Submitted to Knowledge and Information Systems (KAIS)

Inductive Model for Text Classification Using Generalized Relations in Heterogeneous Network

Rafael Geraldeli Rossi[1], Alneu de Andrade Lopes[2], Solange Oliveira Rezende[2]

[1]Federal Uiversity of Mato Grosso do Sul, Três Lagoas, Brasil
[2]Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil

Abstract. Text automatic classification is an important task to manage and extracting knowledge from the plenty of textual data published nowadays. Traditional state-of-the-art supervised inductive learning algorithms for text classification consider that the textual data are represented in a vector space model, in which each text corresponds to a vector, and each dimension corresponds to a text feature. Usually, the features correspond to single terms. In this case, only relations between documents and terms are represented. However, different types of relations carry different information about data, allowing to improve classification performances through a better representation of class patterns. Nevertheless, vector space model approaches that represent relations besides document and terms, such as between documents or between terms, can cause a huge increase in the representation dimensionality, a decrease in the classification performance, or both. On the other hand, heterogeneous networks can be used as a useful alternative since it can represent different types of relations among text entities in an efficient way. Despite the benefits, only heterogeneous networks relations of a single type has been successfully proposed in the literature. Therefore, in this article, we present an algorithm which generates a classification model through the induction of the relevance scores of terms for classes considering two types of relations: (i) relations between documents and terms, and (ii) relations between terms. We called those relations as generalized relation since they can be extracted from any text collection. The results show that the proposed algorithm outperforms state-of-the-art algorithms based on the vector space model or algorithms base on networks composed by relations of a single type with statistically significant differences.

Keywords: text classification; heterogeneous networks; supervised inductive learning; term similarity

1. Introduction

An increasing amount of textual data has been published and stored nowadays. Manual analysis, organization, management, and knowledge extraction of such textual data are impractical without the aid of automatic methods. Text automatic classification (TAC) can be used to perform these tasks automatically and thus has gained importance in the last decades [3, 122, 35, 104].

TAC consists of assigning predefined categories to textual documents. Usually, this is performed using supervised learning techniques to build a classification model considering the labeled texts of a collection. Then, the classification model is used to classify new/unlabeled texts [104].

The main TAC algorithms consider that the text collection is represented in a vector space model (VSM). i.e., each document is represented by a vector, and each dimension corresponds to a feature of the text collection [70]. Usually, the features correspond to single terms, generating the well known bag-of-words [101]. In this case, the bag-of-words only represents information about the relations between documents and terms (called here `document-term` relations). Therefore, other relations besides `document-term` that can be helpful to discriminate texts from different classes are disregarded in bag-of-words. On the other hand, approaches to represent relations among entities of a text collection in VSM, such as relations between documents or relations between terms, can cause a huge increase in the representation dimensionality, a decrease in the classification performance, or even both [76, 6, 57, 18]. On the other hand, disregarding such relations can decrease the classification performance [92, 51, 67].

Networks allow representing relations among entities in an efficient way, treating the leaning problem in a mathematical formalism, and have come up as an alternative to represent text collections [98, 93, 14, 73, 32]. Also, network representations allow extracting patterns which are not extracted by algorithms based on VSM [17, 41]. Additionally, heterogeneous networks can represent different types of relations among text entities, which, according to some author, is crucial to improve classification performance since different relations carry different semantic information [113, 68, 106].

Despite the benefits, most of the network-based algorithms perform transductive learning, and just a few of them perform supervised inductive learning. Most of the supervised learning algorithms adopt two major approaches. The first approach consists in generating networks for each labeled document and performs graph matching to classify new documents [75, 102], which presents a high classification time and high memory consumption due to the need to keep all the term networks in memory. The second approach consists in extracting features from networks to generate a VSM representation [23, 8, 4, 1]. This type of approach also presents a high computation cost if frequent subnetworks extraction are needed to generate the feature [23, 1], generates high dimension representations [4], and might not effectively improve classification performance [8]. Besides, this latter approach has an additional computational cost due to the VSM generation after the network generation.

Besides the previous consideration about the the use of networks for text classification, several network proposals are domain dependent, i.e., the network generation requires specific information about the texts of a domain, e.g., web page tags [23, 102, 1] or controlled vocabulary with semantic information [75]. Just few approaches consider relations that can be extracted for any text collection domain such as: (i) relations between documents based on similarity [10, 136],

which we called here `document-document` relations; (ii) relations between terms based on similarity in sentences or in text collection [98, 73, 109, 121], order of occurrence [8, 109], or syntactic/semantic relationship [109, 111], which we called here `term-term` relations; or (iii) and relations between documents and terms [95, 99, 32]. In this article, we call these relations as *generalized relations*.

Supervised inductive learning approaches based on generalized relations, mainly based on `document-document` or [12] or `term-term` [121] relations, have not been surpassed the results provided by VSM-based algorithms. However, in [93, 99] is presented the *Inductive Model using Bipartite Heterogeneous Networks* (IMBHN), which is based on `document-term` relations, i.e., a bipartite heterogeneous network. IMBHN learning is based on assigning relevance scores to terms considering the document labels and network relations. Extensive experimental evaluations showed that IMBHN outperformed the classification performance of state-of-the-art classification algorithms based on VSM.

Despite the excellent results, IMBHN just considers document-term relations. However, another promising way to set the relevance scores of terms is considering `term-term` relations based on their similarity in the entire text collection, as presented in [98]. The `term-term` relations can be easily combined with `document-term` relations in a network. Thus, considering the facility to combine `document-term` and `term-term` relations, and according to the assumption that different relations can better represent the class patterns, in this article, we present a supervised inductive learning algorithm named *Inductive Model using Heterogeneous Networks* (IMHN). The proposed algorithms consider text represented in a heterogeneous network composed by `document-term` and `term-term` relations for assessing the relevance scores of terms for each class. The relevance scores of terms are obtained through an optimization process, minimizing the differences of the relevance scores of terms and the relevance scores of their related documents, and also minimizing the differences of the relevance scores with their related terms. To the best of our knowledge, this is the first algorithm which induces a classification model using more than one type of relation.

We conduct an extensive experimental evaluation considering the main algorithms based on VSM and based on networks and text collection from different domains and with different characteristics. The results show that the proposed algorithm presented better classification performance with statistically significant differences than all algorithms used in the comparison. Moreover, IMHN surpasses the classification performance of algorithms based on networks composed by relations of a single type in all evaluated collections.

The main contributions of this article fourfold:

- We propose a learning algorithm that effectively makes use of multiple types of relations to improve classification performance;
- We presented standardized notations for VSM and network-based algorithms to highlight the similarities and differences among them;
- We propose a supervised inductive learning algorithm which surpasses the classification performance of state-of-the-art learning algorithms based on VSM or networks.
- We conduct an extensive, comprehensive, and rigorous comparative evaluation of the proposed supervised inductive learning algorithm with traditional and state-of-the-art algorithms based on VSM and based on networks. These algorithms and IMHN were applied to 30 textual document collections from different domains and with different characteristics. The results show that

the proposed algorithm presented better classification performance with statistically significant differences than algorithms based on VSM or networks. Besides, to the best of our knowledge, there is no such evaluation in literature with the diversity of algorithms, algorithms parameters, and text collections as presented in this article.

The remainder of this article is organized as follows. Section 2 presents the background involving structured text representations, notations, and the main algorithms for supervised inductive learning algorithms based on VSM and networks. Section 3 details the proposed classification algorithm which induces a classification model using `term-term` and `document-term` relations. Section 4 presents the details of the experimental evaluation, results and discussions. Finally, Section 5 presents the conclusions and future work.

2. Background, Problem Definition, Notations & Related Works

Manual organization of digital texts is costly and sometimes impossible due to the hume volume generated nowadays. The, manual classifications such as performed in Dewey[1], MeSH[2], Yahoo![3] and DMOZ[4] [34], are difficult even with a large number of trained experts to perform such task. The use of specialists systems to perform automatic text classification has become obsolete due to the need of knowledge engineers/domains specialist, and the difficult to update and apply these systems in other applications/domains [70, 104, 34].

There has been an increasing interest in the use of machine learning techniques to perform automatic text classification due to the time spent in manual classifications or the building of expert systems. Machine learning techniques used for classification purposes infer a set of rules, hyperplanes, probabilities, or feature scores that compose a classification model. The classification models are easily induced and updated in comparison classifiers built by domain experts. Also, they are easily customized according to user needs [34, 104].

