WEIGHT YOUR WORDS: THE EFFECT OF DIFFERENT WEIGHTING SCHEMES ON WORDIFICATION PERFORMANCE

Tatiana Sciammarella

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Gerson Zaverucha

Rio de Janeiro
Março de 2020

WEIGHT YOUR WORDS: THE EFFECT OF DIFFERENT WEIGHTING SCHEMES ON WORDIFICATION PERFORMANCE

Tatiana Sciammarella

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Gerson Zaverucha

Aprovada por: Prof. Gerson Zaverucha
Prof. Aline Marins Paes Carvalho
Prof. Valmir Carneiro Barbosa

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2020

Sciammarella, Tatiana

Meça suas palavras: o efeito de diferentes esquemas de pesagem no desempenho do Wordification/Tatiana Sciammarella. – Rio de Janeiro: UFRJ/COPPE, 2020.

XII, 42 p.: il.; 29, 7cm.

Orientador: Gerson Zaverucha

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2020.

Referências Bibliográficas: p. 36 – 42.

1. Propositionalization. 2. Relational Data Mining. 3. Term Weighting. 4. Wordification. 5. Classification. I. Zaverucha, Gerson. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao meu marido.

Agradecimentos

Agradeço aos meus pais, Elyane e Carmino, e ao meu irmão, Felipe, por todo carinho e apoio em todas as horas. Vocês formam meu pilar fundamental para a vida. Em especial, minha mãe, que me inspira com sua força.

Agradeço também ao meu marido, pelo suporte, conselhos e por ser minha outra grande fonte de inspiração. Tenho sorte de estar rodeada por pessoas tão maravilhosas.

Finalmente agradeço ao professor Gerson e toda a equipe do PESC, que me acolheram e orientaram durante esse período.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MEÇA SUAS PALAVRAS: O EFEITO DE DIFERENTES ESQUEMAS DE PESAGEM NO DESEMPENHO DO WORDIFICATION

Tatiana Sciammarella

Março/2020

Orientador: Gerson Zaverucha

Programa: Engenharia de Sistemas e Computação

Bancos de dados relacionais são amplamente utilizados para armazenar dados reais como operações financeiras e registros médicos. Nesse tipo de estrutura, os dados são representados em tabelas, que são interconectadas por chaves estrangeiras. Para realizar a classificação desses dados, podemos optar por duas abordagens: utilizar um classificador multi-relacional para gerar um modelo diretamente sobre os dados relacionais ou utilizar um método de proposicionalização para transformar o banco de dados em uma tabela única e em seguida, aplicar um classificador propositional padrão. Neste trabalho, focamos em um algoritmo de proposicionalização chamado Wordification. Este algoritmo se destaca por ser simples e rápido comparado com outros métodos. O Wordification constrói atributos, também chamados de witems, a partir do nome da tabela, da coluna e do valor de cada célula. O conjunto de atributos criados para cada registro do banco forma um documento de texto. Cada documento é então convertido em um vetor, em que os witems são os atributos e seus valores são dados por um esquema de pesagem. A implementação original do Wordification permite utilizar apenas os seguintes métodos de pesagem: TF-IDF, o TF e o binário. No entanto, diversos trabalhos na área de classificação de texto e mineração de dados tem mostrado que a escolha do método de pesagem pode influenciar bastante o desempenho da classificação. Por esse motivo, nós avaliamos o desempenho do Wordification associado a outros métodos de pesagem que se mostraram estatisticamente melhores que o TF-IDF nas áreas de classificação de textos e recuperação de informações. Os resultados deste trabalho mostram que é possível melhorar o desempenho da classificação com a combinação certa do esquema de pesagem e do tipo de classificador.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

WEIGHT YOUR WORDS: THE EFFECT OF DIFFERENT WEIGHTING SCHEMES ON WORDIFICATION PERFORMANCE

Tatiana Sciammarella

March/2020

Advisor: Gerson Zaverucha

Department: Systems Engineering and Computer Science

Relational databases are commonly used to organize and store real-world data such as financial transactions and medical records. It consists of multiple relations (tables), which are interconnected through foreign key joins. When it comes to classification, there are mainly two options: apply a multi-relational learner to discover patterns across the inter-connected tables or use a propositionalization technique to transform the relational database into a single-table representation and then use a standard propositional learner. In this work, we focus on the last approach. We evaluate a fast and simple propositionalization algorithm called Wordification. This algorithm constructs features based on the table name, attribute name and its value. The set of features generated for each instance of the database form a text document. Each document is converted into a vector, where the features are the attributes, and their values are given by a weighting scheme. Originally, the implementation of Wordification only explored the TF-IDF, the term-frequency and the binary weighting schemes. On the other hand, many works in the text classification and data mining fields have shown that the proper choice of weighting schemes can boost classification. For this reason, we evaluate the performance of Wordification with weighting schemes that statistically outperformed TF-IDF. Our results show that is possible to improve the classification performance with the right combination of weighting scheme and classification algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Relational databases are commonly used by companies and research centers to store and organize their data. Usually, it is useful to explore these data as they may contain hidden patterns that can guide decision-making processes [1]. However, analyzing a relational dataset can be hard as many data mining algorithms can only be applied to propositional data [2]. In this case, we can use Inductive Logic Programming (ILP) and Relational Data Mining (RDM) algorithms to induce models or patterns from multi-relational data.

One established approach to ILP and Relational Data Mining (RDM) is called propositionalization [3]. It transforms a relational database into a single attribute-value table representation [4]. As a result, each row of the propositional table corresponds to an example of the original database and is described by a fixed set of attributes and its values [5]. Then, when it comes to classification, we have to follow a two-step process: first, we have to use a propositionalization method to transform the relational data into a single, flat table, and second, we apply a propositional learning algorithm on the transformed data [2].

Another option is to use ILP methods that can directly induce relational models from relational data, such as Aleph [6] and Progol [7]. However, it has been shown that propositionalization can outperform these ILP methods in terms of speed [3], scalability [3, 8], and predictive performance [9, 10]. In this work, we are going to focus on a fast propositionalization method called Wordification [3].

Many propositionalization approaches [11–13] constructs first-order features, which act as binary attributes in the new propositional representation of examples. Wordification, on the other hand, creates much simpler features to achieve greater scalability [3]. In [3], Wordification was compared to state-of-the-art propositionalization algorithms, RSD [13] and RelF [14], and the feature construction mode of Aleph [6]. There, the statistical tests showed that the classification accuracy achieved using Wordification is equivalent to the other approaches while presenting significant run-time reduction [3].

Wordification creates features concatenating the table name, attribute name (column name), and its discrete (or discretized) value [3]. These features are called word-items, witems, or simply words. The process starts by selecting the main table of the database, which contains an attribute that identifies the class of each row (also called entry, instance, or an individual example) of this table. Then, the algorithm generates the word-items for each entry of the main table and then for the examples of the related tables. At the end of this step, the generated words are joined together, which results in a text document for each instance of the main table. In the following step, the documents are represented as vectors with the size of the vocabulary of the corpus (collection) of documents. Then, the value of each word in a vector is given by a term weighting scheme, which assigns an appropriate value to a word according to its importance in a document. In case a word w is not present in a document d its value is set to zero.

The original Wordification paper allowed the use of three schemes: the term frequency-inverse document frequency (TF-IDF), the term frequency (TF), and the binary scheme [3]. The TF-IDF value of a word w in a document d increases if the word is frequent in that document and not frequent in the entire corpus of documents. The TF value is given by the count of a word w in a document d and the binary scheme (1/0) just indicates the presence/absence of a word w in a document d. The use of a term weighting scheme is important to enhance the performance of classification [15]. After weighting the features, the set of vectors form the final propositional table, which can be given as input to a propositional classifier.

The transformation of a database into a set of text documents results in loss of information [3]. As a way to overcome this loss, the Wordification paper proposes an extension of the document construction step by also concatenating to each document n-grams of word-items. The n-gram construction takes every combination of length k of word-items, where $1 \leq k \leq n$, from the set of all word-items corresponding to a given individual, and concatenates them [3]. However, setting a higher n-gram value also exponentially increases the number of witems in a document. To solve this problem, the Wordification paper suggests pruning words that occur in less than a predefined percentage of documents [3].

1.1 Motivation

The original implementation of Wordification allowed only the use of conventional weighting schemes. As the weighting schemes used did perform significantly differently on the classification task, the Wordification paper published the result for the TF-IDF scheme as it is prevalent in text mining applications [3]. However,

works in the text classification and data mining fields show that the proper choice of weighting schemes can boost classification and many studies propose weighting schemes that statistically outperforms TF-IDF [16–20]. For this reason, we decided to empirically experiment different term-weighting approaches with Wordification in order to improve the classification performance.

We chose three weighting schemes which performed better than TF-IDF in information retrieval [16, 17] and text classification scenarios [18–20]: the BM25, the DELTA-TF-IDF, and TF-RF. As [21] and [22] indicate that different classifiers may work best with different types of weighting schemes, we test the effect of the weighting schemes on four learners: Decision Tree, Random Forest, SVM, and KNN. Last, but not least, we extend the analysis of [3] regarding the impact of the n-gram setting and the pruning of the words generated by Wordification on the final classification performance.

1.2 Contributions

In summary, the main contributions of this dissertation include:

- We show that the right combination of the weighting scheme and classification algorithm can significantly improve classification performance and also highlight which weighting scheme resulted in the best performance for each classifier. We use statistical tests to check whether the improvement is significant or not.
- We show that there is not a single weighting scheme that performs better for every classifier. On the other hand, for most of our cases, DELTA-TF-IDF outperformed TF-IDF. For this reason, we recommend the DELTA-TF-IDF weighting scheme to weight the Wordification features and the SVM with the linear kernel to classify the instances. This combination achieved the best results among all configurations.
- We show that Random Forest is less sensitive to the choice of n-gram, weighting scheme, and pruning threshold than other classifiers.
- Last, but not least, we show that the choice of n-gram is more relevant to classification performance than the use of pruning.

1.3 Outline

The remainder of this dissertation is organized as follows. In Chapter 2 we introduce the main concepts that are relevant to understand this work. First, we

review the fundamentals of relational datasets and propositionalization. Then, we provide a further explanation of the Wordification method and the term-weighting schemes. Lastly, we present the statistical tests that are used to properly compare the performance of different configurations.

In Chapter 3 we introduce the datasets, platform, and classifiers used in our experiments. Next, we present the experimental results and discuss the outcomes. Finally, we present our conclusions in Chapter 4.

Chapter 2

Background

In this chapter, we start by explaining the characteristics of relational datasets. Then, we provide a deeper insight into propositionalization and the Wordification algorithm. Lastly, we cover the weighting schemes and the statistical tests used in this work.

2.1 Relational dataset

Relational datasets describe individuals (entities) by both their own attributes and by their relations to other individuals [23]. This kind of structure can be found in databases, both relational and object-oriented, knowledge bases, or software models, e.g., UML (unified modeling language) class diagrams. In this work, we focus on relational databases.

In a relational database, each table, also called a relation, contains one or more data categories in columns, the attributes. One of the columns acts as a set of primary keys, which uniquely identifies each row/record of a table. The relationship between tables can then be set via the use of foreign keys, a field in a table that links to the primary key of another table. Formally, we can describe a relational database as a set of relations $\{R_1, ..., R_n\}$ and a set of foreign-key connections between the relations denoted by $R_i \rightarrow R_j$ [3].

The structure of a relational dataset can be expressed as an ER (entity-relationship) diagram [3]. Fig. 2.1 shows the ER diagram of the Trains dataset [24] as an example. The boxes in the ER diagram indicate entities, which are individuals or parts of individuals. In our diagram, the Train entity is the individual, each Car is part of a train. The ovals denote attributes of entities and the diamonds indicate relationships between entities.

