Automation*, pp. 712–721. Springer, 2006.

[16] J. S. Bergstra, R. Bardenet, Y. Bengio and B. Kégl. "Algorithms for Hyper-Parameter Optimization". In *Advances in Neural Information Processing Systems 24*, edited by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira and K. Q. Weinberger, pp. 2546–2554. Curran Associates, Inc., 2011.

[17] C.-L. Huang and C.-J. Wang. "A GA-based feature selection and parameters optimizationfor support vector machines". *Expert Systems with applications*, vol. 31, no. 2, pp. 231–240, 2006.

[18] D. Zeng, S. Wang, Y. Shen and C. Shi. "A GA-based feature selection and parameter optimization for support tucker machine". *Procedia computer science*, vol. 111, pp. 17–23, 2017.

[19] C.-H. Wu, G.-H. Tzeng, Y.-J. Goo and W.-C. Fang. "A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy". *Expert systems with applications*, vol. 32, no. 2, pp. 397–408, 2007.

[20] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim and R. M. Patton. "Optimizing deep learning hyper-parameters through an evolutionary algorithm". In *Proc. the Workshop on Machine Learning in High-Performance Computing Environments*, p. 4. ACM, 2015.

[21] C. A. R. de Sousa, S. O. Rezende and G. E. A. P. A. Batista. "Influence of Graph Construction on Semi-supervised Learning". In *Proc. European Confer. Machine Learning and Knowledge Discovery in Databases*, pp. 160–175, 2013.

[22] D. Zhou, O. Bousquet, T. N. Lal, J. Weston and B. Schölkopf. "Learning with local and global consistency". In *Advances in neural information processing systems*, pp. 321–328, 2004.

[23] P.-N. Tan, V. Kumar and J. Srivastava. "Selecting the right interestingness measure for association patterns". In *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 32–41. ACM, 2002.

[24] L. Geng and H. Hamilton. "Interestingness measures for data mining". *ACM Computing Surveys*, vol. 38, no. 3, 2006.

[25] A. E. Eiben, J. E. Smith *et al.*. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[26] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau and C. Gagné. "DEAP: Evolutionary algorithms made easy". *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[27] R. G. Rossi, R. M. Marcacini and S. O. Rezende. "Benchmarking text collections for classification and clustering tasks". *Institute of Mathematics and Computer Sciences, University of Sao Paulo*, , no. 395, 2013.

[28] G. Salton. "Automatic text processing: The transformation, analysis, and retrieval of". *Reading: Addison-Wesley*, 1989.

[29] J. Demsar. "Statistical Comparisons of Classifiers over Multiple Data Sets". *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.