The inductive supervised learning process to build a classification model has been widely studied [132, 2, 123, 70, 35, 104, 34]. Only the content of the documents and their labels (category descriptors) are considered to perform learning. Traditional/state-of-the-art supervised machine learning algorithms developed for numeric data can also be applied to induce classification models for automatic text classification. However, some of these algorithms are inefficient due to the characteristic of text data, such as high dimensionality and sparsity. Therefore, some machine learning algorithms were developed specifically for text automatic classification [2, 93, 104].

The classification models generated by inductive supervised learning are applied for document retrieval [25, 133], e-mail filtering [29] and spam detection [9], opinion mining [66, 72], and authorship attribution [55], to cite a few. Moreover, non-textual domains has also been used automatic text classification techniques,

[1] *Dewey Decimal or Library of Congress Classification Systems*: `http://en.wikipedia.org/wiki/Dewey_Decimal_Classification`
[2] *Medical Subject Headings*: `http://en.wikipedia.org/wiki/Medical_Subject_Headings`
[3] Topic Hierarchies of Yahoo!: `https://dir.yahoo.com/`
[4] *Open Directory Project*: `http://en.wikipedia.org/wiki/DMOZ`

such as music genre classification [112], protein classification [59, 88], and intrusion detection [107, 64].

Most of the traditional/state-of-the-art algorithms consider the texts represented on the vector space model [2]. However, this way to represent texts preset some drawbacks, as will be presented in Section 2.2. Network-based representations have come-up as an alternative to vector space model and allow an efficient representation and pattern extraction that can not be accomplished by vector space model representations [98, 17]. Although the benefits of network-based representations, existing previous network-based algorithms usually present high computational cost or do not improve classification performance in comparison with the vector space model. Besides, some network-based representations are domain-specific, i.e., cannot be generalized to other domain applications.

In order to state the benefits of the proposed approach, in the next sections, we present the problem definitions, notations that will be used along the article, representations and algorithms based on vector space model and networks for inductive supervised learning in text classification.

2.1. Problem Definition & Notations

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_l\}$ represent the set of class labels, let $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ be the set of terms, and let $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ be the set of labeled documents of a text collection. Finally, let $\mathcal{Y} = \{y_{d_1}, y_{d_2}, \ldots, y_{d_n}\}$ be the labels of the labeled documents.

The goal of inductive supervised learning is to create a classification model to approximate a real category assignment function $R : \mathcal{D} \rightarrow \mathcal{Y}$. The predefined labels for a document $d_i \in \mathcal{D}$ are stored in a weight vector $\mathbf{y}_{d_i} = \{y_{c_1}, y_{c_2}, \ldots, y_{c_l}\}$, which has the value 1 in the position corresponding to the class and 0 in the other positions. The predefined labels of all labeled documents are stored in a matrix $\mathbf{Y} = \{\mathbf{y}_{d_1}, \mathbf{y}_{d_2}, \ldots, \mathbf{y}_{d_n}\}^T$.

Most of inductive supervised learning algorithms build classification models through the induction of class information of terms for classes, i.e., relevance score of terms for classes, hyperplane coefficients, or the probabilities of a term belongs to each class. We use $\mathbf{f}_{t_j} = \{f_{c_1}, f_{c_2}, \ldots, f_{c_l}\}$ to represent the class information of a term t_j for classes and the matrix $\mathbf{F} = \{\mathbf{f}_{t_1}, \mathbf{f}_{t_2}, \ldots, \mathbf{f}_{t_m}\}^T$ to store all class information of the terms.

A classification model is used to set the class information of documents, i.e., the weight, probability, or relevance score of a new document to each class of a text collection. The class information are used to set the label of a new document. We used a class information vector $\mathbf{f}_{d_i} = \{f_{c_1}, f_{c_2}, \ldots, f_{c_l}\}$ to store the class information assigned by a classification model to a new document d_i. For binary or multi-class classification, the class corresponding to the maximum value of $\mathbf{f}_{d_i}$ is used to set the label of a new document. For multi-label classification, techniques such as R-Cut, P-Cut, and S-Cut can be used to defined a set of labels based on the values of $\mathbf{f}_{d_i}$ [100, 128].

In this article, we used vector space model and networks to represent a text collection for inductive supervised learning. In the vector space model, each document is represented by a vector. We used a vector $\mathbf{w}_{d_i}$ to represent a document in a vector space model. The dimensions and their values are explained in the next subsection.

A network is defined by $N = \langle \mathcal{O}, \mathcal{R}, \mathcal{W} \rangle$, in which $\mathcal{O}$ represents the set of

objects, $\mathcal{R}$ represents the set of relations among objects, and $\mathcal{W}$ represents the weights of the relations. According to the study area, objects can also be called nodes, vertices or actors, and relations can be also be treated by edges or links [79].

The networks can be undirected, i.e., if there is a relation between an object o_i and an object o_j ($r_{o_i,o_j} \in \mathcal{R}$), there is also a relation between an object o_j to o_i ($r_{o_j,o_i} \in \mathcal{R}$). Networks can also be weighted or unweighted. In case of unweighted networks, the weight w_{o_i,o_j} is the same for each $r_{o_i,o_j} \in \mathcal{R}$. On the other hand, weighted networks are used if the intent is to quantify the strength of the relation. In this case, w_{o_i,o_j} can correspond to the frequency in which the relation occurred, probability to occur a connection or the similarity between the objects. The relation weights of an object o_i with other objects are also represented by a vector $\mathbf{w}_{o_i} = \{w_{o_i,o_1}, w_{o_i,o_2}, \ldots, w_{o_i,o_{|\mathcal{O}|}}\}$ to facilitate the understanding of the similarities and differences between vector space model and network-based algorithms.

Finally, a network can be homogeneous or heterogeneous. When $\mathcal{O}$ is composed by a single-type object or R contains a single-type relation, the network is called homogeneous network. Otherwise, the network is called heterogeneous network [114].

2.2. Text Representation

Text representation is a fundamental step to perform machine learning for text classification. Here we present two standard ways to generate a structured representation of texts for inductive supervised learning: vector space model and networks.

2.2.1. Vector Space Model

Vector space model representations are the most common in machine learning [2, 105, 110]. Therefore, a wide range of algorithms was developed considering this type of representation.

In the vector space model, each document is represented by a vector, and the dimensions correspond to features of the text collection. Also, there is an additional dimension to store the class label of a document in case of supervised learning. The first step to build the document-term matrix is to define the features of the text collections and consequently the dimensions of the vector space model. This step has a direct impact on the classification performance and building time of a classification model [119].

Words can compose the features. In this case, the features are also treated by terms, and the join of document vectors forms the document-term matrix, as presented in Figure 1.

Commonly, single words are used as terms, generating the *bag-of-words* representation. The bag-of-words is characterized by a high dimensionality, since the number of terms is equal to vocabulary size, and high sparsity, since just a small fraction of terms of a text collection will occur in each document. The vocabulary size can be reduced through some steps such as the removal of stop-words and tokens composed by non-alphabetic characters, and word simplification (stemming and lemmatization) [2].

Although the high dimension and sparsity characteristics, bag-of-words has

[htb]

Table 1. Illustration of a document-term matrix for a text collection with n documents and m terms.

	t_1	t_2	t_3	...	t_{m-2}	t_{m-1}	t_m	Class
d_1	w_{d_1,t_1}	w_{d_1,t_2}	w_{d_1,t_3}	...	$w_{d_1,t_{m-2}}$	$w_{d_1,t_{m-1}}$	w_{d_1,t_m}	c_{d_1}
d_2	w_{d_2,t_1}	w_{d_2,t_2}	w_{d_2,t_3}	...	$w_{d_2,t_{m-2}}$	$w_{d_2,t_{m-1}}$	w_{d_2,t_m}	c_{d_2}
d_3	w_{d_3,t_1}	w_{d_3,t_2}	w_{d_3,t_3}	...	$w_{d_3,t_{m-2}}$	$w_{d_3,t_{m-1}}$	w_{d_3,t_m}	c_{d_3}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
d_{n-2}	w_{d_{n-2},t_1}	w_{d_{n-2},t_2}	w_{d_{n-2},t_3}	...	$w_{d_{n-2},t_{m-2}}$	$w_{d_{n-2},t_{m-1}}$	w_{d_{n-2},t_m}	$c_{d_{n-2}}$
d_{n-1}	w_{d_{n-1},t_1}	w_{d_{n-1},t_2}	w_{d_{n-1},t_3}	...	$w_{d_{n-1},t_{m-2}}$	$w_{d_{n-1},t_{m-1}}$	w_{d_{n-1},t_m}	$c_{d_{n-1}}$
d_n	w_{d_n,t_1}	w_{d_n,t_2}	w_{d_n,t_3}	...	$w_{d_n,t_{m-2}}$	$w_{d_n,t_{m-1}}$	w_{d_n,t_m}	c_{d_n}

been presented competitive classification performances in several application domains. However, only single words cannot be enough in some application domains or in short text classifications [97, 20]. So, some alternatives to bag-of-words were proposed. Some of them consist of considering sequences of n words as terms, called n-grams or statistical phrases [20, 21], or sequences of words that satisfy syntactic structures, called syntactic phrases [76]. Also, sets of words that frequently co-occur in documents, sentences or windows of words can also be used as text features [97, 135, 33, 131]. Although these features can be useful, their use can increase dimensionality, and consequently time and space consumption of the machine learning algorithms.