A relationship between entities can be of three types: one-to-one, one-to-many, and many-to-many [25]. In a one-to-one relationship, each record in one table has at most one related record in another table. In a one-to-many relationship, a record

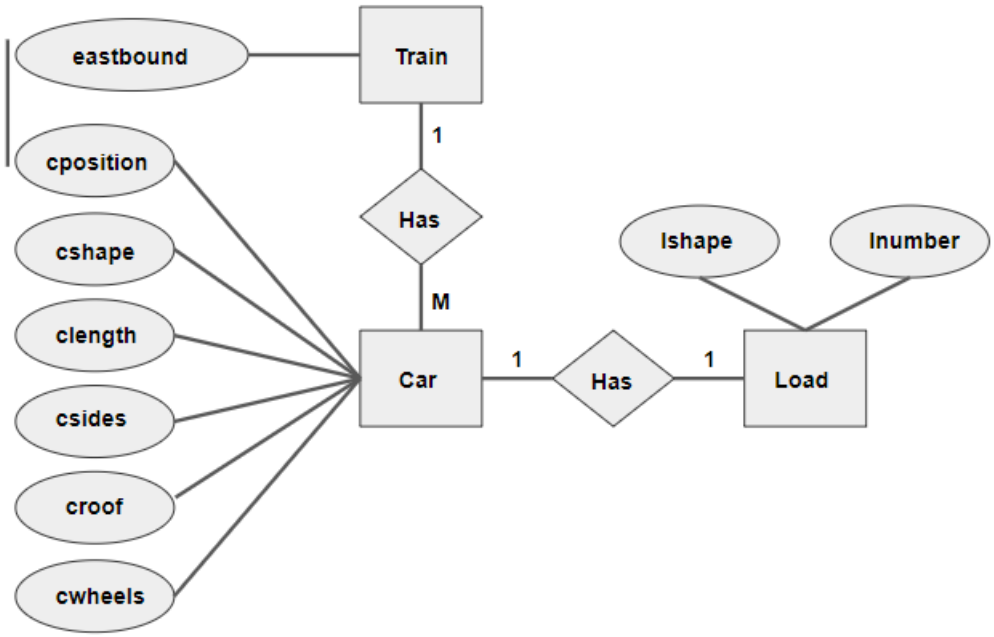


Figure 2.1: Entity-relationship diagram for the Trains dataset [24].

in Table A can have many matching records in Table B, but a record in Table B has only one matching record in Table A. A many-to-many relationship means that for each record in one table there can be many records in another table, and for each record in the second table there can be many in the first. The latter type can be transformed into two or more one-to-many relationships. In our diagram (Fig. 2.1), there is a one-to-many relationship from Train to Car, which means that each train can have an arbitrary number of cars but each car is contained in exactly one train.

When a data structure has only one-to-one relationships, its entities can be combined in a single table. In other words, it becomes a propositional dataset and can be used as input for propositional learners. On the other hand, entities linked by a one-to-many relationship cannot be combined without either introducing significant redundancy or significant loss of information, e.g., introduced through aggregate attributes. For this reason, when a dataset has one-to-many relationships, relational learners and Inductive Logic Programming are used instead of propositional learners [3].

2.2 Propositionalization

Propositionalization is defined as the process of explicitly transforming a relational dataset into a propositional one [5]. The output is a single table, where a row is described by a fixed set of attributes and its respective values. This kind of transformation can only be applied to the so-called individual-centered relational databases, i.e., databases that have a clear notion of an individual example [26]. As

a result, each row of the propositional table corresponds to an individual example of the original relational dataset [3].

The aim of propositionalization is to pre-process relational data for subsequent analysis by propositional (feature-based, attribute-value) learning algorithms instead of relational ones [10]. It is advantageous for three main reasons: (1) it reduces the complexity and speeds up learning; (2) it separates modeling the data from hypothesis construction; and (3) it allows the usage of a wider range of algorithms [5, 10, 12].

Propositionalization techniques generate features that capture the relational properties of the learning examples [10]. They can be divided into two main groups: logic-oriented and database-oriented techniques [12]. The logic-oriented techniques only generate features that can be defined in pure logic, while database-oriented methods include features based on, e.g., aggregation. Each group has its own set of advantages. For instance, logic-oriented methods can handle complex background knowledge and provide expressive first-order models, while database-oriented methods can be more efficient especially on larger datasets. The RSD [13] and RelF [14] methods are logic-oriented, while Wordification is a database-oriented method [3].

2.2.1 Wordification

Logic-oriented propositionalization algorithms construct complex first-order features [13, 14, 27, 28], which act as binary attributes of the new propositional dataset. Compared to these methods, the Wordification algorithm proposed in [3] generates much simpler features to achieve greater scalability.

The process starts when the user selects a target (main) table from the database. This table must contain the column with the class labels that will be used later in the classification step. Then, the Wordification algorithm transforms the relational database into a *corpus* of text documents, where each document represents an instance of the database and each word corresponds to a feature. These features are also called word-items, witems, or words. They are formed by the combination of the table name, attribute name, and its discrete (or discretized) value as shown below:

$$[table\ name]\_[attribute\ name]\_[value]$$

Fig.2.2 shows the documents generated for two instances of the *Trains* dataset [24] presented in Table2.1. The instance with trainID 1 is related to four cars (carID 1,2,3 and 4), while the instance with trainID 15 is only related to two cars (carID 48,49). For this reason, more features are generated for the first instance. As result, the first document showed in Fig.2.2 is longer. In this document, for

Table 2.1: Example of two related tables of the *Trains* dataset [24] before Wordification. Table (a) is the target table, which is connected to table (b) by a one-to-many relationship.

trainID	direction
1	east
...	...
15	west
...	...

(a) Trains table

carID	train	position	shape	len	sides	roof	wheels	load_shape	load_num
1	1	1	rectangle	short	not_double	none	2	circle	1
2	1	2	rectangle	long	not_double	none	3	hexagon	1
3	1	3	rectangle	short	not_double	peaked	2	triangle	1
4	1	4	rectangle	long	not_double	none	2	rectangle	3
...	...	...	...	...	...	...	...	...	...
48	15	1	rectangle	long	not_double	none	2	rectangle	2
49	15	2	u_shaped	short	not_double	none	2	rectangle	1
...	...	...	...	...	...	...	...	...	...

(b) Cars table

!1 Cars_Position_1, Cars_Shape_Rectangle, Cars_Len_Short, Cars_Sides_NotDouble, Cars_Roof_None Cars_Wheels_2, Cars_LoadShape_Circle, Cars_LoadNum_1, Cars_Position_2, Cars_Shape_Rectangle, Cars_Len_Long, Cars_Sides_NotDouble, Cars_Roof_None, Cars_Wheels_3, Cars_LoadShape_Hexagon, Cars_LoadNum_1, Cars_Position_3, Cars_Shape_Rectangle, Cars_Len_Short, Cars_Sides_NotDouble, Cars_Roof_Peaked, Cars_Wheels_2, Cars_LoadShape_Triangle, Cars_LoadNum_1, Cars_Position_4, Cars_Shape_Rectangle, Cars_Len_Long, Cars_Sides_NotDouble, Cars_Roof_None, Cars_Wheels_2, Cars_LoadShape_Rectangle, Cars_LoadNum_3

!0 Cars_Position_1, Cars_Shape_Rectangle, Cars_Len_Long, Cars_Sides_NotDouble, Cars_Roof_None, Cars_Wheels_2, Cars_LoadShape_Rectangle, Cars_LoadNum_2, Cars_Position_2, Cars_Shape_UShaped, Cars_Len_Short, Cars_Sides_NotDouble, Cars_Roof_None, Cars_Wheels_2, Cars_LoadShape_Rectangle, Cars_LoadNum_1

Figure 2.2: Example of generated features for each instance of the *Trains* dataset presented in Table 2.1. The words are the Wordification features and the !1 and !0 at the beginning of each document represents the class positive (east) and negative (west) respectively.

example, the feature *Cars_ Position_ 1* is created from the join of attribute *Position* with value equal to *1* of table *CAR*.

The features of a document are first generated for the target table and then for each entry from the related tables. Each text document can also be represented as Bag-Of-Words (BOW) vectors as illustrated in Fig.2.3, where the size of the vector is the size of the vocabulary of a document and the values correspond to the multiplicity of each word (feature) in the document.

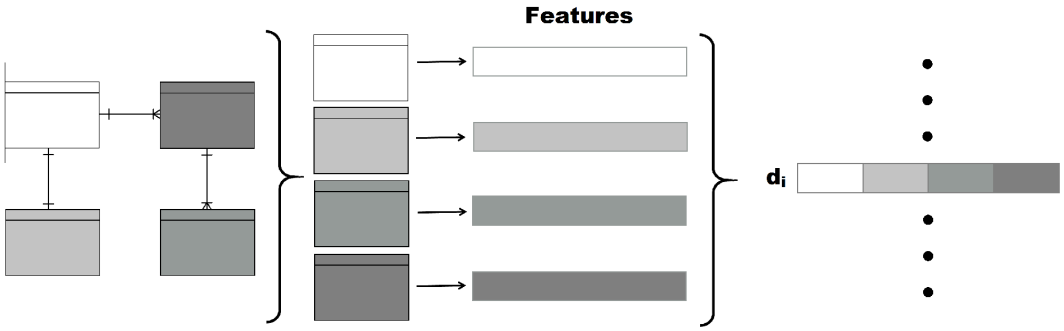


Figure 2.3: The Wordification process of creating a vector representation of a document. The upper-left table is the target table, which is related to the other tables. The features (words) of a document are first generated for the i-th entry of the target table and then for the related entries from the other tables. The generated features can be represented as a Bag-Of-Words (BOW) vector d_i. Each vector has the size of the vocabulary of the document and the values of each feature correspond to the count of each word in the document. This vector corresponds to an intermediate state of the algorithm. Modified from [3].

The downside of this process is that there is some loss of information as a consequence of building a document for each instance by concatenating all word-items from multiple instances of the connected tables [3]. To minimize the loss, the Wordification paper explores the use of n-grams of word-items to model feature dependencies. The n-gram construction takes every combination of k word-items from a set of all word-items corresponding to a given individual and concatenates them using the "__" symbol as follows:

$$[witem_1]\_\_[witem_2]\_\_...\_\_[witem_k]$$

where $1 \leq k \leq n$. These structures represent conjunctions of features occurring together in individual instances (rows of joined tables). Then, the documents for each instance are extended by adding the n-gram of word-items. This solution, however, increases drastically the number of words for each document. For this reason, the paper [3] suggests the use of pruning to limit the number of features, meaning that words that occur in less than a predefined percentage of documents are removed. According to [3], the use of n-grams and pruning is optional.

Table 2.2: Example of the propositional table for the Trains dataset. The value of each feature is given by a weighting scheme, to be seen in Section 2.3.

Train	Cars_Position_1	Cars_LoadNum_3	Cars_Len_Long	...	Class
1	0	1.05	0.32	...	1
...	...	...	...	...	...
15	0	0	0.16	...	0
...	...	...	...	...	...

After Wordification generates a text document for each entry of the main table, the documents are transformed into vectors. These vectors have the size of the vocabulary of the *corpus* and the set of words are the attributes. The value of an attribute is given by a weighting scheme. In the case of a word not appearing in a document, the value of the attribute in the corresponding vector is zero. Each vector is a row of the final propositional table, as shown in Table 2.2, and can already be used as input for a propositional learner.

Algorithm 1 *Wordification*(T, p, k)

Input: table T, pruning percentage p, maximal number of witems per word k
Output: corpus of documents D, propositional table R with TF-IDF values

$D \leftarrow []$ // corpus of documents
$W \leftarrow \varnothing$ // vocabulary set
for $ex \in T$ **do**
  $d \leftarrow Wordify(T, p, k)$
  $D \leftarrow D + [d]$
  $W \leftarrow W \cup keys(d)$
end for
$W \leftarrow prune(W, p)$
return D, calculatedTFIDF(D,W)

Algorithm 2 *Wordify*(T, p, k)

Input: table T, example ex from T, maximal number of witems per word k
Output: document word count d

$d \leftarrow \{\}$
for $i \leftarrow 1$ to k **do**
  for $comb \in attrCombs(ex, k)$ **do**
    $d[word(comb)] \leftarrow d[word(comb)] + 1$
  end for
end for

for $secTable \in connectedTables(T)$ **do**
  for $secEx \in secTable$ **do**
    if $primaryKeyValue(ex) = foreignKeyValue(sexEx)$ **then**
      for $(word, count) \in Wordify(secTable, secEx, k)$ **do**
        $d[word] \leftarrow d[word] + count$
      end for
    end if
  end for
end for
return d

The process just described can be broken down into two algorithms according

to [3]. *Algorithm 1* is the main function, *Wordification*(T, p, k). The parameters T, p and k are set by the user, where T is the main table, p is the pruning percentage and k is the maximum number of witems per word (n-gram) [3]. For each example ex of the main table T, *Algorithm 1* calls *Algorithm 2* (Line 4 in *Algorithm 1*), the function *Wordify*(T, ex, k).

First, the *Wordify* function creates the n-gram of witems for the attributes of the main table (Line 2-6 in *Algorithm 2*), and then, for the related tables (Line 7-15 in *Algorithm 2*). It then returns a dictionary d (Line 16 in *Algorithm 2*), where the keys are the witems and the values correspond to the count of these witems for that instance. In this way, d represents a BOW vector for the example ex.