When choosing terms as text features, the weights of the terms in each document must be assigned. The way to define a a weight of terms in documents are called term weighting scheme and the weight of a term t_j in a document d_i will be denoted by w_{d_i,t_j}. Usually unsupervised term weighting schemes are used. Traditional examples of those schemes are: (i) binary, i.e., w_{d_i,t_j} is 1 if t_j occurred in d_i and 0 otherwise; (ii) term-frequency (tf), in which w_{d_i,t_j} correspond to the number of times a term t_f occured in document d_i; and (iii) term-frequency-inverse-document-frequency (*tf-idf*), in which the frequency of a term t_i in document d_j weighted by the inverse function of the number of documents. Also, supervised term-weighting schemes, can be used, i.e., the weights of the terms also consider the class of the documents. Basically, those schemes consider some purity measure in place of *idf*, such as chi-squared statistics, information gain or gain ratio [24, 7]. The different term weighting-schemes can present impact in classification performance in some specific tasks or algorithms [31, 86, 61, 89].

Due to the high dimension that the vector-space model may present, feature selection and dimensionality reduction techniques can be used to decrease the number of dimensions [36, 57]. In feature selection, a subset of features will be maintained in text representation. This can be performed in an unsupervised way, such as defining a minimum document frequency for a term, select the top-ranked terms considering the sum of their weights in a text collection. In a

supervised way, measures such as Gini Index, Information Gain, Pointwise Mutual Information, and (iv) χ^2 can be used to evaluate subsets of terms or ranking terms [2, 36]. Another approach is to evaluate subsets of terms in the classification model (wrapper approach). However, this procedure is time expensive [116].

The dimensionality reduction techniques can also be viewed as a semantic space projection in which the same attribute represents semantically related terms. So, those techniques are a way to represent relations among terms in the vector space model without causing a huge increase in the number of dimensions. In fact, the number of generated dimensions is usually way smaller than bag-of-words. According to [103], dimensionality reduction technique provide better results than statistical or syntactic phrases, or feature selection algorithms.

The most common dimensionality reduction techniques for text mining are *Latent Dirichlet Allocation* (*LDA*) [15], *Probabilistic Latent Semantic Analyzis* (*PLSI*) [49] and *Non-negative Matrix Factorization* (*NMF*) [127]. PLSA and LDA are probabilistic techniques for topic extraction. The topics correspond to related terms (or subjects) in a text collection. The results of topic extraction are a matrix that contains the weight of terms for the topics (term-topic matrix), a matrix that contains the weight of documents to topics (document-topic matrix), and a vector to store the probability of the topics occur in the text collections. The document-topic matrix can be used as a new representation of the text collections. The NMF is based on matrix factorization. The goal is to generate a document-topic matrix and a term topic-matrix in a way that their multiplication is able to reproduce an original vector space model representation.

Another way to perform dimensionality reduction is through word embeddings [2]. Word embedding corresponds to words mapped in a numeric vector with a predefined size. Words with similar meanings will have similar vectors in the embedding representation. Methods to generate this mapping include neural networks (Continuous Bag-of-Words model (CBOW) and the Skip-Gram model) and dimensionality reduction techniques (term-topic matrix). The word embedding representation can be included as a layer in a neural network or can be used to generate a structured representation for documents, e.g., making the average of the word vectors that occur in a document, or a weighted sum by *tf-idf* values [53, 65].

Besides the relations among terms that can be obtained by sets of sequences of terms, or dimensionality reduction techniques, relations between document can be represented in a vector space model, such as hyperlinks, citations, or co-authoring [6, 5, 77]. However, modeling these relations implies an increase in the dimensionality and sparsity of the representation. For instance, there must be a feature for every hyperlink between a document d_i and a document d_j. Another disadvantage is that even relations are represented, VSM-based algorithms do not consider chains of relations. For instance, if a document d_i are related to a document d_j, and a document d_j are related to a document d_k, this chain of relation could be useful to put d_i and d_k in the same group or make them belong to the same class.

2.2.2. Networks

Networks are a natural and direct way to generate a structured representation for different text mining and natural language processing tasks [73]. According

[htb]

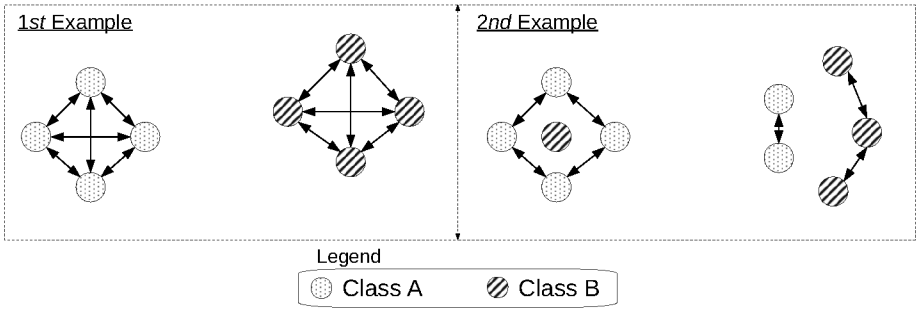


Fig. 1. Illustration of a k-associated network with $k = 3$ in two different document collections.

to [14], networks are a powerful tool to represent topological, statistical, and grammatical aspects in a single and mathematical representation.

Different networks, i.e., networks with different types of objects and relations can be generated from texts and text collections for different purposes in Text Mining and Natural Language Processing tasks, such as text summarization, keyword extraction, document clustering, word sense disambiguation, and machine translation, to cite few [73, 109]. In this section, we present details of network representations used in inductive supervised learning for text classification. They are divided into two categories: homogeneous and heterogeneous networks.

2.2.3. Homogeneous Networks

There are two basic types of homogeneous networks for text classification through inductive supervised learning: document networks and term networks.

In a document network, $\mathcal{O} = \mathcal{D}$, i.e., the network objects represent the documents of a text collection. Relations among documents can be given by: (i) explicit relations, such as hyperlinks or citations [69, 81, 22]; or (ii) implicit relations, such as the similarity [12]. Here we focus on similarity-based document networks since it can model any type of text collection and also provide better results than document networks generated thought explicit relations [10].

There are some approaches to generate a document network based on similarity, such as *k-Nearest Neighbor Network*, *k-Neighborhood* or *b*-matching [50]. However, they are usually used in unsupervised or semi-supervised learning. In a supervised learning scenario, a more adequate approach is to consider the labels to generate the document networks. So, in [12] is presented the k-Associated Network (k-AN).

The building of k-AN is carried out in the following way: (i) for each document d_i, the k nearest neighbors of d_i are computed; and (ii) a directed edge are generated from d_i to d_j if d_j is one of the k nearest neighbors of d_i, and d_i and d_j belongs to the same class ($y_{d_i} = y_{d_j}$). A characteristic of this network is that it is necessarily composed of multiple components (at least one for each class), and those components contains documents of a single class.

In Figure 1 is presented two illustrations of a k-AN for a text collection with two classes and considering $k = 3$. In [12] is also presented a solution to automatically define the value of k for each component, generating the k-Optimal Associated Network (k-OAN).

[htb]

HTML

```
<html>
    <head>
            <title>Machine Learning</title>
    </head>
    <body>
        Different types of learning
        <a herf=...> Unsupervised Learning </a><br>
        <a href=...> Semissupervised Learning </a><br>
        <a href=...> Supervised Learning </a><br>
    </body>
</html>
```

Network Representation

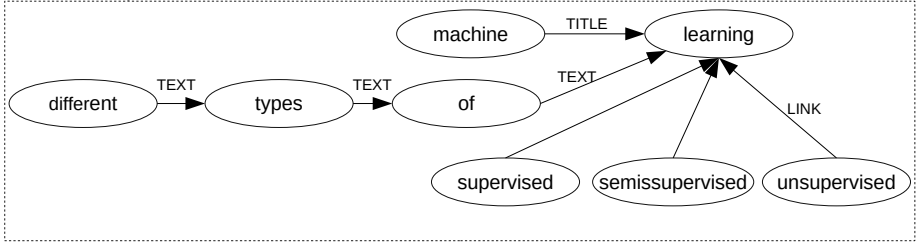


Fig. 2. Illustration of a term network extracted from an HTML example page according to the approach presented [102].

In a term network, $\mathcal{O} = \mathcal{T}$, i.e., the network objects represent the terms of a text collection. There are four common strategies to generate a term network for inductive supervised learning[5]: (i) extract a term network for each document and compute the similarities among the term networks; (ii) extract network features to generate a vector space model representation; (iii) use term networks as a term weighting scheme for vector space model; and (iv) extract vector prototypes.

Examples of approaches that extract term networks for each document and compute the similarities among them are presented in [102] and [75]. In [102] is proposed a directed term network to represent HTML pages. A directed edge is generate from t_i to t_j if a term t_i precedes a term t_j. Moreover, the edges are labeled according to the section that the relation occurred, e.g., TITLE section, body of HTML (named TEXT), and in and hyperlink reference environment (named LINK). The weights of the edges correspond to the frequency in which the relation occurred. Infrequent relations are pruned in this proposal. In Figure 2 is presented an illustration of this type of network.

In [75], authors proposed the generation of a term network based on semantics, named entity recognition, and controlled vocabulary of a biomedical domain. Semantic relations considering words in the controlled vocabulary and extracted entities are searched to generate the networks. The relation types in this network comprises parent, narrow and sibling. In Figure 3 are presented an illustration of a network of a biomedical text according to this proposal.

Examples of approaches that extract network features to generate a vector space model representation are presented in [71, 4, 8]. In [71], the same network

[5] Independently of the strategy to generate relations among terms, we called this type of relation as `term-term` relations.

[htb]

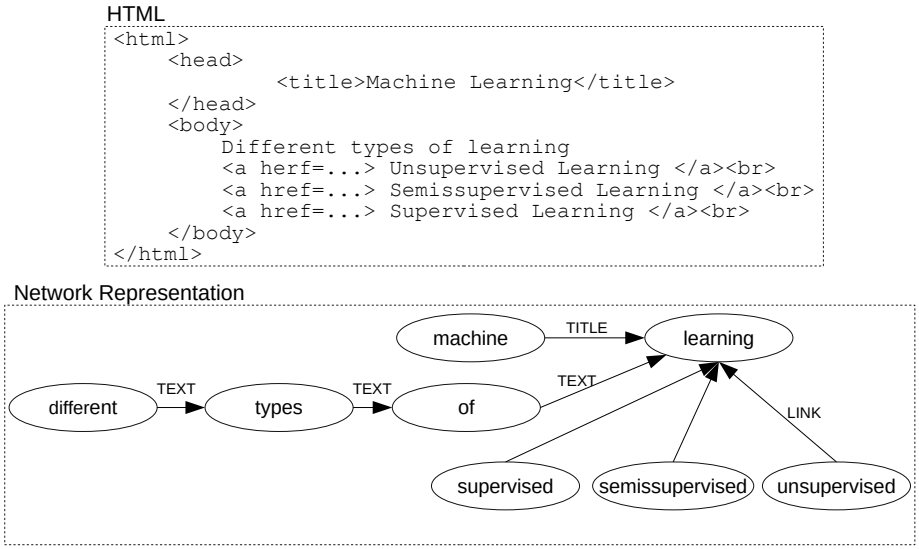


Fig. 3. Illustration of a network of a biomedical according to the approach presented in [75].

proposed in [102] are considered. However, frequent sub-networks are extracted and considered as features in a vector space model representation.

In [4], a directed and unweighted term network is generated for each document, in which a directed relation is generate from t_i to t_j if a term t_i precedes a term t_j in k positions (k is called order). Also, an self-edge is generated for a term t_i when it occurs in a document (this is equivalent to $k = 0$). This type of network is called distance network. The relation weights correspond to the number of times that the relation occurred. Text pre-processing steps are carried out before the network generation. In Figure 5 is presented an illustration of the distance network. After network generation, an attribute in the vector space model is generated for each edge.

In [8], a term network is generated for each document. An undirected relation from a term t_i to a term t_j is generated if t_j succeeds t_i in the document. This type of network is called word adjacency network. Pre-processing steps, such as word sense disambiguation and the transformation of words to its canonical forms, are applied before network generation. Once the network is generated, a set of measures to characterize the topological structure of the network, such as clustering coefficient, average degree, average shortest path length, betweenness, and intermittency are computed. Those measures and their values are used as features and feature weights, respectively.

An example of a strategy that uses a term network to define term weights for vector space model representations is reported in [47]. In this work, a co-occurrence network is generated for each document. In this type of network, a term t_i is linked to a term t_j if they co-occur in a sliding window of words. After network generation, the PageRank algorithm [74] is carried out, and the weights assigned to the nodes will correspond to the term weights in the vector space model. In Figure 6 is presented an illustration of this approach.

[htb]

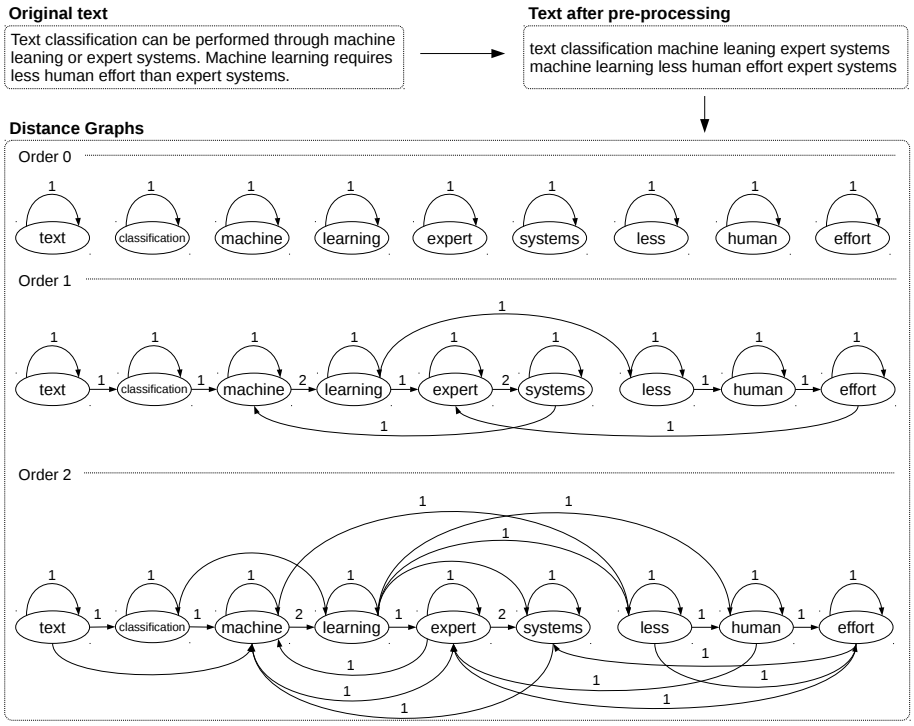


Fig. 4. Example of a distance network [4] with three different orders: 0, 1, and 2.

[htb]

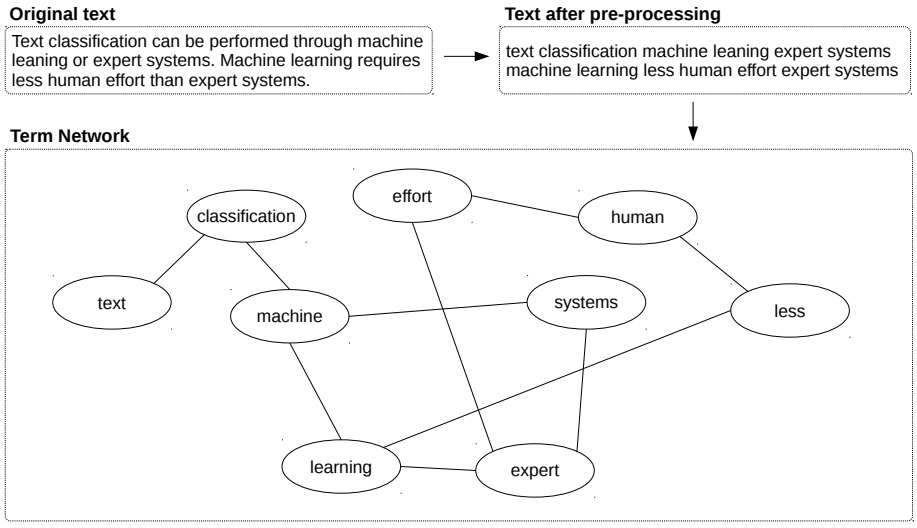
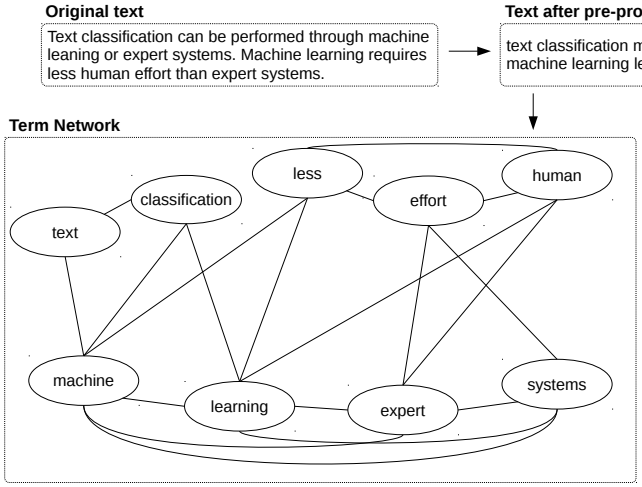


Fig. 5. Illustration of a word adjacency network [8].