When Wordify returns the dictionary d, *Algorithm 1* appends the list of words to D (Line 5 in *Algorithm 1*), the *corpus* of documents. The W is the set of witems created for the *corpus*, which receives the new witems created for ex (Line 6 in *Algorithm 1*). After running these steps for each example of the main table, the set of words W can be reduced according to parameter p (Line 8 in *Algorithm 1*). The last step consists of the weight of witems in W for each document in D (Line 9 in *Algorithm 1*).

The original Wordification paper [3] included the results for three weighting methods: TF-IDF, term-frequency, and binary weighting scheme. However, as these showed no statistical difference in performance, the paper focused on the TF-IDF. In this work, the influence of other weighting schemes in the final classification performance is evaluated. Section 2.3 presents a detailed description of each weighting scheme used in our experiments, including TF-IDF.

2.2.2 Binary text classification

Wordification transforms a database into a corpus D of text documents. In this work, we focus on binary text classification problems, where the documents are separated into positive and negative classes. Therefore, we can rewrite $|D|$ as $|P| + |N|$, where $|P|$ is the number of documents in the positive class and $|N|$ is the number of documents in the negative class. The number of documents where the word w appears is $|d \in D : w \in d|$ and we can rewrite it as $|P_w| + |N_w|$, where $|P_w|$ and $|N_w|$ are, respectively, the number of documents containing w in the positive and negative classes. These notations will be useful for the following weighting schemes and are summarized in table 2.3.

Table 2.3: Notations

Class	#Documents	#Documents with w	#Documents without w
Positive	$\|P\|$	$\|P_w\|$	$\|P\| - \|P_w\|$
Negative	$\|N\|$	$\|N_w\|$	$\|N\| - \|N_w\|$

2.3 Term Weighting

Different terms (i.e, nouns, verbs, pronouns, etc.) have different importance in a text. For this reason, information retrieval and text categorization use term weighting to improve performance by assigning numerical values to terms representing their importance in a document [29]. Here, we present the TF-IDF and other three term weighting methods that outperformed TF-IDF according to [16, 18–20].

We can break down the term weighting schemes into two groups: unsupervised and supervised [30]. In the first group, we present the TF-IDF and BM25 algorithms, which come from the information retrieval field. They are used to weight the terms and then, a ranking function is computed by summing up the assigned weights for each query [31]. For the supervised group, we selected the DELTA TF-IDF and TF-RF, which were specially designed to be applied in the text classification field [18–20]. Supervised term-weighting schemes use the available class information of each document in their formula to improve the classification performance [32].

2.3.1 TF-IDF

TF-IDF is frequently used as a weighting function in information retrieval searches [33] to reflect how important a word is to a document in a collection or *corpus*. The TF acronym stands for *term frequency* and is simply a count of occurrences of a word w in a document d. If a word appears many times in a document, then that word may be relevant to that document. IDF is short for *inverse document frequency* and measures the weight of a term at the *corpus* level. It diminishes the weight of terms that occur repeatedly in the *corpus* D and increases the weight of terms that occur rarely. The TF-IDF measure is defined as follows:

$$tfidf(w, d) = tf(w, d) \times \log \frac{|D|}{|d \in D : w \in d|} \tag{2.1}$$

where $tf(w, d)$ is the count of occurrences of a word w in a document d, $|D|$ is the total number of documents in the corpus and $|d \in D : w \in d|$ is the number of documents where the word w appears.

2.3.2 BM25

BM25, short for Okapi BM25 [17], is a state-of-the-art term weighting for information retrieval [34]. Its structure is very similar to that of TF-IDF, however, the term-frequency part is nonlinear. This is desirable due to the statistical dependence of term occurrences: the information gained by observing a term for the first time is greater than the information gained when subsequently seeing the same term. The

result is that the term weight saturates after a few occurrences. The IDF calculation is also slightly different as shown below:

$$bm25(w,d) = tf'(w,d) \times idf'(w) \tag{2.2}$$

$$tf'(w,d) = \frac{tf(w,d) \times (k_1+1)}{tf(w,d) + k_1 \times (1 - b + b \times \frac{dl}{avgdl})} \tag{2.3}$$

$$idf'(w) = \frac{|D| - n(w) + 0.5}{n(w) + 0.5} \tag{2.4}$$

where dl is the length of the document d in words, $avgdl$ is the average document length in the collection, and $n(w)$ is equal to $|P_w|+|N_w|$. k_1 and b are free parameters whose values are usually chosen as $k_1 \in [1.2, 2.0]$ and $b \in [0.5, 0.8]$ [31]. In the absence of parameter tuning, $k_1 = 1.2$ and $b = 0.75$ are recommended [35].

2.3.3 DELTA TF-IDF

DELTA TF-IDF is a supervised weighting scheme as it takes into account the class of each document [18]. It assigns the weight of a term in a document by calculating the difference of the TF-IDF scores of a word in the positive and negative training corpus [18] as shown below:

$$delta.tfidf(w,d) = tf(w,d) \times \log_2 \frac{|P|}{(|P_w|+1)} - tf(w,d) \times \log_2 \frac{|N|}{(|N_w|+1)} \tag{2.5}$$

According to [18], the DELTA TF-IDF scheme enhances the importance of words that are unevenly distributed between positive and negative classes and discounts evenly distributed words, for which $|P|/|P_w| = |N|/|N_w|$. If a word is evenly distributed, $delta.tfidf(w,d)$ is equal to zero. On the other hand, the more uneven the distribution, the more important the word should be. This behavior is desirable for sentiment classification tasks [18]. In equation 2.5, a +1 is added to denominators to avoid zero division when a word w is not present in $|P_w|$ or $|N_w|$.

2.3.4 TF-RF

TF-RF uses the same TF factor as TF-IDF, however, it replaces the IDF part. The problem is that the traditional IDF factor was designed to improve the discriminating power of terms for the information retrieval field . TF-RF proposes a new factor to improve the term's discriminating power focused on the text classification field [19].

To illustrate the problem with the traditional IDF, consider the example of Ta-

ble 2.4 from [19]. The terms w_1, w_2 and w_3 share the same IDF, but have different ratios of $|P_w|$ to $|N_w|$. We assume they have the same term frequency (TF) and $|D| = 1000$.

Table 2.4: Example of three terms that share the same IDF.

Term	Sum($\|P_w\|$,$\|N_w\|$)	$\|P_w\|$:$\|N_w\|$	IDF
w_1	100	10:1	$log(\|D\|/100) = 3.32$
w_2	100	1:1	$log(\|D\|/100) = 3.32$
w_3	100	1:10	$log(\|D\|/100) = 3.32$

In this example, it is clear that w_1 and w_3 should have more power to discriminate the documents in the positive and negative categories, respectively, than w_2. For this reason, [19] proposes a new factor RF, the $relevance frequency$, which is defined as follows:

$$rf = log_2(2 + \frac{|P_w|}{|N_w| + 1}) \tag{2.6}$$

The constant 2 is used because the base of the logarithm is 2. The RF factor gives more weight to terms that occur more in the positive documents than in the negative ones. The reason behind this idea is that positive documents belong to one class, while the negative ones may include all documents in the remaining classes in the case of a multi-label classification problem [19]. Then, according to this formula, the weight of w_1 becomes higher than those of w_2 and w_3 since w_1 contributes more to the positive category. In equation 2.6, a +1 is also added to denominator to avoid zero division when a word w is not present in $|N_w|$.

2.4 Statistical Testing

The statistical tests are in accordance with [36] to compare the classification result for different weighting schemes on multiple datasets. The Wilcoxon signed ranks test [37] is applied for comparison of two classifiers and the Friedman test [38] with the corresponding post-hoc tests is suitable for comparison of more classifiers over multiple datasets [36]. Both are simple and robust non-parametric tests for statistical comparisons of classifiers [36].

2.4.1 Wilcoxon signed ranks test

First, to compute the Wilcoxon signed ranks test, we have to calculate the differences d_i in performance of two classifiers on the i-th out of N datasets. Then, ranks are assigned from the lowest to the highest absolute difference, and average ranks are

assigned in case of ties. The sum of ranks for the positive (R^+) and negative (R^-) differences are computed as follow:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.7}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.8}$$

Then, we find T equal to the smaller of the sums, $T = \min(R+, R-)$ and compute the value of z:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1))}} \tag{2.9}$$

With a significance level set to $\alpha = 0.05$, we can reject the null-hypothesis, which states that both algorithms perform equally well if z is smaller than -1.96. Alternatively, we can use a table of critical values for the Wilcoxon test, which contains the critical values for T according to the value of N. This method is illustrated in [36] with the following example.

Table 2.5: Comparison of ROC AUC for classifiers A and B. Each cell represents an average of five-fold cross validation. Adapted from [36].

Dataset	A	B	difference	rank
adult	0.763	0.768	+0.005	3.5
breast cancer	0.599	0.591	-0.008	7
breast cancer wisconsin	0.954	0.971	+0.017	9
cmc	0.628	0.661	+0.033	12
ionosphere	0.882	0.888	+0.006	5
iris	0.936	0.931	-0.005	3.5
liver disorders	0.661	0.668	+0.007	6
lung cancer	0.583	0.583	0.000	1.5
lymphography	0.775	0.838	+0.063	14
mushroom	1.000	1.000	0.000	1.5
primary tumor	0.940	0.962	+0.022	11
rheum	0.619	0.666	+0.047	13
voting	0.972	0.981	+0.009	8
wine	0.957	0.978	+0.021	10

Given 14 datasets of UCI repository [39], we desire to compare two classifiers A and B using the area under the receiver operating characteristic curve (ROC AUC) [40] as the performance metric. The intent is to reject the null-hypothesis that both algorithms perform equally well.

After computing the differences, as shown in Table 2.5, we calculate the sum of ranks for the positive differences $R^+ = 3.5+9+12+5+6+14+11+13+8+10+1.5 =$

93 and the negative differences $R^- = 7 + 3.5 + 1.5 = 12$. Then, instead of directly computing z, we consult the table of exact critical values for the Wilcoxon test, for a confidence level of $\alpha = 0.05$ and $N = 14$ datasets. According to this table, the difference between the classifiers is significant if the smaller of the sums is equal to or less than 21. In our example $R^- = 12$, so we can reject the null-hypothesis.

2.4.2 Friedman test

The Friedman test [38] starts by ranking the algorithms for each dataset, the best performing algorithm getting rank of 1, the second-best rank 2, and so on. In case of ties, average ranks are assigned. Then, it computes the average rank for each algorithm:

$$R_j = \frac{1}{N} \sum_i r_i^j \tag{2.10}$$

where r_i^j is the rank of the j-th of k algorithms on the i-th of N datasets. As the null-hypothesis states that all the algorithms are equivalent, their ranks R_j should be equal. After computing R_j, we can calculate the Friedman statistic as follow:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{2.11}$$

However, Iman and Davenport [41] showed that χ_F^2 is undesirably conservative and derived a better statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2} \tag{2.12}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. To reject the null-hypothesis, we have to search the F-distribution table, with the desired significance level, for the corresponding value for $k-1$ and $(k-1)(N-1)$ degrees of freedom and compare it with the calculated F_F. If the value of F_F is higher, we can reject the null-hypothesis. In our experiments, we apply the Friedman test [38] with significance level $\alpha = 0.05$.

Table 2.6 is an adapted example of [36] to illustrate the calculation of the Friedman statistic. In this example, the intent is to compare the performance of classifiers A, B, C, and D for $N = 14$ datasets of the UCI [39] repository using the ROC AUC metric.

We start ranking the algorithms for each dataset. For instance, in the dataset *adult*, classifier D is ranked first because it achieved the highest score. On the other hand, classifier A has the lowest score, so it is ranked last. After this step, the average rank is computed for each classifier. Then, we can calculate χ_F^2 and F_F as

follow:

$$\chi_F^2 = \frac{12 \times 14}{4 \times 5} \left[(3.143^2 + 2.000^2 + 2.893^2 + 1.964^2) - \frac{4 \times 5^2}{4} \right] = 9.28$$

$$F_F = \frac{13 \times 9.28}{14 \times 3 - 9.28} = 3.69$$

F_F is distributed according to the F-distribution with $k-1 = 3$ and $(k-1)(N-1) = 39$ degrees of freedom. In this case, the critical value of F(3.69) for $\alpha = 0.05$ is 2.85. As $F_F = 3.69$, we can reject the null-hypothesis.

Table 2.6: Comparison of ROC AUC for classifiers A, B, C and D. Each cell results from an average over 5-fold cross validation. Adapted from [36].