[htb]

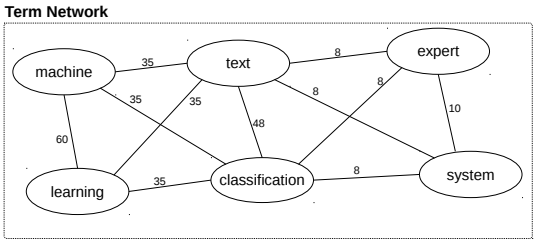


Term	PageRank Score
machine	0.1687
learning	0.1403
effort	0.1186
less	0.1181
expert	0.1138
systems	0.1138
classification	0.0928
human	0.0670
text	0.0669

Fig. 6. Illustration of a term network generated with a sliding window of size 3 and according to the proposal presented in [47]. Also the term weights obtained considering the PageRank algorithm ($\alpha = 0.85$, maximum number iterations $= 100$ and tolerance value $= tol = 1e-06$) are presented.

[htb]

Frequent Itemssets	Itemset Support
{machine, learning}	60
{text, classification}	48
{machine, learning, text, classification}	35
{expert, system}	10
{expert, system, text, classification}	8



Term	PageRank Score	Rank
text	0.2241	1
classification	0.2241	1
machine	0.2049	3
learning	0.2049	3
system	0.0709	5
expert	0.0709	5

Fig. 7. Illustration of a term network generation and scores of terms through PageRank algorithm ($\alpha = 0.85$, maximum number iterations $= 100$ and tolerance value $= tol = 1e-06$) according to the approach presented in [121].

In a similar way to [47], in [121] is also applied the PageRank algorithm in a term network. However, PageRank is applied in an undirected term network generated for each class. Itemset frequent mining is applied to generate relations between terms. In this case, each document is considered as a transaction, and just terms that co-occur above minimum support are connected in the network. Also, the relation weight corresponds to the largest support value among all the frequent itemsets that contain both terms. PageRank is then applied to generate scores for terms. The terms and their ranks are used as class prototypes.

In summary, we can observe that the generation of term networks (i) are

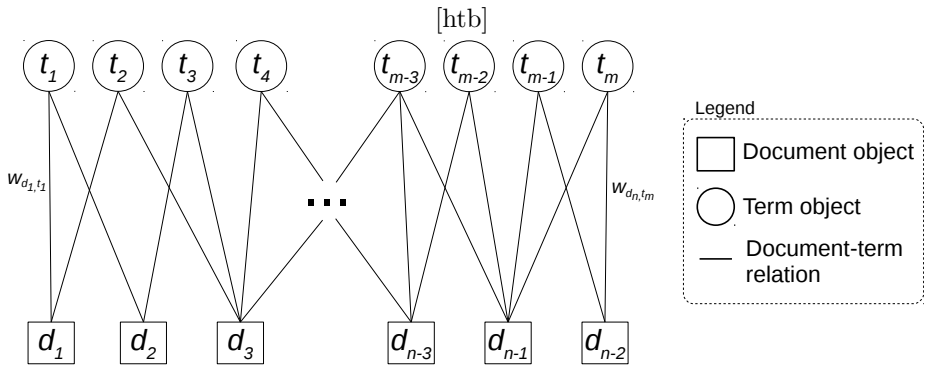


Fig. 8. Illustration of a bipartite network for a text collection with n documents and m terms.

domain-dependent, as in [102] and [75]; (ii) can also generate a high dimensionality in vector space model, as in [102], [75] and [4]; (iii) and steps such as computing frequent sub-networks increase the computational cost of the structured representation generation. On the other hand, some approaches such as presented in [71], [4], and [8] has the advantage to generate vector space model representations, which allows the application of a wide number of algorithms based on this type of representation. However, those approaches do not always improve the results obtained by other vector space model representations such as bag-of-words.

2.2.4. Heterogeneous Networks

One of the simplest heterogeneous network to represent text collections is the bipartite network [93, 32]. In a bipartite heterogeneous network, $\mathcal{O} = \mathcal{D} \cup \mathcal{T}$, i.e., the network objects correspond to documents and terms of a text collection. An object $d_i \in \mathcal{D}$ and an object $t_j \in \mathcal{T}$ are linked if t_j occurs in d_i [99][6]. The weight of the relation between d_i and t_j (w_{d_i,t_j}) is based on the frequency of t_j in d_i. Thus, just the terms and their frequencies in the documents are necessary to generate a bipartite network. Both bipartite networks presented in [93] and [32] are undirected. In Figure 8 we present an illustration of a bipartite heterogeneous network.

In [1], a star network is used to represent each document. The star network structure is composed of a central object, in this case, a document, and objects connected to the central object, in this case, terms. The relations are labeled according to the section that the document occurs (e.g., title, body o link of and HTML page). In Figure 9 is presented an illustration of this network. After network extraction for each document, frequent sub-networks are extracted to generate features in the vector space model. Also, these features are weighted according to the representatives of the sub-networks for the classes.

In [52], words, stems, syntactic, and semantic relations are used to build a directed network from a text. In Figure 10 is presented and illustration of this proposal for the sentence *"The quick brown fox jumped over."*. After the

[6] We called the relations among documents and terms as `document-term` relations.

[htb]

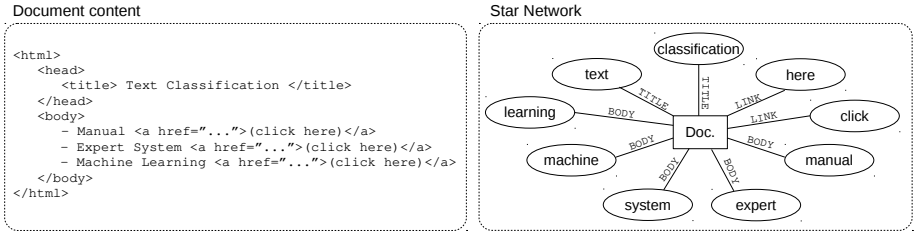


Fig. 9. Illustration of a star network to represent a document according to the proposal presented in [1].

[htb]

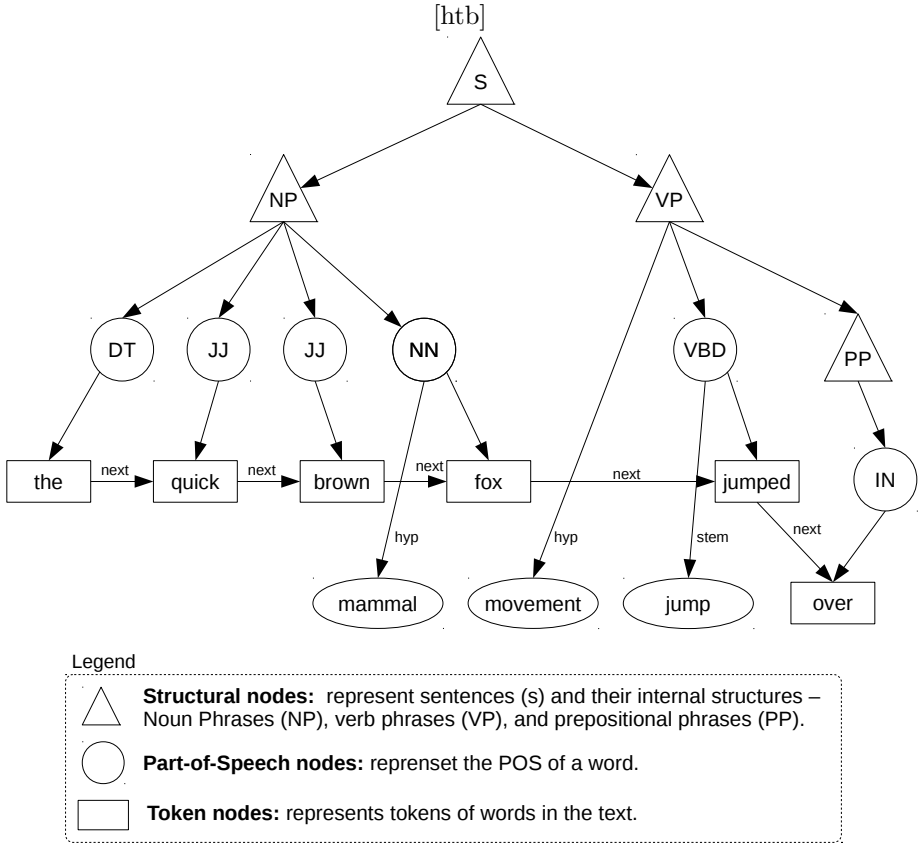


Fig. 10. Illustration of the network generated for the sentence *"The quick brown fox jumped over."* according to the proposal presented in [52] (adapted from [52].

extraction of the networks for each text, the sub-frequent networks are extracted and considered as binary features in a vector space model representation.

Some drawbacks presented in homogeneous networks also occur in heterogeneous network representations, e.g., limitation to some specific domain [1, 52], requirement of semantic resources for the network generation, and increasing in

the computational cost due to the computation of frequent sub-networks. [52]. The exception is the bipartite networks, which are domain-dependent and can be applied to represent any text collection. Besides, bipartite networks are generated faster than document networks and most of the term-networks, since they do not need to compute similarities among all documents, terms, or perform parsing or POS-tagging. Also, bipartite networks do not require parameters, which can drastically change the classification performance in document networks [95, 30]. Moreover, bipartite networks have been providing competitive results with vector space model representations [95, 99].

However, the drawback of the bipartite network representation is that just simple relations between documents and terms are represented. Therefore, it is limited to a single-type relation, and as presented in Section 1, considering different types of relations may provide a better representation of the text collection and consequently improve classification performance. In Section 3 we present a network proposal which extends the bipartite networks including `term-term` relations. We called those generalized relations since they can be extracted from text collection of any domain.

2.3. Inductive Supervised Learning

In this section we present the main inductive supervised learning algorithms for text classification. The algorithms based on vector space model are divide into learning categories, and we chose the most representative algorithms per category. The algorithms based on network are the ones that are domain independent, since the experimental evaluation carried out in this article considers text collections of different domains.

2.3.1. Inductive Supervised Learning Considering Texts Represented on the Vector Space Model

Due to the characteristics of some vector space model representations for text collections, such as high dimensionality and sparsity, some new algorithms and variations of existent inductive supervised learning algorithms are proposed to deal with text data. Here we present the algorithms adapted, indicated or developed specifically to perform text automatic classification organized in 5 learning categories: (i) probabilistic, (ii) linear, (iii) rule-based, (iv) decision tree-based, and (v) similarity-based.

The goal of probabilistic classification is to infer the probability of class c_j occurs given a document d_i ($P(c_j|d_i)$). This probability corresponds to the class information of d_i. Two traditional probabilistic learning algorithms are Naïve Bayes and Logistic Regression.

Naïve Bayes is the most simple probabilistic leaning algorithm [3, 58]. In this algorithm, $P(c_j|d_i)$ é computed by Bayes rule, i.e.,

$$f_{d_i,c_j} = P(c_j|d_i) = \frac{P(d_i|c_j)P(c_j)}{P(d_i)}, \tag{1}$$

where $P(c_j)$ is the probability of the class c_j occurs, and $P(d_i)$ is the probability

of the document d_i occurs. The main aspect of Naïve Bayes is the assumption that terms occur independently. So, $P(d_i|c_j)$ is computed by:

$$P(d_i|c_j) = \prod_{t_k \in \mathcal{T}, w_{d_i,t_k}>0} P(t_k|c_j), \tag{2}$$

where $P(t_k|c_j)$ is the probability of the term t_k belong to the class c_j (class information of terms). Two models commonly used to compute $P(t_k|c_j)$ are: (i) Bernoulli Model, which considers just the presence or absence of words, or (ii) Multinomial Model, which considers the frequency of words. Multinomial usually outperforms Bernoulli Model [2]. $P(t_k|c_j)$ computed thought Multinomial model is given by:

$$P(t_k|c_j) = f_{t_k,c_j} = \frac{1 + \sum_{d_l \in \mathcal{D}} w_{d_l,t_k} y_{d_l,c_j}}{|\mathcal{T}| + \sum_{t_n \in \mathcal{T}} \sum_{d_m \in \mathcal{D}} w_{d_m,t_n} y_{d_l,c_j}}. \tag{3}$$

There are also variations found in literature to compute $P(d_k|c_j)$ which considers the orders that terms in d_i can be sampled and the probability of occurring these samples [3, 125].

$P(c_j|d_i)$ can also be inferred using *Bayesian Logistic Regression* (BLR) [40]. In this algorithm, $P(c_j|, d_i)$ is given by:

$$P(c_j|, d_i) = \varphi(\sum_{t_k \in \mathcal{T}} f_{t_k,c_j} \cdot w_{d_i,t_k}), \tag{4}$$

where $\varphi(...)$ is a logistic function presented in Equation 5.

$$\varphi(\sum_{t_k \in \mathcal{T}} f_{t_k,c_j} \cdot w_{d_i,t_k}) = \frac{exp(\sum_{t_k \in \mathcal{T}} f_{t_k,c_j} \cdot w_{d_i,t_k})}{1 + exp(\sum_{t_k \in \mathcal{T}} f_{t_k,c_j} \cdot w_{d_i,t_k})} \tag{5}$$

The goal of logistic regression is induce information class of terms to maximize conditional likelihood $\prod_{d_i \in \mathcal{D}} P(y_{d_i}|d_i)$. Some researches advise the use of Gaussian and Laplace priors due to the overfitting problem and increase efficiency [40, 35].

The goal of linear classification is to induce weights of terms for classes and make predictions through a linear function resulted in the combination of the set of weights with the feature vector. Basically, there are two ways to perform such task: (i) induce weights of terms for classes to predict precisely the real class information vectors of the training examples, i.e., $\mathbf{y}_{d_i}, d_i \in \mathcal{D}$, or (ii) induces weights of terms for classes to generate a hyperplane to separate each pair of classes.

In the first case, the goal is to minimize the following function:

$$Q(\mathbf{F}(\mathcal{T})) = \sum_{d_i \in \mathcal{D}} \sum_{c_j \in \mathcal{C}} \left|\left| \sum_{t_k \in \mathcal{T}} f_{t_k,c_j} w_{t_k,d_i} - y_{d_i,c_j} \right|\right|^2, \tag{6}$$

in which $\mathbf{F}(\mathcal{T})$ is the matrix that stores the class information of terms. Equation 6 can be minimized using the *Linear Least Squares Fit* (LLSF) [130, 129] which

performs linear regression through Singular Value Decomposition to estimate to class information of terms. There is also a regularized version of LLSF, which uses Ridge Regression to set the class information of terms [63, 134]. In this case, the function to be minimized is given by:

$$Q(\mathbf{F}(\mathcal{T})) = \sum_{d_i \in \mathcal{D}} \sum_{c_j \in \mathcal{C}} \left|\left| \sum_{t_k \in \mathcal{T}} \right|\right|^2 + \lambda \sum_{c_j \in \mathcal{C}} \sum_{t_k \in \mathcal{T}} ||f_{t_k,c_j}||^2. \tag{7}$$

The regularization term (second term of Equation 7) makes the information class of terms tend to 0 for high values of λ. The regularization parameter is used to reduce data overfitting.