Dataset	A	B	C	D
adult	0.763 (4)	0.768 (3)	0.771 (2)	0.798 (1)
breast cancer	0.599 (1)	0.591 (2)	0.590 (3)	0.569 (4)
breast cancer wisconsin	0.954 (4)	0.971 (1)	0.968 (2)	0.967 (3)
cmc	0.628 (4)	0.661 (1)	0.654 (3)	0.657 (2)
ionosphere	0.882 (4)	0.888 (2)	0.886 (3)	0.898 (1)
iris	0.936 (1)	0.931 (2.5)	0.916 (4)	0.931 (2.5)
liver disorders	0.661 (3)	0.668 (2)	0.609 (4)	0.685 (1)
lung cancer	0.583 (2.5)	0.583 (2.5)	0.563 (4)	0.625 (1)
lymphography	0.775 (4)	0.838 (3)	0.866 (2)	0.875 (1)
mushroom	1.000 (2.5)	1.000 (2.5)	1.000 (2.5)	1.000 (2.5)
primary tumor	0.940 (4)	0.962 (2.5)	0.965 (1)	0.962 (2.5)
rheum	0.619 (3)	0.666 (2)	0.614 (4)	0.669 (1)
voting	0.972 (4)	0.981 (1)	0.975 (2)	0.975 (3)
wine	0.957 (3)	0.978 (1)	0.946 (4)	0.970 (2)
average rank	3.143	2.000	2.893	1.964

Whenever the null-hypothesis is rejected, we proceed with a post-hoc test [36]. Here we use the Nemenyi post-hoc test [42] to compare the multiple algorithms to one another.

2.4.3 Nemenyi post-hoc test

The Nemenyi test performs a pairwise test of performance [42]. According to [36], the performance of two classifiers is significantly different if the corresponding average ranks differ by at least a critical difference CD (critical difference), given by:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{2.13}$$

where q_α is based on the Studentized [43] range statistic divided by $\sqrt{2}$. The result of this test can be visualized with *CD* diagrams.

Table 2.7: Critical values for the Nemenyi test. Adapted from [36].

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	**2.291**	2.459	2.589	2.693	2.780	2.855	2.920

For the example of Table 2.6, where $k = 4$, and choosing $\alpha = 0.10$, we find $q_\alpha = 2.291$ according to Table 2.7. Then we can compute *CD* as follow:

$$CD = 2.291\sqrt{\frac{4 \cdot 5}{6 \cdot 14}} = 1.12$$

Fig. 2.4 shows the corresponding critical distance diagram, where groups of classifiers that are not significantly different (for $\alpha = 0.10$) are connected. According to this diagram, the performance of A is significantly worse than that of D for $\alpha = 0.10$.

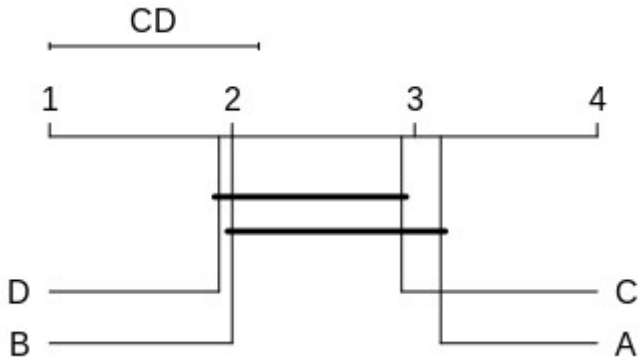


Figure 2.4: Example of critical distance diagram for $\alpha = 0.10$

Chapter 3

Experiments

In this chapter, we present the datasets, platform, and classifiers used in our experiments. Then we present the results and discuss the outcomes.

3.1 Datasets

For this work, we selected 13 relational datasets. The datasets *Trains*, *IMDB*, *Mutagenesis 42*, *Mutagenesis 188*, *Carcinogenesis* and *Financial* were used in the original Wordification paper[1] [3]. We also added to our experiments the *Hepatitis*, *Musk Large*, *Musk Small*, *Pima*, *Toxicology*, *Student Loan* and *NBA* databases[2] [44]. The instances of these datasets must be divided into positive and negative examples to perform binary classification. Each instance is assigned to a positive or negative class according to a convention based on an attribute of the main table. Table 3.1 shows the distribution of instances according to their classes.

Trains: this relational dataset was first used in the East-West Trains challange [24] to predict whether a train was East-bound or West-bound. The original dataset has three tables: Trains, Car, and Loads, where a train can contain a variable number of cars, with different shapes and loads. In this dataset, East-bound Trains are considered positive examples.

IMDB: originally, this is a moderately large, real database of movies. However, we use the same version of the Wordification paper [3], which consists only of movies whose titles and years of production exist on the IMDB top 250 and bottom 100 charts of July 2, 2012. The result is a database with 166 instances, along with all of their actors, genres, and directors. Movies present in the IMDB top 250 charts were considered as positive examples, while those in the bottom 100 were regarded as negative.

[1] Datasets available at http://kt.ijs.si/janez$_k$*ranjc/ilp_datasets/*

[2] Datasets available at https://relational.fit.cvut.cz/

Table 3.1: Class Distributions.

Database	Positive	Negative
IMDB	122	44
Trains	10	10
Mutagenesis 42	13	29
Mutagenesis 188	125	63
Carcinogenesis	182	147
Financial	606	76
Hepatitis	294	206
Musk Large	39	63
Musk Small	47	45
Pima	268	500
Toxicology	152	191
Student Loan	643	357
NBA	15	15

Mutagenesis 42 and *Mutagenesis 188*: these datasets are used to predict the mutagenicity of aromatic and heteroaromatic nitro compounds [45]. The compounds with positive levels of mutagenicity are labeled "active" and are considered positive examples. On the other hand, the compounds labeled "inactive" are negative examples. The original database contains 230 compounds, but the data were split into two subsets: the Mutagenesis 188 with 188 compounds and the Mutagenesis 42, a smaller dataset with 42 compounds [45].

Carcinogenesis: this dataset is used to predict the carcinogenicity of a diverse set of chemical compounds [46]. The compounds classified as carcinogens are positive examples.

Financial: this dataset was artificially constructed as part of the PKDD99 Discovery Challenge to predict successful loans [47]. It has 8 tables with data about clients of a bank, their accounts, transactions, permanent orders, granted loans, and issued credit cards [48]. A successful loan is a positive example.

Hepatitis: this is a modified version of the PKDD02 Discovery Challenge [49] database, which removes tests with null values. It contains basic information about hepatitis B and C infected patients and their laboratory test results. Here, patients with Hepatitis C are considered positive examples.

Musk Large and *Musk Small*: these datasets contain two tables, the molecules, and conformations, which are joined by a one-to-many association [50]. The conformations of a molecule determine if it is a musk or not. The instances classified as musk are positive examples. There are two versions of the dataset, MuskSmall, containing 92 molecules and 476 confirmations, and MuskLarge, containing 102 molecules and 6598 confirmations.

Pima: This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases, which conducted a study on 768 adult Pima Indians, Native Americans who live around Arizona [51]. The aim is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Patients with diabetes are considered positive examples.

Toxicology: The Predictive Toxicology Challenge (2000) dataset [52] consists of more than 300 organic molecules marked according to their carcinogenicity on mice and rats. The challenge aimed to predict the carcinogenicity of chemicals based on the molecule structure only. Carcinogenic molecules are positive examples.

Student Loan: This dataset contains data about student enrollment and employment status [53]. The students that are not obligated to pay their loans back are the positive examples.

NBA: This dataset contains data about basketball matches from the National Basketball Association. For each player, there are 19 continuous player statistics recorded, such as the number of free throws attempts and the total number of player rebounds [54]. The games where team 1 won are the positive examples.

3.2 Clowdflow

Clowdflow [55] is an open-source, web-based data mining platform. It offers many widgets to build data mining workflows, including a widget with the Wordification implementation. This widget is used in our workflow, as shown in Fig. 3.1, to generate a corpus of documents for each database.

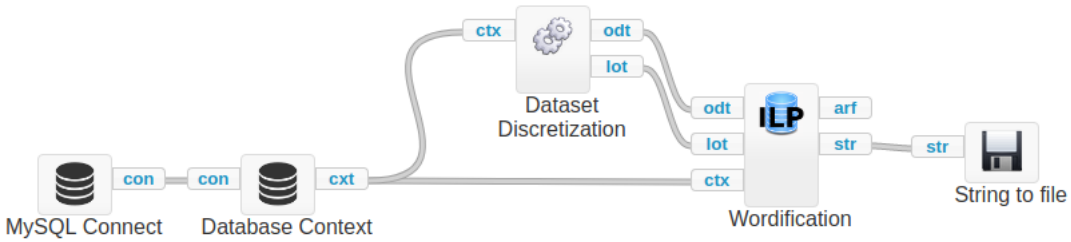


Figure 3.1: Wordification workflow on Clowdflow.

The process starts when we use the *Database Connect* widget to access a database on a MySQL [56] database server. Then, the *Database Context* widget is used to select the target table, the related tables, and the attribute used to separate the instances into classes. After that, the *Dataset Discretization* widget is used to discretize the continuous attributes of the selected tables. It supports three discretization methods: equal-width interval, equal-frequency intervals, and class-aware discretization [57]. We use the equi-width discretization, the same used in the

Wordification paper [3]. Finally, the Wordification widget takes as input the target table, the list of additional tables, and the database context with information about the relation between tables, generating a document for each instance of a database.

Each document shows the class of the instance and a list of features, as shown in Fig. 2.2, generated based on the database schema and the values of the attributes for that instance. At the end of the workflow, the corpus of documents for each dataset is saved to a text file.

3.3 Propositional table

In possession of the text file, we transform the list of features of each document into a vector, where the words are the attributes and the values are given by a term-weighting scheme (TF-IDF, BM25, TF-RF, or DELTA-TF-IDF). After that, we create a propositional table, where each row is a vector that represents an instance of a database and each column is a word of the Wordification algorithm as shown in Table 2.2.

3.4 Classifiers

The resulting table can be used as input to our classifiers, for which we apply the 5-fold cross-validation to estimate the ROC AUC and *accuracy*. We chose the ROC AUC metric because some of our databases are imbalanced, as shown in Table 3.1, but we also present the accuracy score. We repeat this process for each combination of dataset, weighting scheme, and classifier.

We use the Random Forest, Decision Tree, SVM with linear kernel and KNN classifiers set to the default configurations from the library scikit-learn [58]. Random Forest and SVM are known to have the best performance among the classifiers [59, 60]. SVM and KNN are also frequently used in works of text categorization [19, 32, 61]. Besides, Decision Tree and SVM were the two learners used in the Wordification paper [3].

3.5 Results

In this section, we present the results from the 5-fold cross-validation applied for each combination of database, weighting scheme, and classifier. Based on these results, we apply the statistical tests to compare the performance of these configurations. We also evaluate the use of n-gram and feature filtering.

3.5.1 ROC AUC score

For each combination of dataset and classifier, we performed 5-fold cross-validation measuring the ROC AUC score. Tables 3.2, 3.3, 3.4 and 3.5 present the average scores for each database for Decision Tree, Random Forest, SVM and KNN respectively. Each row also shows the weighting schemes (TF-IDF, BM25, DELTA TF-IDF, and TF-RF) ranked according to the Friedman test. The best score for each database is highlighted and ranks are in brackets.

We summarize in Table 3.6 the occurrences of best scores for each combination of classifier and weighting scheme. From this table, we can see that DELTA TF-IDF achieved the highest scores in 9 of 13 databases for Decision Tree, SVM, and KNN, while TF-RF scored better in 6 cases for Random Forest.