Another set of linear classifiers aims to induce a hyperplane to separate the examples from different classes. A hyperplane is defined by:

$$\sum_{t_k \in \mathcal{T}} f_{t_k} \cdot w_{d_i,t_k} + b, \tag{8}$$

where f_{t_k} represents the linear coefficient corresponding to term t_k (weight of assigned to t_k) and b is the bias. Since the hyperplane is able to separate a pair of classes, i.e., a binary classifier, techniques such as `one-vs-one` or `one-vs-all` must be carried out [116]. In those approaches, iteratively one class will be considered as positive (+1) and other(s) will be considered as negative (-1). A document d_i is labeled as +1 if it is above the hyperplane and -1 is below the hyperplane, i.e.,

$$y_{d_i} = \begin{cases} +1, & \text{se } \sum_{t_k \in \mathcal{T}} f_{t_k} \cdot w_{d_i,t_k} + b > 0; \\ -1 & \text{se } \sum_{t_k \in \mathcal{T}} f_{t_k} \cdot w_{d_i,t_k} + b < 0. \end{cases} \tag{9}$$

Thus, the goal is inducing a hyperplane in a way that all positive documents are above the hyperplane and all negative documents are below the hyperplane. One of the state-of-the-art to induce hyperplane to separate documents from different classes are *Support Vector Machines* (SVM) [54, 120]. This algorithms has a different characteristic from other classifiers: it obtains hyperplane with maximal margin separation, i.e., the hyperplane which distance from examples of different classes are maximal. This is obtained through the minimization of the function:

$$Q(\mathbf{f}(\mathcal{T})) = \frac{||\mathbf{f}||^2}{2} + C(\sum_{d_i \in \mathcal{D}} \xi_{d_i}) \tag{10}$$

$$\text{subjected to } y_{d_i}(\sum_{t_k \in \mathcal{T}} f_{t_k} \cdot w_{d_i,t_k} + b) \geq 1, \forall d_i \in \mathcal{D},$$

in which *slack* variable (ξ) contains the error (distance do the hyperplane) when d_i is on the wrong side of the hyperplane, and C controls the trade-off between error and the size of the margin. SVM can also be modified to consider non-linear separations through the use of kernels [70]. The non-linear hyperplanes allow obtaining solutions similar to other classifiers, such as Neural Networks [46].

The goal of rule-based classification is to induce a set of rules such as:

$$freq(t_a) > X \wedge freq(t_b) > Y \wedge \ldots \wedge freq(t_c) \leq Z \ldots \rightarrow c_j,$$

in which $freq(t_a)$, $freq(t_b)$ and $freq(t_c)$ represents the frequency of terms t_a, t_b and t_c respectively, X, Y are Z numeric values, $\wedge$ is the `AND` boolean operator, and c_j is a class from a text collection. The left hand side contains a conjunction of conditions and the right hand side contains a class. For each class, the rules are induced until they cover all training examples, i.e., all training documents satisfies the conditions of at least one generated rule.

Rule-based algorithms are usually greedy. They start with an empty rule, and this rule is increased with conditions until the examples covered by this rule are correctly classified. The *Repeated Incremental Pruning to Produce Error Reduction* (RIPPER) algorithm [26, 27] is one of the most used rule-based algorithms used for text classification. In RIPPER, the documents covered by a rule are removed from the training set. Then, another rule is built in a greedy way. This process is repeated for all classes.

RIPPER considers grow rules with simple conditions using a subset of training documents named $\mathcal{D}^{Grow}$ (usually 2/3 of the training set), and prunes rules using a subset of training documents named $\mathcal{D}^{Prune}$ (usually 1/3 of the training set). A rule starts with an empty condition, and the conditions are added to this rule according to the highest relative information gain [27]. Thus, a rule r_i is incremented r_i' considering the maximization of the following function:

$$Gain(r_i', r_i) = |\mathcal{D}^{Grow}_{r_i', c_j}| \cdot \Big(-\log_2 \frac{|\mathcal{D}^{Grow}_{r_i, c_j}|}{|\mathcal{D}^{Grow}_{r_i, c_j}| + |\mathcal{D}^{Grow}_{r_i, \mathcal{C}-c_j}|} + \log_2 \frac{|\mathcal{D}^{Grow}_{r_i', c_j}|}{|\mathcal{D}^{Grow}_{r_i', c_j}| + |\mathcal{D}^{Grow}_{r_i', \mathcal{C}-c_j}|} \Big) \quad (11)$$

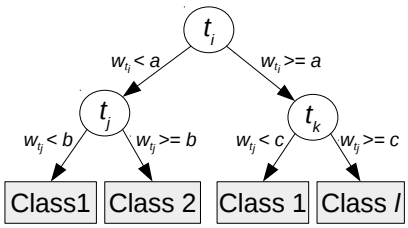
where $|\mathcal{D}^{Grow}_{r_i, c_j}|$ is the number of documents of the class c_j covered by r_i in $\mathcal{D}^{Grow}$. After reach the stopping criteria, final conditions of the rule are pruned in a way to maximize the function:

$$f(r_i) = \frac{|\mathcal{D}^{Prune}_{r_i', c_j}| - |\mathcal{D}^{Prune}_{r_i', \mathcal{C}-c_j}|}{|\mathcal{D}^{Prune}_{r_i', c_j}| + |\mathcal{D}^{Prune}_{r_i', \mathcal{C}-c_j}|}, \quad (12)$$

where $|\mathcal{D}^{Prune}_{r_i', c_j}|$ is the number of documents from class c_j covered by the new rule r_i' considering documents in $\mathcal{D}^{Prune}$. The pruning is used to avoid overfitting. A new document is classified according to the class of the rules that cover it. Some sorting criteria can also be used in order to chose one classification rule, such as, the size of the rule.

Decision tree-based algorithms induce a classification tree in a way that (i) each non-leaf (internal) is labeled with a term $t_i \in \mathcal{T}$, (ii) branches represent tests on the weight of the terms ($<$ or $\geq$), and (iii) leaf nodes are labeled with a class $c_i \in \mathcal{C}$. In Figure 11 we present an illustration of a decision tree.

Decision trees are usually built in a greedy way, and a term t_i and a test on the term value that best splits the set of documents from different classes is chosen as a branch. The best split is defined using measures based on information theory, such as *Entropy*, *Gini Index*, and *Information Gain* [116]. When a splitting produces a pure partition, i.e., documents of a single class, a leaf node labeled with the corresponding class is generated. Besides a single decision tree, more than a decision tree can be generated from a dataset. For instance, in the Random Forests algorithm, decision trees are induced in different data



[htb]

Fig. 11. Illustration of a decision tree.

sub-samples of the dataset. Then, the different decision tree acts as an ensemble to classify new documents [11].

Decision trees can also be pruned through prepruning or post-pruning techniques to avoid data overfitting [116, 87] and increase the classification performance [116]. Those procedures can make leaf nodes impure. In this case, the leaf node is labeled considering the majority class of the covered examples. A new document is classified through the class of the leaf node reached by the the conditions satisfied by the features of the new document.

Documents can also be classified considering the similarities among them. Basically, there are two approaches: (i) instance-based classification and (ii) prototype-based classification. In instance-based classification, a new document is directly compared with other training documents, and the class of the most similar documents are used to label the new document.

The most used instance-based algorithm is *k-Nearest Neighbors* (kNN) [3, 126, 45]. In this algorithm the labels of the k most similar documents (also called neighbors) are considered in the classification of a new document. The classification performed by kNN can be *unweighted* or *weighted*. Unweighted classification considers the number of classes of the nearest neighbors to define the class information of a new document d_i, i.e.,

$$f_{d_i,c_j} = \sum_{d_m \in \mathcal{K}^{d_i}} y_{d_m,c_j}, \tag{13}$$

where $\mathcal{K}^{d_i}$ represents the set of k nearest neighbors of d_i. Weighted classification considers the similarity of the neighbors to weight their contribution to set the class information of a new document d_i, i.e.,

$$f_{d_i,c_j} = \sum_{d_m \in \mathcal{K}^{d_i}} \frac{1}{(1 - sim(\mathbf{w}_{d_i}, \mathbf{w}_{d_m})) \cdot y_{d_m,c_j}}. \tag{14}$$

In both ways to perform kNN, a new document d_i is classified considering the class corresponding to the maximum value of $\mathbf{f}_{d_i}$.

Prototypes can be used to speed up the classification performance of similarity-based algorithms. The goal of prototype-based classification is to create and representative object, also called prototypes or pseudo-instances, to represent each class $c_i \in \mathcal{C}$ and then to compute the similarity of a new document only to these prototypes [44, 60, 91].

One of the most famous prototype-based algorithm is Rocchio [91]. The i-th dimension prototype of a class c_j is given by:

$$f_{t_i,cj} = \frac{\alpha}{|\mathcal{D}_{c_j}|} \sum_{d_k \in \mathcal{D}_{c_j}} w_{d_k,t_i} - \frac{\beta}{|\mathcal{D}_{\mathcal{C}-c_j}|} \sum_{d_k \in \mathcal{D}_{\mathcal{C}-c_j}} w_{d_k,t_i}, \tag{15}$$

where $\mathcal{D}_{c_j}$ is the set of labeled documents from class c_j and $\mathcal{D}_{\mathcal{C}-c_j}$ is the set of labeled document from a class different from c_j, α and β weight the importance of documents from c_j and c_k in the prototype definition. *Centroid Based Classifier* (CBC) [44, 3] is a special case of Rocchio for $\alpha = 1$ and $\beta = 0$. A new document will be labeled with a class corresponding to the most similar prototype.

Usually, the cosine measure is used to compute similarities in both instance-based and prototype-based text classification [116]. The cosine similarity between the weight vectors of documents d_i and d_j are given by:

$$cosine(\mathbf{w}_{d_i}, \mathbf{w}_{d_j}) = \frac{\mathbf{w}_{d_i} \cdot \mathbf{w}_{d_j}}{||\mathbf{w}_{d_i}|| \cdot ||\mathbf{w}_{d_j}||} = \frac{\sum_{t_k \in \mathcal{T}} w_{d_i,t_k} \cdot w_{d_j,t_k}}{\sqrt{\sum_{t_k \in \mathcal{T}} (w_{d_i,t_k})^2} \cdot \sqrt{\sum_{t_k \in \mathcal{T}} (w_{d_j,t_k})^2}}. \tag{16}$$

2.3.2. Inductive Supervised Learning Considering Texts Represented by Networks

In the following subsection, we present divided into the three network types: (i) term networks, (ii) document networks, and (iii) bipartite networks.

There are three categories of algorithms for text classification based on term-networks: (i) similarity-based , (ii) prototype-based, or (iii) feature extraction-based. In similarity-based classification, a term network is generated for each document, and a k-Nearest Neighbor strategy [116] can be used to classify new documents. The key in this approach is computing the similarity among the term networks. This can be performed through Common Subgraphs [102], or the use of similarity kernels [75].

Maximum Common Subgraphs (MCS) computes the similarity between a term network N_i and a term network N_j in the following way:

$$MCS(N_i, N_j) = 1 - \frac{|MCS(N_i, N_j)|}{max(|N_i|, |N_j|)} \tag{17}$$

where $|MCS(N_i, N_j)|$ returns the size of the maximum common subgraph between the term networks N_i and N_j.

In [75], the authors proposed a kernel function to compute the similarity among term networks. The kernel function is given by:

$$K(N_i, N_j) = K_O(N_i, N_j) + K_R(N_i, N_j), \tag{18}$$

where $K_O(N_i, N_j)$ is the kernel function to compute similarity considering the objects of the term networks N_i and N_j, and $K_R(N_i, N_j)$ is the kernel function to compute the similarity among the relations of the networks N_i and N_j. $K_O(G_i, G_j)$ is computed by:

$$K_O(N_i, N_j) = \sum_{o_k \in \mathcal{O}_i} \sum_{o_l \in \mathcal{O}_j} \frac{I(o_i, o_j) \cdot w_{t_k,d_i}}{w_{t_k,d_i} + w_{t_l,d_j} - I(o_i, o_j) \cdot w_{t_k,d_i}}, \tag{19}$$

in which $I(o_i, o_j)$ returns 1 if $o_i = o_j$ and 0 otherwise. The similarity among edges ($K_R(N_i, N_j)$) is computed by:

$$K_R(N_i, N_j) = \sum_{e_{o_k,o_l} \in \mathcal{R}_i} \sum_{e_{o_m,o_n} \in \mathcal{R}_j} \frac{I(e_{o_k,o_l}, e_{o_m,o_n}) \cdot w_{t_k,d_i}}{w_{t_k,d_i} + w_{t_m,d_j} - I(e_{o_k,o_l}, e_{o_m,o_n}) \cdot w_{t_k,d_i}}, \quad (20)$$

in which $\mathcal{R}_i$ and $\mathcal{R}_j$ are the relation sets of the networks N_i and N_j respectively, $I(e_{o_k,o_l}, e_{o_m,o_n})$ returns 1 if $o_k = o_m$ and $o_l = o_n$, and 0 otherwise. The computed kernel are stored in a kernel matrix and allow the use of kNN or SVM to perform automatic text classification.

The frequent subgraph extraction can be used to representative graphs from classes. [23] and [1] use the Subdue algorithm [28] to extract the frequent subgraphs [23, 1]. Operations such as insertion, removal are also considered to compute subgraphs. However, a cost is assigned to each operation, and a limit is established to consider that two networks are isomorphic. Then, a new document is classified according to the class with the most similar sub-graphs.

Instead of compute similarities among graphs, term networks can also be use to generate class prototypes. In [121] is proposed the Term Graph Model (TGM) algorithm. In this algorithm, term networks are generated for each class, in which two terms are connected if they occur above a user threshold. Then the weighted version of PageRank algorithm is used to set the class information of terms [74]. Thus, the class information of a term t_i, in a term network of a class c_j is given by:

$$f_{t_i,c_j}^{(s+1)} = (1 - \lambda) + \lambda \cdot \frac{\sum_{t_k \in \mathcal{T}} w_{t_k,t_i} \cdot f_{t_k,c_j}^{(s)}}{\sum_{t_k \in \mathcal{T}} w_{t_k,t_i}}, \quad (21)$$

in which $0 < \lambda < 1$ is a damping factor, and s in the iteration index of the PageRank algorithm. Equation 21 is applied iteratively until a fixed number of iterations or until the difference between the class information of all terms in two consecutive iterations is above a threshold.

For each network, terms are ranked according to their class information. The ranking position of the terms for each class are used as class prototypes. The terms of a new document are ranked considering their frequency, and this ranking is compared with the class prototypes using Spearman's correlation. Since the number of terms in a document can be different than the number of terms in a class prototype, different approaches are proposed in [121] to allow computing the rank correlation.

2.3.3. Classification based on Document Networks

In [12] is proposed the k-Associated Classifier (KAC), which allow to induce a classification model based on a k-Associated Graph (see Section 2.2.2). Since each component of a k-Associated Graph is associated with a class of the text collection, the classification procedure assigns probabilities to a new document belongs to a class based on the probability of a document belongs to all compo-

nents associated with such class. Thus, the probability of a document d_i belongs to a class c_j is

$$f_{d_i,c_j} = P(c_j|o_i) = \sum_{N_l \in N, class(N_l)=c_j} P(o_i \in N_l|\mathcal{N}^{o_i}), \tag{22}$$

in which $class(N_l)$ returns to class of the component N_l, and $\mathcal{N}^{o_i}$ are the nearest neighbors of an object o_i (or document d_i). The probabilities are directly related to the class of the neighbors and to the density of the neighbor's components. The number of nearest neighbors in k-AG can be manually or automatically specified [12].

2.3.4. Classification based on Bipartite Networks

In [95, 99] is presented the Inductive Model based on Bipartite Heterogeneous Network (IMBHN) algorithm. IMBHN induces a classification model through the structure of a bipartite heterogeneous network. The classification model consists of the class information of terms for classes, which are induced through the minimization of the following function:

$$Q(\mathbf{F}(\mathcal{T})) = \frac{1}{2}\left(\sum_{c_j \in \mathcal{C}} \sum_{d_k \in \mathcal{D}} \left(\lambda\Big(\sum_{t_i \in \mathcal{T}} w_{d_k,t_i} \cdot f_{t_i,c_j}\Big) - y_{d_k,c_j}\right)\right)^2 \tag{23}$$

The function $\lambda(...)$ returns the value 1 for the corresponding class to the $\arg\max_{c_j \in \mathcal{C}} \sum_{t_i \in \mathcal{T}} w_{d_k,t_i} \cdot f_{t_i,c_j}$ and 0 otherwise. Thus, the class information of terms are induced to minimize the squared sum of the differences between the predicted and real classes of the training documents. In [94] is proposed a version of IMBHN without the λ function. We call this latter by IMBHN$^{\mathrm{R}}$ and the version with λ function by IMBHN$^{\mathrm{C}}$. In both versions, the Least-Mean-Square method [124] is used to induce the class information of terms.

IMBHN$^{\mathrm{C}}$ outperformed the classification performance of traditional and state-of-the-art inductive supervised learning algorithms based on VSM, as presented in [95, 99]. To the best of our knowledge, this is the unique algorithm to perform supervised learning considering texts represented in a bipartite network. However, as IMBHN$^{\mathrm{C}}$ works on a bipartite network, the class information of terms are induced only considering `document-term` relations. In the next section, we proposed an approach to compose `document-term` relations with `term-term` relations to induce the class information of terms, and in Section 4 we demonstrate how the proposed approach can improve the classification performance.

3. Proposed Method: Inductive Model using Heterogeneous Networks

Most of the existing classification algorithms to induce a classification model through supervised learning consider texts represented in vector space model. When considering text represented in a vector space model, the algorithms assumes that documents and (or) terms are independent. When considering texts represented in networks, algorithms do not explore the possibility of different types of relations to generate a network, or when consider so, they are language or domain dependent.