Table 3.2: Average ROC AUC scores for Decision Tree.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	**0.61** (1.0)	0.60 (2.0)	0.50 (3.5)	0.50 (3.5)
Trains	**0.95** (2.0)	**0.95** (2.0)	**0.95** (2.0)	0.85 (4.0)
Mutagenesis 188	**0.93** (1.5)	0.92 (3.5)	0.92 (3.5)	**0.93** (1.5)
Mutagenesis 42	**0.95** (2.0)	0.93 (4.0)	**0.95** (2.0)	**0.95** (2.0)
Carcinogenesis	0.50 (4.0)	0.52 (3.0)	**0.57** (1.0)	0.53 (2.0)
Financial	**0.61** (2.5)	**0.61** (2.5)	**0.61** (2.5)	**0.61** (2.5)
Hepatitis	**0.66** (2.5)	**0.66** (2.5)	**0.66** (2.5)	**0.66** (2.5)
Musk Large	0.57 (3.0)	**0.63** (1.0)	0.55 (4.0)	0.59 (2.0)
Musk Small	0.59 (3.0)	0.59 (3.0)	0.59 (3.0)	**0.63** (1.0)
Pima	0.52 (3.5)	0.52 (3.5)	**0.53** (1.5)	**0.53** (1.5)
Toxicology	**0.55** (2.0)	0.54 (4.0)	**0.55** (2.0	**0.55** (2.0)
Student Loan	0.65 (4.0)	0.67 (2.5)	**0.68** (1.0)	0.67 (2.5)
NBA	0.40 (3.0)	**0.45** (1.5)	**0.45** (1.5)	0.35 (4.0)

Table 3.3: Average ROC AUC scores for Random Forest.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	**0.58** (1.5)	**0.58** (1.5)	0.46 (4.0)	0.54 (3.0)
Trains	0.75 (4.0)	**0.90** (1.5)	0.80 (3.0)	**0.90** (1.5)
Mutagenesis 188	0.97 (2.5)	**0.98** (1.0)	0.96 (4.0)	0.97 (2.5)
Mutagenesis 42	0.87 (4.0)	**1.00** (1.0)	0.95 (2.0)	0.92 (3.0)
Carcinogenesis	0.53 (4.0)	0.59 (2.5)	0.59 (2.5)	**0.61** (1.0)
Financial	0.60 (3.5)	**0.62** (1.0)	0.60 (3.5)	0.61 (2.0)
Hepatitis	**0.66** (2.0)	0.65 (4.0)	0.68 (2.0)	**0.66** (2.0)
Musk Large	0.68 (3.5)	0.68 (3.5)	**0.75** (1.0)	0.71 (2.0)
Musk Small	0.63 (3.0)	0.69 (2.0)	0.61 (4.0)	**0.71** (1.0)
Pima	0.64 (3.0)	0.62 (4.0)	0.65 (2.0)	**0.67** (1.0)
Toxicology	0.63 (2.0)	0.60 (4.0)	0.62 (3.0)	**0.64** (1.0)
Student Loan	0.65 (4.0)	0.66 (3.0)	**0.68** (1.0)	0.67 (2.0)
NBA	**0.63** (1.0)	0.45 (3.5)	0.60 (2.0)	0.45 (3.5)

Table 3.4: Average ROC AUC scores for SVM.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	**0.65** (2.5)	**0.65** (2.5)	**0.65** (2.5)	**0.65** (2.5)
Trains	0.40 (4.0)	0.80 (2.5)	**0.90** (1.0)	0.80 (2.5)
Mutagenesis 188	0.93 (4.0)	0.95 (3.0)	0.96 (2.0)	**0.97** (1.0)
Mutagenesis 42	0.95 (2.5)	0.95 (2.5)	**1.00** (1.0)	0.93 (4.0)
Carcinogenesis	0.60 (4.0)	0.61 (2.5)	**0.62** (1.0)	0.61 (2.5)
Financial	**0.61** (1.0)	0.45 (3.0)	0.54 (2.0)	0.44 (4.0)
Hepatitis	**0.61** (1.5)	0.56 (3.5)	**0.61** (1.5)	0.56 (3.5)
Musk Large	0.76 (3.5)	0.81 (2.0)	**0.92** (1.0)	0.76 (3.5)
Musk Small	0.78 (4.0)	0.81 (2.0)	**1.00** (1.0)	0.79 (3.0)
Pima	0.67 (3.5)	0.67 (3.5)	**0.79** (1.0)	0.68 (2.0)
Toxicology	0.53 (3.0)	**0.60** (1.0)	0.54 (2.0)	0.51 (4.0)
Student Loan	0.67 (3.0)	**0.71** (1.0)	0.67 (3.0)	0.67 (3.0)
NBA	0.60 (2.0)	0.55 (3.5)	**0.80** (1.0)	0.55 (3.5)

Table 3.5: Average ROC AUC scores for KNN.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	0.57 (2.0)	**0.58** (1.0)	0.50 (3.5)	0.50 (3.5)
Trains	0.50 (4.0)	0.90 (2.0)	0.85 (3.0)	**0.95** (1.0)
Mutagenesis 188	0.81 (3.0)	0.80 (4.0)	0.92 (2.0)	**0.93** (1.0)
Mutagenesis 42	0.68 (3.0)	0.54 (4.0)	**0.85** (1.0)	0.80 (2.0)
Carcinogenesis	**0.62** (1.0)	0.61 (2.5)	0.61 (2.5)	0.53 (4.0)
Financial	**0.56** (2.0)	**0.56** (2.0)	**0.56** (2.0)	0.55 (4.0)
Hepatitis	0.59 (2.0)	0.58 (3.5)	**0.63** (1.0)	0.58 (3.5)
Musk Large	0.57 (3.5)	0.57 (3.5)	**0.79** (1.0)	0.60 (2.0)
Musk Small	0.60 (4.0)	0.64 (2.5)	**0.83** (1.0)	0.64 (2.5)
Pima	0.63 (4.0)	0.64 (3.0)	**0.69** (1.0)	0.67 (2.0)
Toxicology	0.56 (4.0)	0.61 (2.0)	**0.63** (1.0)	0.58 (3.0)
Student Loan	0.64 (4.0)	**0.67** (1.5)	**0.67** (1.5)	0.66 (3.0)
NBA	0.63 (3.0)	0.65 (2.0)	**0.75** (1.0)	0.20 (4.0)

Table 3.6: Summary of best ROC AUC scores for each combination of classifier and weighting scheme.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
Decision Tree	7/13	5/13	**9/13**	7/13
Random Forest	3/13	5/13	3/13	**6/13**
SVM	3/13	3/13	**9/13**	2/13
KNN	2/13	3/13	**9/13**	2/13

3.5.1.1 Statistical tests

After ranking each weighting scheme, for each classifier according to the Friedman Test, we present the Table 3.7 with the average ranks. The lower the rank, the better. We can see from these tables that DELTA TF-IDF achieved the best rank in most of

the cases, except for Random Forest. We can also notice that TF-IDF was ranked last for Random Forest, SVM, and KNN. We apply the Friedman test, for each classifier separately according to [36], to check if we can reject the null-hypothesis, which states that all weighting schemes are equivalent, with a significance level of $\alpha = 0.05$. If the null-hypothesis is rejected, we can proceed with a post hoc test, in our case the Nemenyi test.

Table 3.7: Average ROC AUC based on all datasets and average rank according to the Friedman test for each classifier.

Weighting scheme	Average ROC AUC	Average rank
TF-IDF	0.65	2.6
BM25	0.66	2.7
DELTA TF-IDF	0.65	**2.3**
TF-RF	0.64	2.4

(a) Decision Tree

Weighting scheme	Average ROC AUC	Average rank
TF-IDF	0.68	2.9
BM25	0.69	2.5
DELTA TF-IDF	0.69	2.6
TF-RF	0.70	**2.0**

(b) Random Forest

Weighting scheme	Average ROC AUC	Average rank
TF-IDF	0.67	3.0
BM25	0.70	2.5
DELTA TF-IDF	0.77	**1.5**
TF-RF	0.69	3.0

(c) SVM

Weighting scheme	Average ROC AUC	Average rank
TF-IDF	0.61	3.0
BM25	0.64	2.6
DELTA TF-IDF	0.71	**1.7**
TF-RF	0.63	2.7

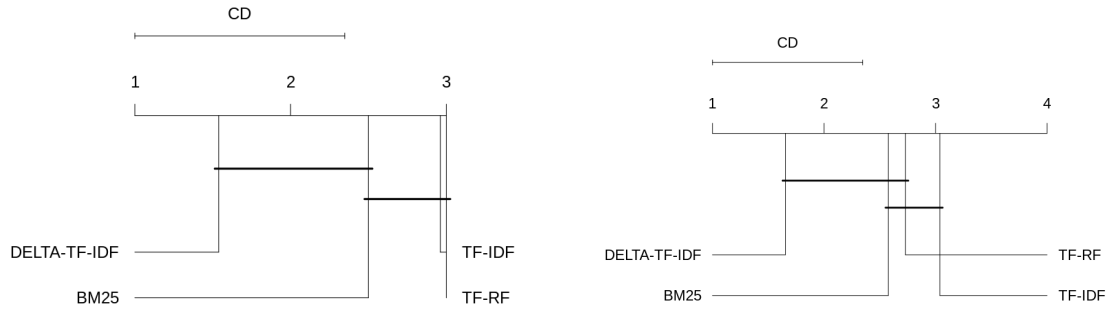
(d) KNN

For the Decision Tree and Random Forest classifiers, the null-hypothesis cannot be rejected (p-value > 0.05; Decision Tree: 0.759; Random Forest: 0.249), which means the results are statistically equivalent even if the weighting scheme is changed. On the other hand, for both the SVM and KNN classifiers, the null-hypothesis was successfully rejected (p-value < 0.05; SVM = 0.004; KNN = 0.029). Then, we can say that the performance of these classifiers varies significantly depending on the choice of weighting scheme.

We then apply the Nemenyi test for SVM and KNN. The results can be visualized

compactly with the critical distance diagram in Fig. 3.2. The diagram interconnects the algorithms in which performance is statistically equivalent. For the SVM classifier, the diagram in Fig. 3.2a shows that the performance using DELTA TF-IDF was significantly better than the performance using TF-IDF or TF-RF. We can also notice from Fig. 3.2b that DELTA-TF-IDF performed significantly better than TF-IDF for KNN.

According to table 3.7, the combination of SVM with the DELTA TF-IDF weighting scheme resulted in the highest average ROC AUC score. On the other hand, in the original Wordification paper, the best-reported results were achieved using the weights from TF-IDF as input to a Decision Tree classifier [3]. Then, we apply the Wilcoxon signed ranks test [37] to compare these two configurations using the AUC as the performance metric. The test shows that the results of SVM with DELTA TF-IDF are significantly better than those of Decision Tree with TF-IDF (p-value $= 0.043$; $\alpha = 0.05$).



(a) SVM (b) KNN

Figure 3.2: Critical distance diagram for the reported (a) SVM and (b) KNN classification AUC.

3.5.2 Accuracy score

We present Tables 3.8, 3.9, 3.10 and 3.11, which show the average accuracy results from the 5-fold cross-validation for each classifier. The best score for each database is highlighted and ranks are in brackets. From Table 3.12, we can notice that all classifiers achieved the highest scores in most of the databases using the DELTA TF-IDF weighting scheme.

3.5.2.1 Statistical tests

Table 3.13 shows the average ranks for each weighting scheme and classifier according to the Friedman test. From these tables, we can see that DELTA TF-IDF achieved the best rank in all cases. On the other hand, the TF-IDF was ranked in the last

Table 3.8: Average accuracy scores for Decision Tree.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	0.57 (3.5)	0.57 (3.5)	**0.74** (1.5)	**0.74** (1.5)
Trains	0.90 (3.0)	0.90 (3.0)	**0.95** (1.0)	0.90 (3.0)
Mutagenesis 188	**0.95** (2.0)	0.94 (4.0)	**0.95** (2.0)	**0.95** (2.0)
Mutagenesis 42	**0.98** (2.0)	0.95 (4.0)	**0.98** (2.0)	**0.98** (2.0)
Carcinogenesis	0.47 (4.0)	**0.56** (1.0)	0.53 (2.5)	0.53 (2.5)
Financial	**0.88** (2.5)	**0.88** (2.5)	**0.88** (2.5)	**0.88** (2.5)
Hepatitis	**0.66** (2.5)	**0.66** (2.5)	**0.66** (2.5)	**0.66** (2.5)
Musk Large	0.55 (3.0)	**0.71** (1.0)	0.58 (2.0)	0.54 (4.0)
Musk Small	0.58 (3.0)	**0.60** (1.0)	0.59 (2.0)	0.54 (4.0)
Pima	0.58 (4.0)	0.60 (2.0)	**0.61** (1.0)	0.59 (3.0)
Toxicology	0.55 (3.0)	**0.56** (1.5)	**0.56** (1.5)	0.54 (4.0)
Student Loan	0.67 (3.5)	**0.69** (1.5)	0.67 (3.5)	**0.69** (1.5)
NBA	0.47 (2.5)	**0.53** (1.0)	0.47 (2.5)	0.38 (4.0)

Table 3.9: Average accuracy scores for Random Forest.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	0.57 (3.5)	0.57 (3.5)	**0.74** (1.5)	**0.74** (1.5)
Trains	0.75 (4.0)	**0.80** (2.0)	**0.80** (2.0)	**0.80** (2.0)
Mutagenesis 188	**0.95** (1.5)	**0.95** (1.5)	0.94 (3.5)	0.94 (3.5)
Mutagenesis 42	**0.86** (2.0)	**0.86** (2.0)	0.84 (4.0)	**0.86** (2.0)
Carcinogenesis	0.58 (2.5)	0.58 (2.5)	**0.59** (1.0)	0.57 (4.0)
Financial	**0.89** (1.0)	0.88 (3.0)	0.88 (3.0)	0.88 (3.0)
Hepatitis	**0.66** (2.5)	**0.66** (2.5)	**0.66** (2.5)	**0.66** (2.5)
Musk Large	0.66 (2.5)	0.66 (2.5)	**0.67** (1.0)	0.65 (4.0)
Musk Small	0.65 (3.5)	0.71 (2.0)	**0.72** (1.0)	0.65 (3.5)
Pima	0.65 (3.0)	**0.67** (1.0)	0.65 (3.0)	0.65 (3.0)
Toxicology	**0.59** (2.0)	0.57 (4.0)	**0.59** (2.0)	**0.59** (2.0)
Student Loan	0.68 (3.0)	0.69 (2.0)	0.67 (4.0)	**0.70** (1.0)
NBA	0.50 (2.0)	0.47 (3.5)	**0.53** (1.0)	0.47 (3.5)

place for all classifiers. We apply the Friedman test to check if we can reject the null-hypothesis, with a significance level of $\alpha = 0.05$.

For Decision Tree, Random Forest and SVM, the null-hypothesis could not be rejected (p-value > 0.05; Decision Tree: 0.076; Random Forest: 0.751; SVM: 0.489). Only for KNN was the null-hypothesis successfully rejected (p-value < 0.05; KNN $= 0.007$). The result of the Nemenyi test for the KNN classifier is shown in Fig. 3.3. Once more the DELTA TF-IDF performed significantly better than TF-IDF for KNN.

We can see from table 3.13, that DELTA TF-IDF with SVM achieved the highest average accuracy. We also compare it with the best configuration of the Wordification paper [3], TF-IDF with a Decision Tree classifier. We use the Wilcoxon signed ranks test using accuracy as the performance metric. The test shows that the results

Table 3.10: Average accuracy scores for SVM.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	**0.74** (2.5)	**0.74** (2.5)	**0.74** (2.5)	**0.74** (2.5)
Trains	0.45 (4.0)	**0.70** (2.0)	**0.70** (2.0)	**0.70** (2.0)
Mutagenesis 188	0.89 (4.0)	**0.95** (1.5)	0.93 (3.0)	**0.95** (1.5)
Mutagenesis 42	0.91 (2.0)	0.88 (3.0)	**0.98** (1.0)	0.83 (4.0)
Carcinogenesis	0.62 (2.5)	0.61 (4.0)	**0.64** (1.0)	0.62 (2.5)
Financial	**0.89** (2.5)	**0.89** (2.5)	**0.89** (2.5)	**0.89** (2.5)
Hepatitis	**0.62** (2.0)	**0.62** (2.0)	0.61 (4.0)	**0.62** (2.0)
Musk Large	0.72 (4.0)	0.77 (2.0)	**0.88** (1.0)	0.73 (3.0)
Musk Small	0.75 (3.0)	0.75 (3.0)	**0.97** (1.0)	0.75 (3.0)
Pima	0.67 (2.5)	0.67 (2.5)	**0.73** (1.0)	0.65 (4.0)
Toxicology	**0.59** (2.0)	**0.59** (2.0)	0.58 (4.0)	**0.59** (2.0)
Student Loan	**0.70** (1.5)	0.69 (3.5)	0.69 (3.5)	**0.70** (1.5)
NBA	0.43 (2.5)	0.43 (2.5)	**0.60** (1.0)	0.40 (4.0)

Table 3.11: Average accuracy scores for KNN.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
IMDB	0.69 (4.0)	0.70 (3.0)	**0.74** (1.5)	**0.74** (1.5)
Trains	0.55 (4.0)	0.75 (2.5)	**0.80** (1.0)	0.75 (2.5)
Mutagenesis 188	0.78 (3.5)	0.78 (3.5)	**0.90** (1.0)	0.87 (2.0)
Mutagenesis 42	0.70 (3.5)	0.70 (3.5)	0.77 (2.0)	**0.81** (1.0)
Carcinogenesis	**0.60** (1.0)	0.58 (2.0)	0.56 (3.0)	0.51 (4.0)
Financial	0.88 (3.0)	0.88 (3.0)	0.88 (1.0)	**0.89** (1.0)
Hepatitis	0.63 (2.5)	0.63 (2.5)	**0.66** (1.0)	0.62 (4.0)
Musk Large	0.56 (4.0)	0.62 (2.5)	**0.70** (1.0)	0.62 (2.5)
Musk Small	0.57 (3.0)	**0.63** (1.0)	0.58 (2.0)	0.52 (4.0)
Pima	0.65 (3.0)	0.66 (2.0)	**0.69** (1.0)	0.46 (4.0)
Toxicology	0.55 (4.0)	0.58 (2.0)	**0.59** (1.0)	0.57 (3.0)
Student Loan	0.65 (4.0)	**0.70** (1.0)	0.68 (2.0)	0.67 (3.0)
NBA	0.57 (2.0)	0.43 (3.0)	**0.60** (1.0)	0.27 (4.0)

Table 3.12: Summary of best accuracy scores for each combination of classifier and weighting scheme.

Database	TF-IDF	BM25	DELTA TF-IDF	TF-RF
Decision Tree	4/13	**8/13**	**8/13**	6/12
Random Forest	5/13	5/13	**8/13**	6/13
SVM	5/13	6/13	**9/13**	7/13
KNN	1/13	2/13	**8/13**	3/13

of SVM with DELTA TF-IDF are significantly better than the Decision Tree with TF-IDF (p-value = 0.049; $\alpha = 0.05$).

Table 3.13: Average accuracy based on all datasets and average rank according to the Friedman test for each classifier based on accuracy scores.

Weighting scheme	Average accuracy	Average rank
TF-IDF	0.68	3.0
BM25	0.70	2.2
DELTA TF-IDF	0.70	**2.0**
TF-RF	0.69	2.8

(a) Decision Tree

Weighting scheme	Average accuracy	Average rank
TF-IDF	0.69	2.5
BM25	0.70	2.5
DELTA TF-IDF	0.71	**2.3**
TF-RF	0.70	2.7

(b) Random Forest

Weighting scheme	Average accuracy	Average rank
TF-IDF	0.69	2.7
BM25	0.71	2.5
DELTA TF-IDF	0.76	**2.1**
TF-RF	0.71	2.7

(c) SVM

Weighting scheme	Average accuracy	Average rank
TF-IDF	0.64	3.2
BM25	0.66	2.4
DELTA TF-IDF	0.70	**1.6**
TF-RF	0.64	2.8

(d) KNN

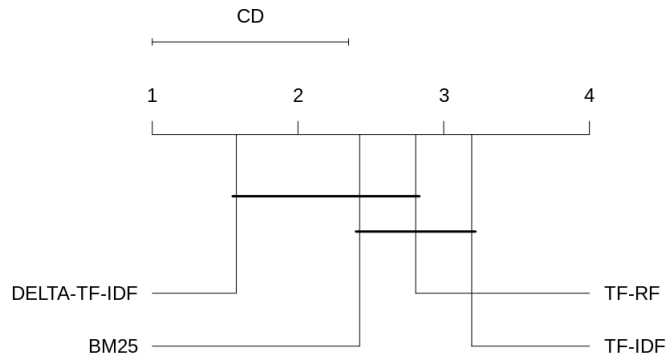


Figure 3.3: Critical distance diagram for the reported KNN classification accuracy.

3.5.3 Feature construction and filtering

In this part, we test the effect of the n-gram construction and pruning on classification. We use the Trains database [24] as input to the Wordification workflow and vary the n-gram and the pruning threshold parameters. Fig 3.4 shows the size of

the resulting feature set according to the choice of n-gram (1 to 5) and the pruning threshold (0 to 50%).

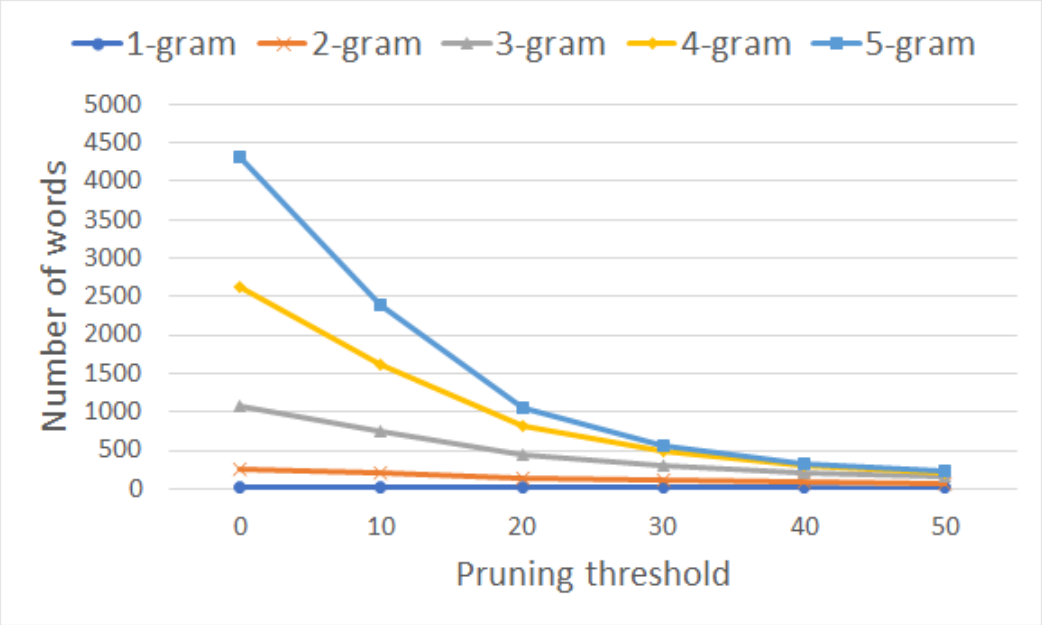


Figure 3.4: Feature set size according to the n-gram and the pruning threshold.

To simplify the analysis, we vary one parameter and keep the other constant. In the first scenario, we vary the n-gram parameter and keep the pruning threshold constant and equal to zero. We perform 5-fold cross-validation measuring the ROC AUC for each combination of classifier, weighting scheme, and n-gram. Fig 3.5 shows the results for each configuration.

We can see from Fig 3.5 that the performance of Random Forest is more stable than that of the other classifiers. This figure shows that the ROC AUC score for SVM and KNN varies a lot with the choice of weighting scheme and the n-gram parameter, although Decision Tree seems more sensitive to the choice of n-gram.

According to Fig 3.5, Random Forest achieved the highest average results using the TF-RF weighting scheme with 2-grams and had the lowest for DELTA TF-IDF. Decision Tree performed best for all weighting schemes using unigrams. In contrast, SVM and KNN achieved the highest scores using DELTA TF-IDF with 5-grams and 3-grams respectively. Another interesting point is that DELTA TF-IDF achieved good results for all classifiers with unigrams. We can also notice that SVM and KNN performed poorly with TF-IDF.

In the second scenario, we analyze the effect of pruning on classification. To do so, we use 5-grams and vary the pruning threshold. Once more, we perform the 5-fold cross-validation and measure the ROC AUC for each combination of classifier, weighting scheme, and pruning threshold. Fig 3.6 shows the results for each configuration. We can see that the average values for Random Forest and

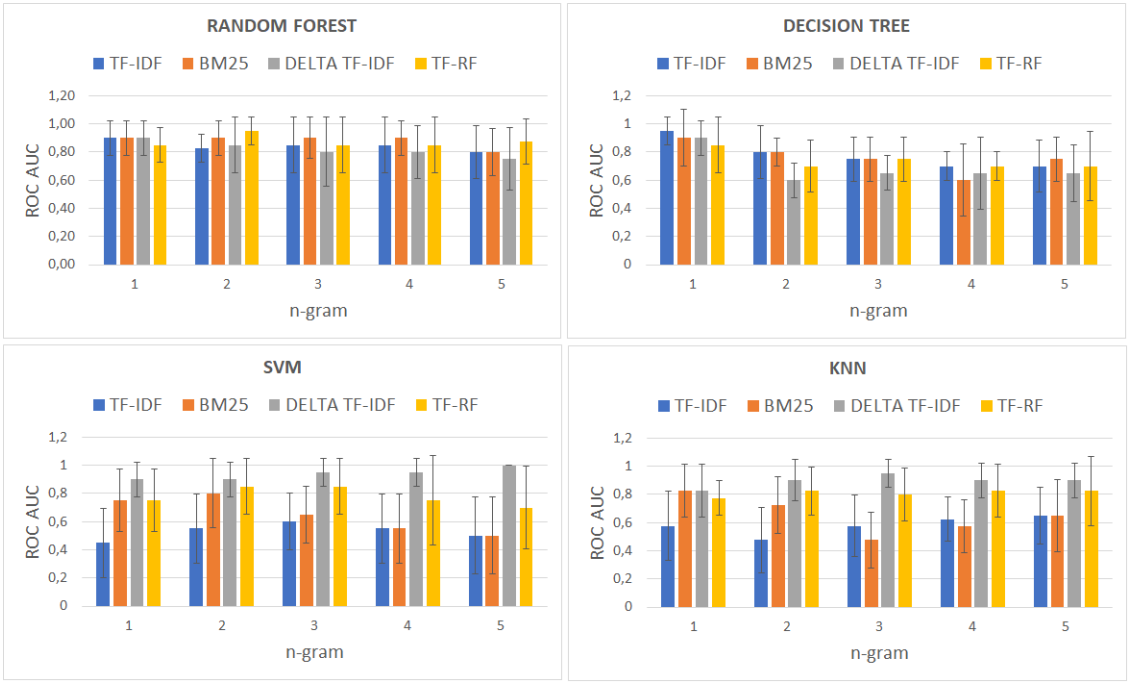


Figure 3.5: Average ROC AUC score of 5-fold cross-validation for each classifier and weighting scheme. We vary the n-gram (1 to 5) and keep the pruning threshold equal to zero.

Decision Tree almost never vary when the pruning threshold changes. We can also notice that the average scores for SVM with TF-IDF and BM25 increase when the pruning threshold rises. A similar pattern is observed for KNN with BM25 and DELTA TF-IDF.

3.6 Discussion

In our experiments, we analyze the results for two metrics: the ROC AUC and accuracy, as in the Wordification paper [3]. The results showed that the choice of weighting scheme can have a great impact on the performance of a classifier. However, a given weighting scheme may not be the best option for all classifiers. For example, Table 3.6 shows that DELTA TF-IDF achieved the highest ROC AUC scores for most of the datasets for Decision Tree, SVM, and KNN, but it did not perform well with Random Forest. We can also notice that Random Forest and Decision Tree performed well using TF-RF, although SVM and KNN performed poorly. According to this table, only Decision Tree achieved the best ROC AUC score for most datasets (7 of 13) using TF-IDF. On the other hand, Table 3.12 shows that all classifiers achieved the best accuracy scores for most datasets using DELTA TF-IDF.

Although it may not be the best solution in every case, DELTA TF-IDF achieved

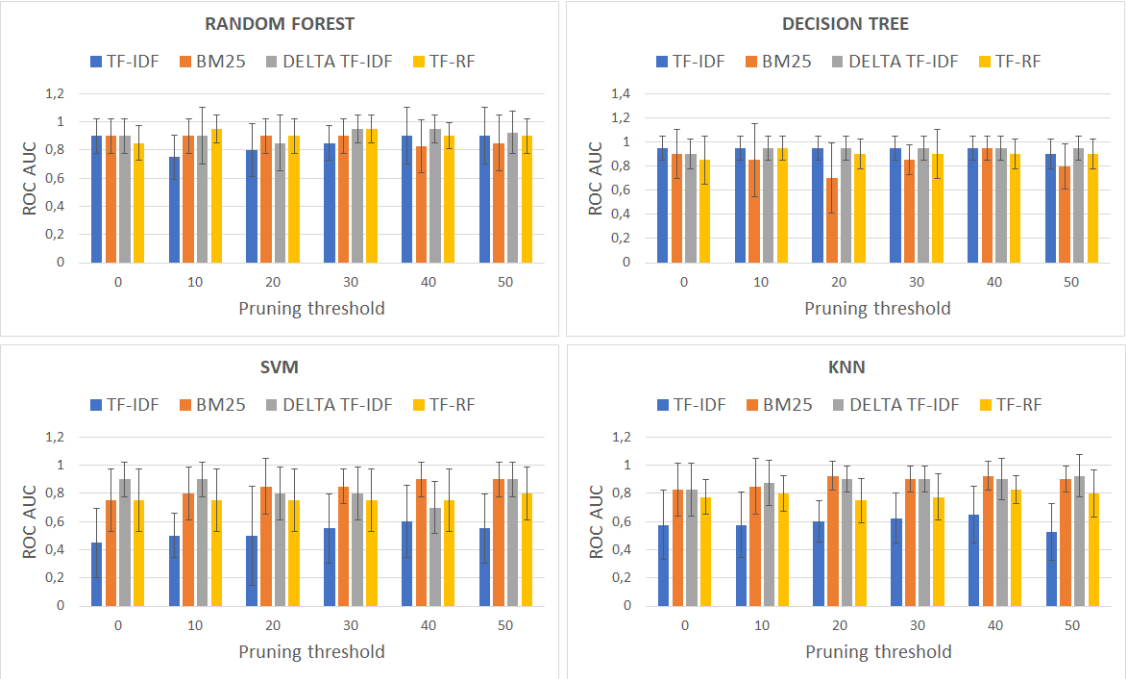


Figure 3.6: Average ROC AUC score of 5-fold cross-validation for each classifier and weighting scheme. We vary the pruning threshold (0 to 50%) and use 5-grams.

the highest average accuracy and ROC AUC scores for most classifiers. Especially for the SVM and KNN, the statistical tests showed a significant difference between DELTA-TF-IDF and TF-IDF for the ROC AUC score. Besides, DELTA-TF-IDF with KNN also performed statistically better than TF-IDF for accuracy. Among all combinations of classifier and weighting scheme, we can see from Tables 3.13 and 3.7 that the highest average score for accuracy (0.76) and ROC AUC (0.77) were achieved by SVM with DELTA TF-IDF.

We statistically compared the best configuration of this work, DELTA-TF-IDF with SVM, with the one in the original Wordification paper [3], TF-IDF with Decision Tree. We used the Wilcoxon signed ranks test with a significance level of $\alpha = 0.05$ to compare the accuracy and ROC AUC scores. The results showed, for both metrics, a statistically significant difference between the two configurations, in favor of DELTA-TF-IDF with SVM.

In Section 3.5.3 we also perform experiments to test the effect of using higher values of n-gram and pruning. We can see from Fig. 3.4 that the number of features increases exponentially when we increase the maximum word length (n-gram). The original Wordification paper [3] suggests the use of pruning to reduce the feature space and improve classification accuracy. However, we could observe only small improvements in the classification performance for isolated cases. Besides, Fig 3.6 shows that the pruning threshold practically did not affect Random Forest and Decision Tree. For this reason, we consider the benefit of pruning irrelevant for

classification performance.

On the other hand, our experiments showed that the choice of the n-gram parameter can considerably change classification performance. For instance, in Fig. 3.5, Decision Tree performed best for all weighting schemes using unigrams. In contrast, SVM and KNN had the best results using a higher n-gram setting with DELTA TF-IDF.

The Wordification paper [3] tested the n-gram parameter with a Decision Tree classifier and the TF-IDF weighting scheme. In these conditions, the paper [3] achieved the best results for most of the datasets using unigrams and stated that larger n-grams of witems only marginally improve classification. According to our result, this statement is true only in certain conditions. As we have seen, Decision Tree did indeed achieve the best scores using unigrams. However, this fact does not hold for SVM and KNN. The downside of using larger n-grams is that this results in longer running times of the propositionalization step due to a larger feature space [3]. In these cases, the use of pruning can be advantageous to reduce the running time to calculate the weighting schemes and build a model.

To sum up, Table 3.14 shows the configurations we recommend for Wordification to achieve the best results.

Table 3.14: Best combinations of classifier, weighting scheme and n-gram.

Classifier	Weighting scheme + n-gram
Random Forest	(TF-RF + 2-grams), (DELTA TF-IDF + 1-gram)
Decision Tree	(TF-IDF + 1-gram), (DELTA TF-IDF + 1-gram)
SVM	(DELTA TF-IDF + 1-gram/3-gram/5-gram)
KNN	(DELTA TF-IDF + 3-gram)

Chapter 4

Conclusions

This study was inspired by works in the text classification and information retrieval fields that explore new weighting schemes to improve performance. Weighting schemes can be considered an important step for Wordification as they produce many text documents based on the instances of a database. The words of these documents are the features that are used in the learning step and therefore, their values affect the performance of a classifier. Although the original Wordification paper used TF-IDF, which is one of the most used weighting schemes, there are other methods in the literature that statistically outperform it. For this reason, we proposed the use of other schemes to improve the Wordification workflow.

As expected, our empirical experiments showed that it is possible to improve Wordification performance by changing the weighting scheme. However, more important than the choice of a weighting scheme is the combination of weighting schemes and classifiers. According to our statistical tests, some classifiers, such as Decision Tree and Random Forest, are very robust and are less impacted by the choice of a weighting scheme. However, the performance of other classifiers, such as SVM and KNN, can greatly differ according to this choice.

According to our experiments, the use of DELTA TF-IDF weighting scheme and SVM with linear kernel resulted in the best scores for most datasets. On the other hand, SVM and KNN performed poorly with TF-IDF. We compared the use of SVM with DELTA TF-IDF with the original proposal of Wordification, where the weights of the features were given by TF-IDF and were used as input to a Decision Tree classifier [3]. Using the Wilcoxon signed ranks test with $\alpha = 0.05$, we showed that the difference of accuracy and ROC AUC for both configurations are statistically significant in favor of SVM with DELTA-TF-IDF.

We also extended the experiment with n-grams and pruning using four options of classifiers (Decision Tree, Random Forest, SVM, and KNN) and four options of weighting scheme (TF-IDF, DELTA TF-IDF, BM25, and TF-RF). The Random Forest classifier presented the most stable performance. Our results also showed

that Decision Tree performs better for most weighting schemes using unigrams. In contrast, SVM and KNN combined with DELTA TF-IDF can have their performance improved by the use of higher n-gram values, although it may result in longer running times. Additionally, we could notice that, for most configurations, increasing the pruning threshold did not improve classification performance. For this reason, it should only be used when the intent is to reduce the running time.

In this work, we only compared TF-IDF with three other weighting schemes, BM25, DELTA-TF-IDF, and TF-RF. Although there are many more options available in the literature, we could demonstrate that it is possible to improve Wordification performance with the right combination of parameters and classifier. For cases where is not possible to test different configurations, we recommend the use of DELTA-TF-IDF with SVM with the linear kernel instead of the traditional TF-IDF.

4.1 Future Works

The original Wordification paper [3] suggested the use of pruning to filter words that appear in less than a predefined percentage of documents. However, there are more sophisticated techniques of feature selection based on chi-square, information gain, etc. It is possible to explore these methods to improve the performance of classification. Besides, we can extend the analysis of n-gram and pruning to other datasets. We also propose the comparison of Wordification with recently proposed propositionalization approaches, such as Cardinalization [4] and the Bottom Clause Propositionalization (BCP) [62].

References

[1] ŢĂRANU, I., OTHERS. "Data mining in healthcare: decision making and precision", *Database Systems Journal*, v. 6, n. 4, pp. 33–40, 2016.

[2] KASSARNIG, V., WOTAWA, F. "Evolutionary propositionalization of multi-relational data", *International journal of software engineering and knowledge engineering*, v. 28, n. 11n12, pp. 1739–1754, 2018.

[3] PEROVŠEK, M., VAVPETIČ, A., KRANJC, J., et al. "Wordification: Propositionalization by unfolding relational data into bags of words", *Expert systems with applications*, v. 42, n. 17, pp. 6442–6456, out. 2015. ISSN: 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2015.04.017>.

[4] AHMED, C. F., LACHICHE, N., CHARNAY, C., et al. "Flexible propositionalization of continuous attributes in relational data mining", *Expert Systems with Applications*, v. 42, n. 21, pp. 7698–7709, 2015.

[5] LACHICHE, N. "Propositionalization", *Encyclopedia of Machine Learning*, pp. 812–817, 2010.

[6] SRINIVASAN, A. "The aleph manual". 2001.

[7] MUGGLETON, S. "Inverse entailment and Progol", *New generation computing*, v. 13, n. 3-4, pp. 245–286, 1995.

[8] ALFRED, R., KAZAKOV, D. "Pattern-Based Transformation Approach to Relational Domain Learning Using Dynamic Aggregation for Relational Attributes." In: *DMIN*, pp. 118–124, 2006.

[9] KNOBBE, A., HAAS, M., SIEBES, A. "Propositionalisation and Aggregates". In: *5th European Conference on Principles of Data Mining and Knowledge Discovery*, p. 277–288, 2001.

[10] KRAMER, S., LAVRAČ, N., FLACH, P. "Propositionalization Approaches to Relational Data Mining". In: Džeroski, S., Lavrač, N. (Eds.), *Relational*

Data Mining, Springer Berlin Heidelberg, pp. 262–291, Berlin, Heidelberg, 2001. ISBN: 9783662045992. Disponível em: <https://doi.org/10.1007/978-3-662-04599-2_11>.

[11] LAVRAČ, N., FLACH, P. A. "An extended transformation approach to inductive logic programming", *ACM Transactions on Computational Logic (TOCL)*, v. 2, n. 4, pp. 458–494, 2001.

[12] KROGEL, M.-A., RAWLES, S., ŽELEZNỲ, F., et al. "Comparative evaluation of approaches to propositionalization". In: *International Conference on Inductive Logic Programming*, pp. 197–214. Springer, 2003.

[13] ŽELEZNÝ, F., LAVRAČ, N. "Propositionalization-based relational subgroup discovery with RSD", *Machine learning*, v. 62, n. 1, pp. 33–63, fev. 2006. ISSN: 0885-6125, 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-006-5834-0>.

[14] KUŽELKA, O., ŽELEZNÝ, F. "Block-wise construction of tree-like relational features with monotone reducibility and redundancy", *Machine learning*, v. 83, n. 2, pp. 163–192, maio 2011. ISSN: 0885-6125, 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-010-5208-5>.

[15] ALSMADI, I., HOON, G. K. "Term weighting scheme for short-text classification: Twitter corpuses", *Neural Computing and Applications*, v. 31, n. 8, pp. 3819–3831, 2019.

[16] PAIK, J. H. "A Novel TF-IDF Weighting Scheme for Effective Ranking". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pp. 343–352, New York, NY, USA, 2013. ACM. ISBN: 9781450320344. Disponível em: <http://doi.acm.org/10.1145/2484028.2484070>.

[17] ROBERTSON, S. E., WALKER, S., JONES, S., et al. "Okapi at TREC-3", *NIST Special Publication*, v. 109, pp. 109, 1995.

[18] MARTINEAU, J. C., FININ, T. "Delta tfidf: An improved feature space for sentiment analysis". In: *Third international AAAI conference on weblogs and social media*, 2009. Disponível em: <https://www.aaai.org/ocs/index.php/ICWSM/09/paper/viewPaper/187>.

[19] LAN, M., TAN, C. L., LOW, H.-B. "Proposing a new term weighting scheme for text categorization". In: *AAAI*, v. 6, pp. 763–768, 2006. Disponível em: <https://www.aaai.org/Papers/AAAI/2006/AAAI06-121.pdf>.

[20] KIM, Y., ZHANG, O. "Credibility Adjusted Term Frequency: A Supervised Term Weighting Scheme for Sentiment Analysis and Text Classification". In: *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, maio 2014.

[21] DOMENICONI, G., MORO, G., PASOLINI, R., et al. "A comparison of term weighting schemes for text classification and sentiment analysis with a supervised variant of tf. idf". In: *International Conference on Data Management Technologies and Applications*, pp. 39–58. Springer, 2015.

[22] SCIAMMARELLA, T., ZAVERUCHA, G. "Weight Your Words: The Effect of Different Weighting Schemes on Wordification Performance". In: *International Conference on Inductive Logic Programming*, pp. 114–128. Springer, 2019.

[23] HUCHARD, M., HACENE, M. R., ROUME, C., et al. "Relational concept discovery in structured datasets", *Annals of Mathematics and Artificial Intelligence*, v. 49, n. 1-4, pp. 39–76, 2007.

[24] MICHIE, D., MUGGLETON, S., PAGE, D., et al. "To the international computing community: A new East-West challenge", *Distributed email document available from http://www. doc. ic. ac. uk/˜ shm/Papers/ml-chall. pdf*, 1994.

[25] DELOBEL, C. "Normalization and hierarchical dependencies in the relational data model", *ACM Transactions on Database Systems (TODS)*, v. 3, n. 3, pp. 201–222, 1978.

[26] FLACH, P., LACHICHE, N. "1BC: A first-order Bayesian classifier". In: *International Conference on Inductive Logic Programming*, pp. 92–103. Springer, 1999.

[27] KRAMER, S., PFAHRINGER, B., HELMA, C. "Stochastic propositionalization of non-determinate background knowledge". In: *Inductive Logic Programming*, pp. 80–94. Springer Berlin Heidelberg, 1998. Disponível em: <`http://dx.doi.org/10.1007/BFb0027312`>.

[28] LAVRAČ, N., DŽEROSKI, S., GROBELNIK, M. "Learning nonrecursive definitions of relations with linus". In: *Machine Learning — EWSL-91*, pp. 265–281. Springer Berlin Heidelberg, 1991. Disponível em: <`http://dx.doi.org/10.1007/BFb0017020`>.

[29] EL-KHAIR, I. A. "Term Weighting". In: LIU, L., ÖZSU, M. T. (Eds.), *Encyclopedia of Database Systems*, pp. 3037–3040, Boston, MA, Springer US, 2009. ISBN: 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_943. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_943>.

[30] KOŁCZ, A., TEO, C. H. "Feature weighting for improved classifier robustness". In: *CEAS'09: sixth conference on email and anti-spam*, 2009.

[31] ROBERTSON, S., ZARAGOZA, H. "The Probabilistic Relevance Framework: BM25 and Beyond", *Foundations and Trends® in Information Retrieval*, v. 3, n. 4, pp. 333–389, 2009. ISSN: 1554-0669. Disponível em: <http://dx.doi.org/10.1561/1500000019>.

[32] LAN, M., TAN, C. L., SU, J., et al. "Supervised and traditional term weighting methods for automatic text categorization", *IEEE transactions on pattern analysis and machine intelligence*, v. 31, n. 4, pp. 721–735, abr. 2009. ISSN: 0162-8828, 0098-5589. Disponível em: <http://dx.doi.org/10.1109/TPAMI.2008.110>.

[33] SPARCK JONES, K. "A statistical interpretation of term specificity and its application in retrieval", *Journal of documentation*, 1972. Disponível em: <https://www.emeraldinsight.com/doi/abs/10.1108/eb026526>.

[34] MIROŃCZUK, M. M., PROTASIEWICZ, J. "A recent overview of the state-of-the-art elements of text classification", *Expert systems with applications*, v. 106, pp. 36–54, set. 2018. ISSN: 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2018.03.058>.

[35] TROTMAN, A. "Learning to Rank", *Information retrieval*, v. 8, n. 3, pp. 359–381, jan. 2005. ISSN: 1386-4564, 1573-7659. Disponível em: <https://doi.org/10.1007/s10791-005-6991-7>.

[36] DEMŠAR, J. "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of machine learning research: JMLR*, v. 7, n. Jan, pp. 1–30, 2006. ISSN: 1532-4435, 1533-7928. Disponível em: <http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf>.

[37] WILCOXON, F. "Individual Comparisons by Ranking Methods". 1945. Disponível em: <http://dx.doi.org/10.2307/3001968>.

[38] FRIEDMAN, M. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance", *Journal of the American Statis-*

tical Association, v. 32, n. 200, pp. 675–701, dez. 1937. ISSN: 0162-1459. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522>.

[39] ASUNCION, A., NEWMAN, D. "UCI machine learning repository". 2007.

[40] HANLEY, J. A., MCNEIL, B. J. "The meaning and use of the area under a receiver operating characteristic (ROC) curve", *Radiology*, v. 143, n. 1, pp. 29–36, abr. 1982. ISSN: 0033-8419. Disponível em: <http://dx.doi.org/10.1148/radiology.143.1.7063747>.

[41] INMAN, R., DAVENPOT, J. "Approximations of the critical region of the Friedman statistic", *Communications in Statistics, Theory and Methods A*, v. 9, pp. 571–595, 1980.

[42] NEMENYI, P. "Distribution-free multiple comparisons". In: *Biometrics*, v. 18, p. 263, 1962.

[43] HARTER, H. L. "Tables of range and studentized range", *The Annals of Mathematical Statistics*, pp. 1122–1147, 1960.

[44] MOTL, J., SCHULTE, O. "The CTU prague relational learning repository", *arXiv preprint arXiv:1511.03086*, 2015.

[45] DEBNATH, A. K., LOPEZ DE COMPADRE, R. L., DEBNATH, G., et al. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity", *Journal of medicinal chemistry*, v. 34, n. 2, pp. 786–797, 1991.

[46] SRINIVASAN, A., KING, R. D., MUGGLETON, S., et al. "Carcinogenesis predictions using ILP". In: *International Conference on Inductive Logic Programming*, pp. 273–287. Springer, 1997.

[47] ZYTKOW, J., RAUCH, J. *Principles of Data Mining and Knowledge Discovery: Third European Conference, PKDD'99 Prague, Czech Republic, September 15-18, 1999 Proceedings*. Springer, 2004.

[48] BERKA, P. "The pkdd discovery challenges on thrombosis data". In: *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2001.

[49] ELOMAA, T., MANNILA, H., TOIVONEN, H. *Principles of Data Mining and Knowledge Discovery: 6th European Conference, PKDD 2002, Helsinki, Finland, August 19–23, 2002, Proceedings*, v. 2431. Springer, 2003.

[50] KNOBBE, A. J. *Multi-relational data mining*, v. 145. Ios Press, 2006.

[51] BOZKURT, M. R., YURTAY, N., YILMAZ, Z., et al. "Comparison of different methods for determining diabetes", *Turkish Journal of Electrical Engineering & Computer Sciences*, v. 22, n. 4, pp. 1044–1055, 2014.

[52] HELMA, C., KING, R. D., KRAMER, S., et al. "The Predictive Toxicology Challenge 2000–2001 ", *Bioinformatics*, v. 17, n. 1, pp. 107–108, 01 2001. ISSN: 1367-4803. doi: 10.1093/bioinformatics/17.1.107. Disponível em: <https://doi.org/10.1093/bioinformatics/17.1.107>.

[53] PAZZANI, M., BRUNK, C. "Finding accurate frontiers: A knowledge-intensive approach to relational learning". 1994. Disponível em: <http://hdl.handle.net/2060/19940029538>.

[54] SCHULTE, O., ROUTLEY, K. "Aggregating predictions vs. aggregating features for relational classification". In: *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 121–128. IEEE, 2014.

[55] KRANJC, J., PODPEČAN, V., LAVRAČ, N. "ClowdFlows: A Cloud Based Scientific Workflow Platform". In: *Machine Learning and Knowledge Discovery in Databases*, pp. 816–819. Springer Berlin Heidelberg, 2012. Disponível em: <http://dx.doi.org/10.1007/978-3-642-33486-3_54>.

[56] ORACLE CORPORATION. "MySQL". 2019. Disponível em: <https://www.mysql.com/>.

[57] FAYYAD, U. M., IRANI, K. B. "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning", *Proceedings of the conference of International Joint Conferences on Artificial Intelligence*, 1993. ISSN: 1045-0823. Disponível em: <https://www.semanticscholar.org/paper/1dc53b91327cab503acc0ca5afb9155882b717a5>.

[58] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine Learning in Python", *Journal of machine learning research: JMLR*, v. 12, n. Oct, pp. 2825–2830, 2011. ISSN: 1532-4435, 1533-7928. Disponível em: <http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>.

[59] FERNÁNDEZ-DELGADO, M., CERNADAS, E., BARRO, S., et al. "Do we need hundreds of classifiers to solve real world classification problems?" *Journal of Machine Learning Research*, 2014. Disponível em: <http://www.jmlr.org/papers/volume15/delgado14a/delgado14a.pdf>.

[60] ZHANG, C., LIU, C., ZHANG, X., et al. “An up-to-date comparison of state-of-the-art classification algorithms”, *Expert systems with applications*, v. 82, pp. 128–150, out. 2017. ISSN: 0957-4174. Disponível em: <https://dx.doi.org/10.1016/j.eswa.2017.04.003>.

[61] YANG, Y., LIU, X., OTHERS. “A re-examination of text categorization methods”. In: *Sigir*, v. 99, p. 99, 1999. Disponível em: <http://people.csail.mit.edu/jim/temp/yang.pdf>.

[62] FRANÇA, M. V., ZAVERUCHA, G., GARCEZ, A. S. D. “Fast relational learning using bottom clause propositionalization with artificial neural networks”, *Machine learning*, v. 94, n. 1, pp. 81–104, 2014.